

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE
ARTIFICIAL INTELLIGENCE CENTRE

**LONG-TERM AUTONOMY
OF MOBILE ROBOTS
IN CHANGING
ENVIRONMENTS**

Habilitation Thesis

Tomáš Krajník

Kralupy nad Vltavou, Mar, 2018

CONTENTS

1	INTRODUCTION	11
2	VISUAL NAVIGATION FROM A LONG-TERM PERSPECTIVE	15
3	ROBUST PERCEPTION AND NAVIGATION	17
3.1	Improving Position Error Model	17
3.2	Trainable Image Features	18
3.3	Research outreach	19
4	SPATIO-TEMPORAL REPRESENTATIONS TO MODEL AND PREDICT ENVIRONMENT VARIATIONS	21
4.1	Spectral Robotic Mapping	22
4.2	FreMEn: Frequency Map Enhancement	24
4.2.1	FreMEn for robot navigation in human-populated environments . . .	25
4.2.2	Information-based Spatio-Temporal Exploration	25
4.2.3	Exploration and exploitation of the spatio-temporal models	27
4.2.4	Warped hypertime	27
4.3	Outreach	28
5	MARKER-BASED LOCALISATION	29
5.1	Research outreach	30
6	CONCLUSIONS	33
A	KEY ARTICLE [1] - JOURNAL OF FIELD ROBOTICS 2010	47
B	KEY ARTICLE [9] - IROS 2018 (IN REVIEW)	73
C	KEY ARTICLE [12] - ROBOTICS AND AUTONOMOUS SYSTEMS 2017	83
D	KEY ARTICLE [14] - ICRA 2014	101
E	KEY ARTICLE [22] - ICRA WORKSHOP ON LONG-TERM AUTONOMY 2016	109
F	KEY ARTICLE [24] - IEEE TRANSACTIONS ON ROBOTICS 2017	115
G	KEY ARTICLE [27] - ROBOTICS AND AUTOMATION MAGAZINE 2017	131
H	KEY ARTICLE [32] - ROBOTICS AND AUTOMATION LETTERS 2017	145
I	KEY ARTICLE [34] - ROBOTICS AND AUTONOMOUS SYSTEMS 2017	155
J	KEY ARTICLE [34] - ROBOTICS AND AUTOMATION LETTERS 2018 (IN REVIEW)	169
K	KEY ARTICLE [40] - JOURNAL OF INTELLIGENT AND ROBOTIC SYSTEMS	179

ACKNOWLEDGEMENTS

I would like to express my gratitude to all of my colleagues, who cooperated with me on the research presented here¹: Jan Faigl for his guidance in early stages of this research, Tom Duckett for his great ideas regarding long-term autonomy and for helping me to write in 'look how easy this is' way, Vojtěch Vonásek for his valuable feedback and help with experiments, Martin Saska for taking me onboard his UAV research and for his inspiring swarm-based approach to student supervision and scientific publishing, Jaime Pulido Fentanes for being the (often overlooked) mother of FreMEn, Matias Nitsche for his help with visual navigation and localisation, Miroslav Kulich for his aid with planning methods, Karel Košnar for interesting dialogues, Marc Hanheide for his patience and for forcing me to integrate the developed methods into the STRANDS system, making them finally useful, Joao Santos for being the best PhD student I had the privilege to supervise so far, Sol Pedre for being a true inspiration during the times I was about to give up, Dan Fišer for co-authoring a paper that no one believed in, Pablo Cristoforis for his cooperation on teach-and-repeat navigation systems, Petr Vaněk for his enthusiastic help with tedious experiments and demonstrations, Grzegorz Cielniak for sacrificing his privacy in the name of science, Vojtěch Spurný for his co-operation on UAV research, and Keerthy Kusumam for her help with computer vision, inspiring discussions, and for the best offer I ever refused. I would like to thank other people who, for reasons unknown, did not appear in the co-author list, like Farshad Arvin, Petr Čížek, Claudio Coppola, Christian Dondrup, Nick Hawes, Jan Chudoba, Peter Lightbody, Oscar Mozos, Michal Pěchouček, Libor Přeučil, Monika Svědihrová, Hana Szüczová, Petr Štěpán, Waqar Qureshi and Yan Zhi. Furthermore, I have to say thanks to ~300 anonymous reviewers, who, similarly to the other 300, had to withstand the onslaught of articles that we managed to produce. Special thanks goes to Tomáš Vintř, who is now struggling with his present reckless supervisor. I am very grateful to my family and close friends for their support.

ووسا ال وادل ا دسا

(Fremen proverb)

During my research, I have been supported by the Czech Technical University grants CTU0706113, CTU0904313 and SGS10/185/OHK3/2T/13. The Ministry of Education of the Czech Republic funded my work by the grants LH11053, MSM6840770038, 2Co6005, MEB111009, 7E08006 and 7AMB18FR018. The Czech Science Foundation supported this work through project 17-16900Y and the European Union funded this work through grants Replicator 216240 and Symbion 216342. Khalifa University supported this research through the MBZIRC 2017 funding. Most of the research presented here was funded through the EU FP7 project STRANDS No. 600623.

¹ by the rank of google scholar co-citations

COPYRIGHT

This work is a compilation of papers published throughout the course of my research efforts. The works included in this habilitation thesis are protected by the copyright of Wiley Periodicals, IEEE, Springe-Verlag, Elsevier, IOS Press and ACM. They are presented and reprinted in accordance with the copyright agreements with the respective publishers. Further copying or reprinting can be done exclusively with the permission of the respective publishers.

©Tomáš Krajník, 2017
©Wiley Periodicals, Inc., 2010
©IEEE 2012–2018
©Springer-Verlag 2014, 2016
©Elsevier 2014, 2016, 2017
©IOS Press 2015
©ACM 2017

ABSTRACT

This habilitation thesis presents research that aims to enable long-term deployment of mobile robots in changing environments. The presented approaches encompass methods that ensure robustness of autonomous visual navigation in outdoor environments for prolonged time periods, spatio-temporal representations that explicitly model the environment changes over time, and supporting software modules that enable robust and accurate robot localisation.

The main contribution of the thesis is a novel approach that allows to incorporate the notion of time into most stationary environment models used in mobile robotics. This is achieved by representing the uncertainty of the environment states not by fixed probabilities, but by probabilistic functions of time, represented in the frequency domain. The method allows to integrate unlimited numbers of sparse and irregular observations obtained during long-term deployments of mobile robots into memory-efficient models that reflect the persistence and recurrence of environment variations. The frequency-enhanced spatio-temporal models allow to predict the future environment states, which improves the efficiency of mobile robot operation in changing environments. In this thesis, we present a series of articles, which demonstrate that the proposed approach improves mobile robot localization, path and task planning, activity recognition, human-robot interaction and allows for life-long spatio-temporal exploration of perpetually-changing environments.

Tato habilitační práce se zabývá problémem dlouhodobé samostatnosti mobilních robotů v prostředích, které podléhají postupným změnám. Práce prezentuje metody, které zvyšují spolehlivost dlouhodobé vizuální navigace mobilních robotů v exteriérech, prostoročasové reprezentace světa, které explicitně modelují změny prostředí v čase, a nástroje, které umožňující spolehlivou a přesnou lokalizaci mobilních robotů.

Hlavním přínosem práce je princip, který umožňuje rozšířit většinu prostorových reprezentací světa používaných v robotice tak, aby tyto zahrnovaly pojem času. Tohoto rozšíření je dosaženo tím, že neurčitost stavů prostředí není modelována jako statická pravděpodobnost, ale jako pravděpodobnostní funkce času, která je reprezentována ve frekvenční doméně. Výše uvedený princip umožňuje integrovat neomezená množství dlouhodobých nepravidelných pozorování stavů daného prostředí robotem do paměťově nenáročných modelů, které efektivně reprezentují základní časové charakteristiky pozorovaných změn. Tyto reprezentace umožňují mobilnímu robotu předpovídat budoucí stavy prostředí, což v dlouhodobém časovém horizontu zvyšuje jeho efektivitu. V této práci ukážeme, že použitím výše uvedené metody se při dlouhodobých nasazeních mobilních robotů zlepšuje efektivita jejich lokalizace, plánování, rozvrhování, rozpoznávání a interakce s lidmi. Dále ukážeme, že kombinace této metody s teorií informace vede k inteligentnímu průzkumu měnících se prostředí.

INTRODUCTION

As robots leave the well-structured worlds of factory assembly lines and enter natural environments, new challenges appear. One of the main problems that mobile robots had to solve was reliable navigation in unstructured and uncertain environments. This challenge gave birth to the field of probabilistic mapping, which enabled the representation of incomplete world knowledge obtained through noisy sensory measurements. The advances of robotic mapping lead to methods that can represent large environments with high fidelity, which resulted in the mobile robots' ability for reliable self-localization and autonomous navigation. However, as robots became gradually able to operate autonomously for longer periods of time, a new problem appeared – that the appearance and structure of natural environments are subject to change [73].

This issue motivated research into methods that are able to suppress the effects of appearance changes, which, in short term, is caused by varying outdoor illumination. One of the popular methods [74] can produce illumination-invariant images captured outdoors by exploiting the fact that the wavelength distribution of the main outdoor illuminant, the sun, is known. This method was used to improve localization and navigation in outdoor environments [75, 76, 77, 78].

However, appearance changes are not caused just by varying illumination, but also by the fact that the environment structure changes over time. Valgren and Lilienthal [79] addressed the question of environment change in visual-based localization by studying the robustness of SIFT and SURF image features to seasonal variations. The paper indicated, that as the robots are gradually becoming able to operate for longer and longer time periods, their navigation systems will have to address the fact that environment itself, not only the illumination, is subject to constant, albeit slow, changes.

Some approaches aimed at solving the problem by using long-term observations to identify which environment features are more stable. Dayoub and Duckett [80] presented a method that continuously adapted the environment model by identifying stable image features and forgetting the unstable ones. Rosen et al. [81] used Bayesian-based survivability analysis to predict which features will still be visible after some time and which features are going to disappear. Carlevaris et al. [82] proposed to learn visual features that are robust to the appearance changes and showed that the learned features outperform the SIFT and SURF feature extractors. Lowry et al. [83] used principal component analysis to determine which aspects of a given location appearance are influenced by seasonal factors and presented a method that can calculate 'condition-invariant' images.

Some researchers proposed to use multiple, condition-dependent representations of the environment. For example, Churchill and Newman [84] clustered different observations of the same place to form "experiences" that characterize the place appearance in particular conditions. McManus et al. [85] used dead reckoning to predict which place the vehicle is close to, loads a bank of Support Vector Machine classifiers associated with that place and uses these to obtain a metric pose estimate.

Methods based on deep learning, which has had a big impact on the field of computer vision, were also applied to the problem of persistent navigation. Neubert and Protzel [86] showed that image descriptors based on Convolutional Neural Networks (CNN) outper-

form the best holistic place recognition methods while being able to handle large viewpoint changes. Sünderhauf [87, 88] also demonstrated impressive results with CNN-based methods. However, the relatively recent outcome of the Visual Place Recognition in Changing Environments, or VPRiCE Challenge [89] indicated that novel, yet classic-feature-based approaches, such as [90] performed better than the CNN-based methods.

However, some of the researchers realized that rather than trying to suppress the effect of environment changes, one can try to learn about the environment from the changes observed and use the knowledge to improve the efficiency of robot operation. Thus, some works use the long-term observations to build models that can predict the appearance of a given location at a particular time. Among these, Lowry et al. [91] applied linear regression techniques directly to the image space in order to predict the visual appearance of different locations in various conditions. Sünderhauf and Neubert [92, 93] mined a dictionary of superpixel-based visual-terms from long-term data and used this dictionary to translate between the appearance of given locations across seasons.

Environment representations that explicitly model the environment dynamics have shown their usefulness not only for the problem of visual-based localization. For example, the authors of [94] demonstrated that representing variations of cells in occupancy grids with hidden Markov models improves the robustness of laser-based localization as well. ‘Dynamic’ or ‘temporal’ occupancy grids that model the environment changes not only improve long-term localization as shown by [95], but also allow to segment dynamic objects [96], and predict their future paths [97]. The authors of [98] demonstrated that reasoning about changes in 3D point clouds obtained over a period of several weeks allows to separate movable objects and refine the static environment structure at the same time.

This thesis presents research that drew inspiration from the aforementioned works, and was aimed at enabling reliable operation of mobile robots over long periods of time. Influenced mainly by the work of [79], we studied the persistence of image features, which resulted in *trainable image features which are robust to environment variations*. The idea of learning from the observed changes, mentioned in [92, 94, 95] influenced the development of the *Frequency Map Enhancement*, which can turn static environment models into models that explicitly represent the environment changes over time. To have the technology for experimental verification of the aforementioned approaches, we had to develop and continuously improve *a robust and reliable teach-and-repeat navigation system*, and *computationally efficient and accurate marker-based localisation system*.

SCHEMATIC OVERVIEW OF THE PRESENTED RESEARCH

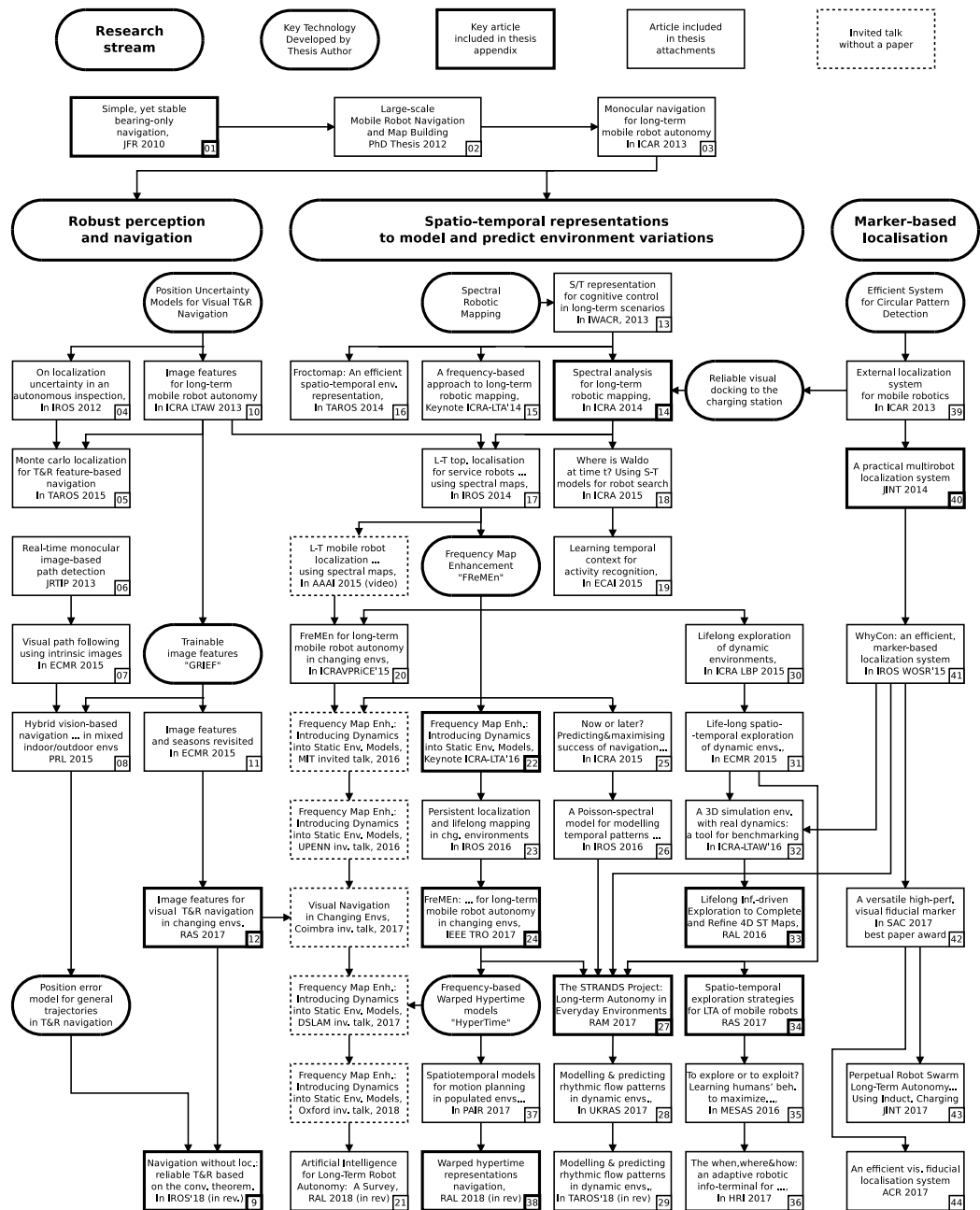


Figure 1: Overview of long-term autonomous systems research performed by the thesis author from 2010 to 2018. Numbers conform to the works in the reference section.

The research presented in this thesis can be divided in three streams, each resulting in a number of scientific articles. A schematic overview of the published works and their relations is shown in Figure 1. Details on the individual research streams are presented in the following chapters.

 VISUAL NAVIGATION FROM A LONG-TERM PERSPECTIVE

Our research aimed at long-term autonomy of mobile robots started with work presented in [1]. Here, we introduced a monocular vision-based teach-and-repeat navigation system, based on a novel mathematical model of robot position uncertainty evolution during visual navigation. The model allowed to simplify visual teach-and-repeat navigation, making it easy to implement, computationally efficient, and robust to camera imperfections and odometry noise. While the navigation method proved to be reasonably robust to illumination changes and seasonal appearance variations, it became clear that environment appearance transitions between summer and winter cause serious problems. Furthermore, the experiments clearly showed that using the same map for navigation during day and night is not possible, because the environment looks in a completely different way.

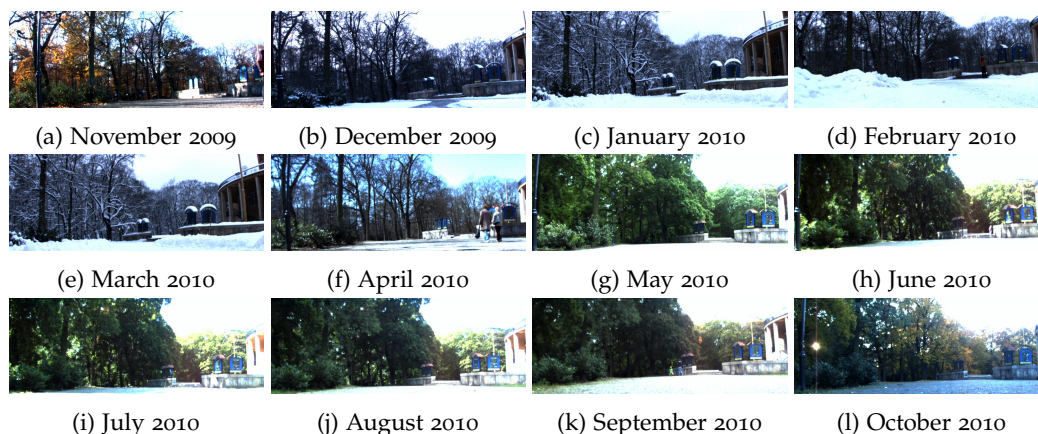


Figure 2: An example of seasonal appearance changes from a robot perspective.

The problem of appearance changes was later studied in the thesis [2], where we proposed to create new maps during autonomous navigation, and use these maps for later autonomous traversals. Prior to an autonomous traversal, the robot calculated mutual information between the current view and all the candidate maps, and selected a map with the highest informative value. This approach was verified in several outdoor experiments performed every month over the period of one year. Since it is not practical to store all the maps from all autonomous runs, we used the dataset we gathered during the aforementioned experiments and we investigated which map could be used at which time. In the end, we came to a conclusion that storing two maps, representing winter and summer environment appearance should be sufficient for outdoor, daylight visual navigation [3].

The main result of the aforementioned papers was the *Position Uncertainty Model for Visual Teach And Repeat Navigation* thoroughly described in [1], and datasets illustrated in Figure 2. This model and datasets allowed us to evaluate the impact of method extensions and improvements on the accuracy and reliability of visual teach and repeat navigation under diverse conditions.

ROBUST PERCEPTION AND NAVIGATION

This research stream aimed to improve the navigation robustness of mobile robots to appearance changes through enhancements of the individual modules of the robots' navigation systems. The research was funded primarily through bilateral collaboration grants, and was performed in close cooperation with our colleagues from the University of Buenos Aires. Most of the works performed extend the teach-and-repeat navigation system introduced in [1]. The first significant result of this research stream is a *general model of robot position uncertainty during teach-and-repeat navigation*, which allows the prediction of navigation system accuracy and reliability given the fidelity of the robot sensors and environment characteristics along the robot path. The second result of this research stream are *efficient trainable image features robust to appearance changes*.

IMPROVING POSITION ERROR MODEL

To improve the localisation accuracy at mission-critical locations, we proposed to take into account the position uncertainty predicted by the aforementioned model. In particular, we incorporated the position uncertainty model into a self-organizing map method, which then could propose robot paths so that during autonomous navigation, the position uncertainty at the desired locations would be minimized. This resulted in significant improvement of the chance that a robot will visit a given location with a desired accuracy [4].

The dependence of our navigation system on accurate dead reckoning was addressed in [5], where an application of particle filter-based localisation allowed the system deployment on flying robots with imperfect velocity estimation.

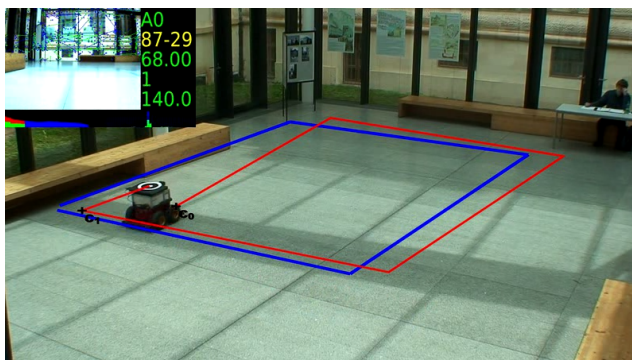


Figure 3: Gradual convergence of the real robot trajectory (in red) to the learned path (in blue). The convergence was predicted by the mathematical model developed in this research stream. See videos at <https://youtu.be/M2krTZCbdaY> and at <https://youtu.be/1ATh0FF48Ao>.

To suppress the dependence of the navigation system on point-based visual features, which seemed to be inherently unstable, we considered to employ segmentation techniques

for path following [6]. To make the path detection robust to the effects of variable illumination and shadows, we proposed to preprocess the camera images by a photometric technique, which allows to calculate illumination invariant image by exploiting the fact that the spectrum of the main outdoor illuminant, the Sun, conforms to the Planck’s law [7].

Finally, we simplified and generalised the mathematical model, so that it would encompass a broader set of navigation techniques. This allowed combination of path-following [6] and map-based navigations [1] into a single framework, introduced in [8]. The insights learned during the development of the combined system allowed further simplification and generalisation of the mathematical model of robot position uncertainty in teach-and-repeat scenarios. This generalised model led to the improved teach-and-repeat navigation system, capable of navigating arbitrary paths in adverse lighting conditions while utilising arbitrary combinations of feature extractors [9].

TRAINABLE IMAGE FEATURES

In [10], we investigated the impact of using various feature extraction methods on the robustness of the navigation to seasonal appearance variations. Much to our surprise, the Binary Robust Independent Elementary Features (BRIEF) [99] outperformed the favoured Scale Invariant Feature Transform (SIFT) [100]. We speculated, that in teach-and-repeat navigation, unlike in other applications, scale- rotation- and viewpoint- invariances are only of a secondary importance. Learning about the BRIEF [99] feature robustness was exploited in our subsequent works related to visual localisation in long-term scenarios.

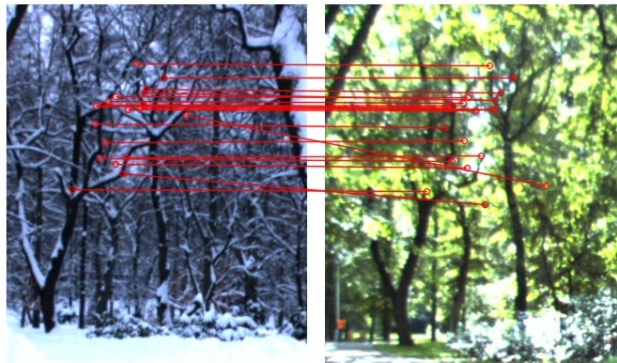


Figure 4: Examples of tentative matches of the developed ‘GRIEF’ image features across seasonal changes, see a video at <https://youtu.be/CEtGG01z4GE>

Further investigation of the reasons behind BRIEF’s performance resulted in an idea of training the sequence of BRIEF binary comparisons for a specific scenario. In [11], we propose a basic scheme for the training, and we compare the resulting feature, coined GRIEF (Generated BRIEF) with the most popular feature detector/descriptor combinations. Encouraged by the feedback on [11], we further improved the training scheme and performed an exhaustive comparison of GRIEF’s performance to several detector/descriptor combinations, including those based on convolutional neural networks (CNN). The resulting investigation [12], performed on 5 different datasets gathered over the period of one year, has indicated that while the GRIEF performance is comparable to CNN-based features, while being faster by two orders of magnitude. The source code for the GRIEF feature training was made public <http://github.com/gestom/grief> and was integrated into the navigation system described in [9].

RESEARCH OUTREACH

The teach-and-repeat navigation methods developed in this research stream were integrated into a user-friendly, robust navigation system [45, 46] that proved its reliability by its repeated success in outdoor robotic contests [47, 48, 49]. The system was also integrated in larger frameworks for field robots [50] as well as for robotic swarms [51]. The navigation system was also extended to work with aerial vehicles [52, 53, 54] and the modules of this UAV system were used by roboticists from top research institutes, including the JPL, KAIST and EPFL. The system was extensively used in experiments that concerned coordination



Figure 5: UAV-UGV team performing autonomous inspection. The ground robot is guided by the navigation system described. See a video at <https://youtu.be/RPCUB6xjQTI>

and cooperation of mixed UGV-UAV teams [55, 56, 57, 58, 59]. Furthermore, the map building module of the navigation system was applied in agricultural robotics [60, 61]. Finally, the simplicity of the navigation method allowed its integration on a single chip, which was performed by Jan Šváb [62] and the most capable student that I supervised, Petr Čížek [63].

 SPATIO–TEMPORAL REPRESENTATIONS TO MODEL AND PREDICT ENVIRONMENT VARIATIONS

Unlike in the previous stream of research, where we wanted to achieve robustness to environment change through improvement of the individual components of the localisation and navigation systems, this research stream aimed at the data structures that represent the environment itself. The core idea of this research is that many of the environment changes are not random, but exhibit certain temporal and spatial properties of the hidden processes that cause them. Through re-observation of the same spatial locations, the properties of these processes can be identified, stored and re-used for long-term predictions. Since many of the contemporary models used in robotics are composed of a set of independent components with binary, but uncertain states (e.g. 2d and 3d grids with cells that are occupied or free, maps with landmarks that are (in)visible, topological maps with edges that are (non)traversable), representation of the temporal properties of the observed changes can be achieved by modeling the uncertainty evolution of these elementary components over time. In other words, instead of representing the uncertainty of the elementary states by a value of probability, which is updated only through a direct observation, one can represent the uncertainty as a probabilistic function of time, which reflects the temporal properties obtained by past observations. In the approach presented, we assumed that in mid- to long-term perspective, some of the environment variations are driven by natural day-night or seasonal cycles, and thus, some of their properties are periodic. Thus, we modeled the uncertainties of environment models' components as combinations of harmonic functions, which were identified by means of spectral analysis.

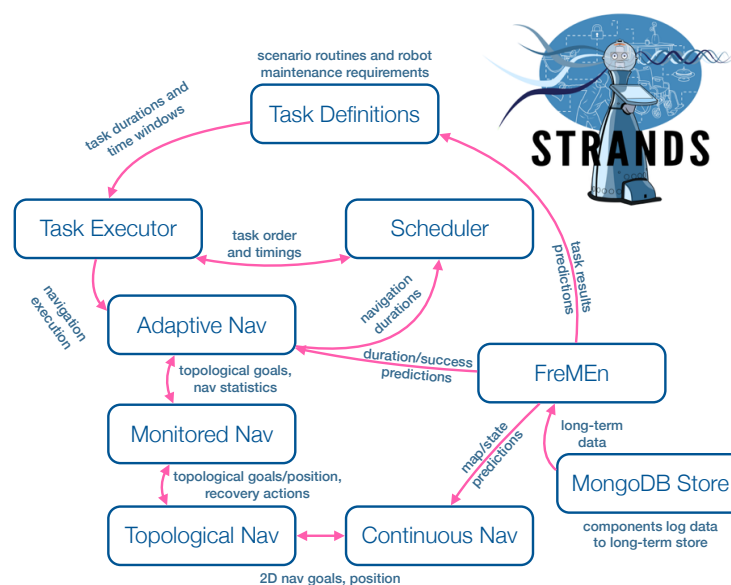


Figure 6: Software structure of the STRANDS system. Courtesy of [27].

This research stream was conducted primarily within the EU FP7 project Spatio-Temporal Representations and Activities for Cognitive Control in Long-term Scenarios (STRANDS), and the results would not have been achieved without the wonderful STRANDS team. The STRANDS project aimed to create robots, that would be able to run for months in dynamic human environments while understanding the 3D space and how it changes over time. The core method developed for spatio-temporal modeling, called Frequency Map Enhancement (FreMEn), processed data that the robot gathered in the past, created spatio-temporal models and used these to provide predictions to the planning and localisation modules of the robot navigation system, see Figure 6. This endowed the STRANDS robots with the ability to learn the long-term dynamics of the surrounding world and to use this knowledge to improve the efficiency of their operation over time. A comprehensive summary of the FreMEn approach with illustrative videos, links to papers and source code compatible with the Robot Operating System (ROS) is available at <http://fremen.uk>.

SPECTRAL ROBOTIC MAPPING

The first version of the approach, called *Spectral Robotic Mapping*, was based on the application of Fast Fourier Transform (FFT) over the binary states constituting the spatial representations. The core idea is to process the past observations of a particular state by FFT and select the most prominent spectral components, which correspond to the cyclic behaviour of the state’s variations. By calculating an inverse FFT from the selected spectral components, we obtain a function of time that not only represents the state’s past history, but can also be used to predict its future with a given confidence, see Figure 7. A presentation of the preliminary idea [13] was met with a positive feedback, which influenced the later research. A complete formulation of the proposed technique, along with an analysis of its ability to compress long-term observations and to predict future environment states, was presented in [14] and in an invited talk [15]. The experiments

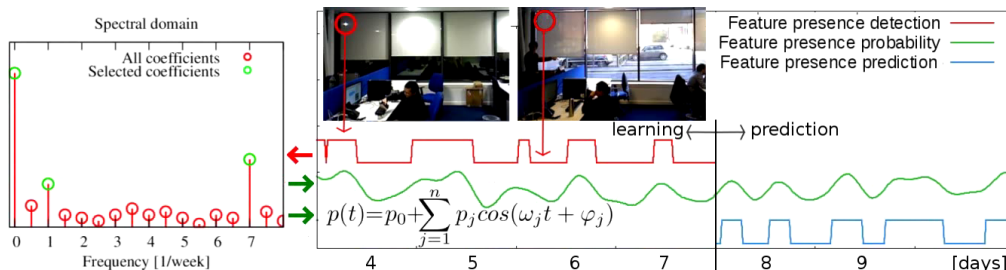


Figure 7: Spectral Maps for visual localisation: The observations of image feature visibility (centre, red) are transferred to the spectral domain (left). The most prominent components of the model (left, green) constitute an analytic expression (centre, bottom) that represents the probability of the feature being visible at a given time (green). This is used to predict the feature visibility at a time when the robot performs self-localisation (blue). See also https://youtu.be/Qw1kS_5zVwE

performed in [14] indicated that the use of FFT allows for efficient representation of the temporal domain, achieving compression rates up to 1:100. By combining this temporal modeling technique with an efficient, octree-based spatial model [101], we obtained a 4d (3d + time) representation, called FROctomap [16]. The experiments presented in [16] demonstrated that the FROctomap method was able to integrate week-long observations of a small university office into an efficient spatio-temporal model, achieving compression rates up to 1:10000. The experimental evaluations performed in the aforementioned works indicated that the *Spectral Robotic Mapping* can represent long-term spatio-temporal dynamics with low memory requirements and provide accurate predictions of future states.

However, aforementioned works were aimed solely at the spatio-temporal representation itself and they did not investigate the impact of the method on the efficiency of mobile robot operation.

Thus, in our next step, we have investigated the utility of the aforementioned spectral representation for robot localisation [17], task planning [18] and activity recognition [19]. The work presented in [17] concerns topological localisation of a mobile robot operating on a 24/7 basis in an open-plan office. Here, the robot uses a laser rangefinder-based navigation stack to perform regular patrols capturing visual appearance of several designated locations every 10 minutes for one week. Using the data gathered, the robot builds a temporal model of image features' visibilities at each location. Using these models, the robot can predict which features are going to be visible at which location and time. In other words, the robot can generate a time-specific map of the most likely visible features at each of these locations. After one week and after three months, the robot was repeatedly placed at the mapped locations at various times of the day and it was supposed to determine its location. To do that, the robot matched the BRIEF [10] image features extracted from its onboard camera image with the features predicted to be visible at that location and time. The experimental results, presented in [17], demonstrate that the use of *Spectral Maps* to predict the feature visibility resulted in significant reduction of the localisation error. The summary of the approach is indicated in Figure 7 and in a video accessible through https://youtu.be/Qw1kS_5zVwE.

To verify the utility of the approach for planning, we applied the concept to a robotic search scenario [18]. In this scenario, a robot has to find a person in an apartment or an office as quickly as possible. To do that efficiently, the robot has to take into account the time it takes to navigate between various locations where the person can occur as well as the probability of the person presence at these locations. The likelihood of the person presence at different locations can be represented either as a static probability, neglecting the person dynamics, or by *Spectral Map* concept which reflects the person's daily and weekly routines. To perform the experimental evaluation, we build two different simulated environments, which were fed by real-world data from a continuously monitored apartment [102] and a university office. Then, we let the robot perform repeated searches (over the simulated period of several weeks) while using different temporal models predicting the person presence. The experiments indicated that compared to other probabilistic models, the use of the *Spectral Maps* reduced the time required to find the person by 25% to 65% [18].

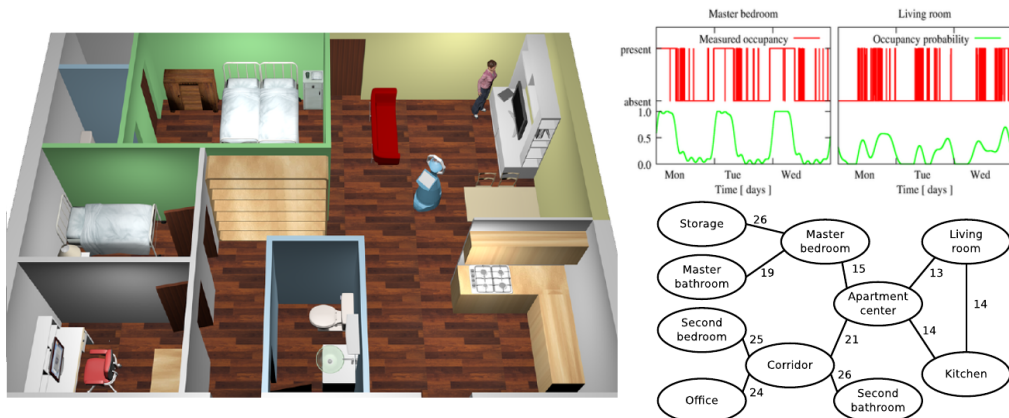


Figure 8: Simulation of the apartment used in [18,19,31,32,34]. 3D structure is left, topological layout and temporal models of the person presence are on the right. The structure of the apartment and the behaviour of the person was obtained from real-world data of the CASAS dataset [76].

The data from the aforementioned two environments included not only the locations of the people at particular times, but also their activities [19]. To investigate if the temporal models could improve the robot’s ability to recognize the activity a person is performing, we used these models as prior probabilities for an activity classifier used by the simulated robot. Every time the robot classified an activity, it added the classification result to its temporal model, which was then used as a prior probability for the classifier. Thus, as the robot observed the people’s activities in the simulated environments over time, it learned about the typical activity patterns and gradually reduced the classification error. In [19], we performed comparison of several temporal models acting as priors for the classification, concluding that the best performing models were based on *Spectral Maps* and Gaussian Mixtures.

While the aforementioned experiments demonstrated that the *Spectral Robotic Mapping* can result in improvement of the robot performance through explicit representation of the environment dynamics, the approach has a major limitation. This limitation comes from the fact that the method is based on the Fast Fourier Transform (FFT), which is based on the assumption that the observations of the environment states can be performed frequently and on a regular basis. However, regular sampling over long-time periods is hard to achieve even in laboratory settings. Furthermore, the requirement of regular observations means that prior to the robot deployment, it has to spend considerable time learning the environment dynamics by re-observation of different individual locations on a regular basis. Once this learning phase is over and the robot starts to perform its tasks in an not-exactly regular manner, it cannot update its spatio-temporal models and thus it cannot adapt to variations that were not present during the learning phase. Thus, the predictive capability of its models will deteriorate over time, which will negatively affect the efficiency of robot operation in long-term deployments. To allow the robot to cope with the changing dynamics through life-long learning, we had to develop a method that can build the spatio-temporal model incrementally from sparse and irregular observations.

FREMEN: FREQUENCY MAP ENHANCEMENT

To deal with the aforementioned problem, we have abandoned the use of the Fast Fourier Transform and we have used the original definition of the Fourier Transform, which allows to derive a set of formulas that can update the frequency spectrum incrementally from irregularly-sampled data.

The core concept of the *Frequency Map Enhancement* (FreMEn) was first mentioned at the Workshop of Visual Place Recognition for Changing Environments [20], where it received significant attention that resulted in a series of talks at top robotics laboratories, including MIT and Oxford. The feedback provided at these presentations allowed us to put this research in a broader perspective and integrate it in a survey paper authored primarily by members of the Oxford robotics group [21]. Further details of the method were introduced in a keynote talk during the Workshop on Long-term Autonomy that took place during the ICRA 2016 conference [22]. In this presentation, we wrapped up the research performed so far, outlined the possible applications and introduced the concept of Information-based Spatio-Temporal Exploration that is enabled by the *FreMEn* method.

In a subsequent paper, we integrated the FreMEn into a occupancy grid representation that forms the core of the Robot Operating System (ROS) navigation stack. The ROS navigation stack was changed so that the robot creates temporally local maps during routine operation and integrates them into a global spatio-temporal representation. This representation is then used to generate time-dependent occupancy grids that are essential for robot localisation and path planning. Then, using data gathered over the period of one week, we have evaluated the impact of the aforementioned FreMEn component on the efficiency of the robot localisation and planning [23].

Finally, we integrated the final version of the FreMEN method as a general component into the Robot Operating System (ROS). This finalised version, along with newly-gathered data, was then used to significantly extend the initial evaluations described in [14, 16, 17, 23] and present them in a coherent way in a research paper in one of the top robotics journals [24].

FreMEN for robot navigation in human-populated environments

To investigate the *FreMEN* ability to process data that were gathered in a highly irregular manner, we applied it to the problem of topological navigation. Here, a robot has to construct the most fail-safe path between two locations in a topological map, i.e. it has to create a sequence of edges leading from a start to a goal node. Each edge of the topological map is associated with data indicating the time of navigation failures or successes and thus, one can use these past data to predict the future success of each edge traversal. This allows to calculate the probability of success of every existing path in the topological map and therefore, the mobile robot can construct a path that will maximise the likelihood of reaching the desired goal location. However, the success of traversing a given edge is influenced by the environment the edge leads through, e.g. an edge leading through a door can be traversed only if the door is open and traversing an edge leading through a corridor might be difficult during busy times. Since the environment states are influenced by the current time, an edge traversal success is likely to be time-dependent as well. In [25], we build temporal models of edge traversals from data gathered by a mobile robot over the period of two months and let the method predict the navigation success of the robot. The experiments indicated, that the *Frequency Map Enhancement* method identified the temporal behaviour of the navigation success being influenced by time of the day and day of the week [25]. Furthermore, the experiments have shown that by choosing the right time to navigate to a certain location, the overall navigation success rate can be significantly improved.

One of the challenging problems for a navigating robot are human crowds, which are either stationary, or form ‘flows’, where most of the people move in a similar direction and velocity. A robot that wants to co-exist with people in their natural environment, has to be able to either avoid the crowded areas, or even better, plan its motion in a way that would allow to naturally blend in the people movement flow [103]. However, representing human crowds merely by their absence or presence is somewhat insufficient. Having the ability to predict the number of people within a given area is more useful, which required the extension of the model towards representation of Poisson, not only Bernoulli processes [26]. This way, the robot could predict the number of people in certain areas of interest.

The *Frequency Map Enhancement* was integrated into the STRANDS [27] system, which controlled two autonomous robots in two human-populated environments (a care home in Austria and a security office in UK). The component allowed both robots to gradually learn the environment dynamics, leading to continuous improvement of their efficiency over time [27]. In both scenarios, the robots were autonomously operating over the period of several weeks.

The problem of predicting the movement of pedestrians by FreMEN was further investigated in [28, 29], which take into account not only the presence of people but also their most likely direction at a given time.

Information-based Spatio-Temporal Exploration

The ability of the FreMEN to deal with sparse and irregular data opened another interesting question: How should a mobile robot gather its observations to build, maintain and refine spatio-temporal models in an efficient way?

The aforementioned problem of active building of environment models is, in the context of mobile robotics, referred to as exploration. While the state-of-the-art exploration methods consider the environment as being static, we have to take into account the world dynamics as well. This adds an extra temporal dimension to the explored space and causes the exploration to be never-ending data-gathering process that attempts to keep the robot's environment model up-to-date. While the static environment exploration methods are primarily concerned with the problem of *where* to gather the next batch of data, spatio-temporal exploration has to also reason about *when* to do it. To deal with this problem, we applied information-theoretic exploration to FreME_n representations. The bulk of the research on Spatio-Temporal Exploration was performed in close cooperation with João Santos, whom I had the privilege to supervise during his PhD studies.

An initial attempt to apply information-theoretic exploration to spatio-temporal models was presented during the late-breaking research session of the ICRA conference [30]. While the provided feedback was positive, it became that there are two core questions to be addressed: Which temporal model to use and what strategy to employ when scheduling the robot observations? An initial insight into these two problems performed in [31] revealed that the most promising combination is the use of Monte-Carlo scheduling scheme over entropy-based FreME_n model.

Since exploration is an active process, where the robot takes decisions on how, where and when to perform observations, the data it gathers are different in every trial. Thus, one cannot simply compare the different exploration strategies and spatio-temporal models on datasets. On the other hand, it is important to ensure fair comparison and repeatability of the evaluations performed. We addressed this problem by building a dynamic simulator, which reflects the evolution of a real environment over the period of several weeks, obtained by ambient sensors. This tool allows to replay the history of changes in a real environment, while providing sensory inputs for a simulated exploring robot. The spatio-temporal model that the robot build can be then compared with the simulation, providing a quantitative evaluation of the exploration strategy [32].

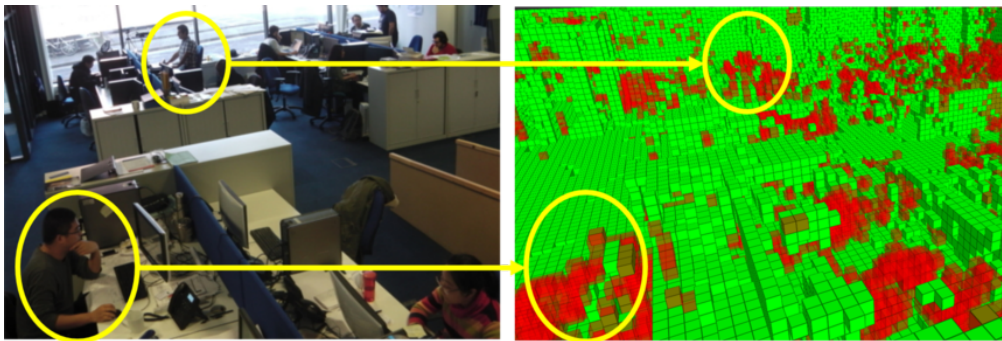


Figure 9: A 4d (3d space + time) occupancy grid created by the robot during one of the exploration experiments. The green cells correspond to areas that are stationary and the red cells relate to areas that exhibit changes occurring on a daily basis. See also <https://youtu.be/do0MpLEjNzw>

The simulation was essential to build and evaluate a robotic system capable of creating, maintaining and refining 4d (3d + time) spatio-temporal maps of the environment [33]. The spatio-temporal map created by the aforementioned system is shown on Figure 9 and is available as a video at <https://youtu.be/do0MpLEjNzw>. Another work that builds on the initial paper on spatio-temporal exploration [31], is presented in [34]. In particular, [34] provides a detailed insight into the efficiency of exploration methods based on various representations and exploration strategies, including curiosity- and novelty-driven exploration behaviours.

Exploration and exploitation of the spatio-temporal models

However, the aforementioned exploration papers were concerned only with the faithfulness of the spatio-temporal models created by the different exploration methods. Typically, the main task of a mobile robot is not to create an accurate representation of the outer world, but to perform useful tasks that exploit the knowledge contained in these models. Since the time a mobile robot spends with exploration cannot be spent with the task the model was built for, the robot has to carefully balance model exploration and exploitation. In the STRANDS project [27], one of the tasks the robot performed was an info-terminal service. Here, the robot was provided with a set of locations, where it could act as an interactive information stand. The robot had to determine when and where to provide the info-terminal service in order to maximise its usefulness, measured by a number of interactions. For that, the robot tied each of the predetermined location with a temporal model representing the probability of an interaction at a given time. As indicated in [31], building and maintaining these models requires that the robot visits these locations when the probability of the interaction is around 0.5, because that maximises the information gain obtained by measuring if an interaction occurs. However, since the robot is trying to maximise the number of interactions, it should visit these locations when the probability of interaction is close to 1. This represents an exploration/exploitation problem over the spatio-temporal space representing the likelihood of human/robot interaction.

To get an insight into the problem, we used the data gathered during one of the STRANDS deployments to build a simulator of people interactions in one of the deployment sites. Then, we performed several simulations, combining different exploration/exploitation strategies, temporal models and planning horizons [35]. Based on the simulated results, we have selected a number of promising strategies and integrated these into the STRANDS system. The results have shown, that employment of these exploration/exploitation strategies allowed the deployed robot to gradually increase the number of obtained interactions over time [27, 36].

Warped hypertime

The original purpose of the FreMEen, i.e. to introduce the notion of dynamics into environment models that compose of independent binary states, became its fundamental limitation. In a real robotic system, there are important non-binary variables that would be useful to predict, such as the robot velocity or number of people within a given area. Furthermore, considering the elements of fine-grained environment models, such as occupancy grids, as independent results in inefficient use of memory and incorrect predictions. To deal with this, we proposed yet another method for introducing time into discrete and continuous spatial representations used in mobile robotics. Unlike FreMEen, the proposed method does

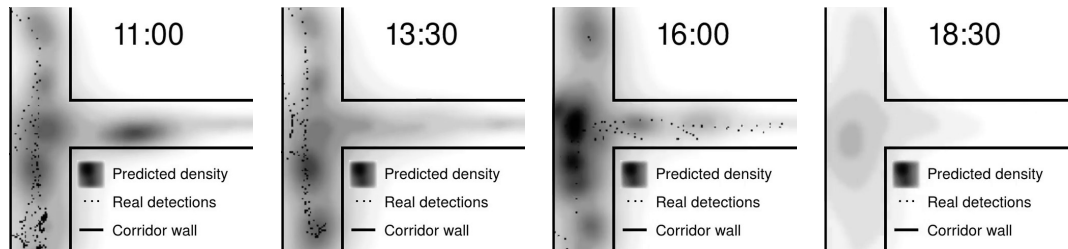


Figure 10: Spatio-temporal model of people presence in a corridor of the University of Lincoln, UK at various times of a day [38]. See a video at <https://youtu.be/4SW4j7DDxYE>.

not treat time and space separately, and its continuous nature respects both the temporal and spatial continuity of the modeled phenomena. The method extends the given spatial model with a set of wrapped ‘Hypertime’ dimensions that represent the periodicities of observed changes identified by the FreMEn. By performing clustering over this extended representation, we obtain a model that allows us to predict future states of both discrete and continuous spatial representations. We applied the method to pedestrian data gathered over several weeks and demonstrated that it efficiently predicts spatial distribution of people within the monitored area [37]. Later on, we applied the Hypertime method to several long-term datasets and demonstrated that it can predict future states of representations with different dimensions, performing better or equal to FreMEn [38].

OUTREACH

The main impact of this research stream is related to the STRANDS project, where the FreMEn method became an important component of the software systems responsible for robot planning and navigation. Therefore, many of the works published by the STRANDS consortium during 2016-2017 benefited from the reliability of the robot, influenced by the FreMEn methods. Furthermore, the simplicity and versatility of the spectral representations attracted considerable attention, resulting in a number of invited talks. To facilitate the use of the method by other researchers, we provide the method overview, most relevant papers, link to datasets, and method source codes at <http://fremen.uk>.

MARKER-BASED LOCALISATION

In our effort to achieve long-term autonomy of mobile robots, we often encountered technical issues related to accuracy reliability and computational efficiency of mobile robot localisation. Often, our robots had problems localising or navigating at specific locations, or the accuracy and reliability of their standard navigation methods was not sufficient for the task required (e.g. docking to a recharging station or passing through a narrow corridor). While the aforementioned problems were not subject of our research, we still had to make sure that all of these ‘technical’ problems are resolved, so the robots could reliably operate on their own. One of the standard practices to help a robot to operate reliably is to augment the environment with elements that provide the robot with extra information. For this purpose, we tried to use printable QR codes and data-matrices, but the detection of these elements was somewhat unreliable and computationally costly.

Therefore, we developed a method for efficient detection and accurate localisation of circular markers, which was used in a number of components throughout our projects. This method was an enhancement of a previous system intended for efficient relative localisation of robot swarms [64]. In [39], we have introduced the core method, and provided experiments which indicated the method’s accuracy and computational efficiency. This method became the core of an essential component of the STRANDS system providing the robots with the ability to accurately and reliably dock to their charging stations. During the STRANDS project, the robots used this method more than 100 000 times – see an example of docking at <https://youtu.be/btYD1HJo8yA>. Furthermore, we used the marker detection as an external localisation system to evaluate the several experiments related to localisation and navigation accuracy. The method was further improved by enhancing the accuracy of the localisation, and its detailed description and mathematical model was provided in [40]. Furthermore, we integrated the method into ROS, released its source codes and advertised it in [41]. Compatibility with the ROS led to further usage of the method in the STRANDS system, e.g. as special cards used for human-robot interaction, see <https://twitter.com/lindastrands>.

It became clear that detection and location of the markers was not sufficient, and that it’s highly desirable to extend the method so that it’s capable to distinguish the individual markers from each other. After several attempts to introduce different types of identification schemes based on slight alterations of the circular pattern shape, our experimental evaluations indicated that the most suitable scheme to encode an ID into the circular marker is through the use of ‘Necklace codes’ [104]. The use of the Necklace codes allowed a smooth extension of the circular marker, while preserving its computational efficiency, detection range and backward compatibility. Our work, that introduced this novel marker [42], was awarded as the best paper at the Symposium on Applied Computing in 2017. Later on, the marker was integrated into an platform that allows continuous, perpetual operation of a robotic swarm and automated evaluation of the swarm behaviours over long periods of time [43]. The work [44] extends [42] in several aspects that further improve the accuracy of the detection and reliability of the identification.

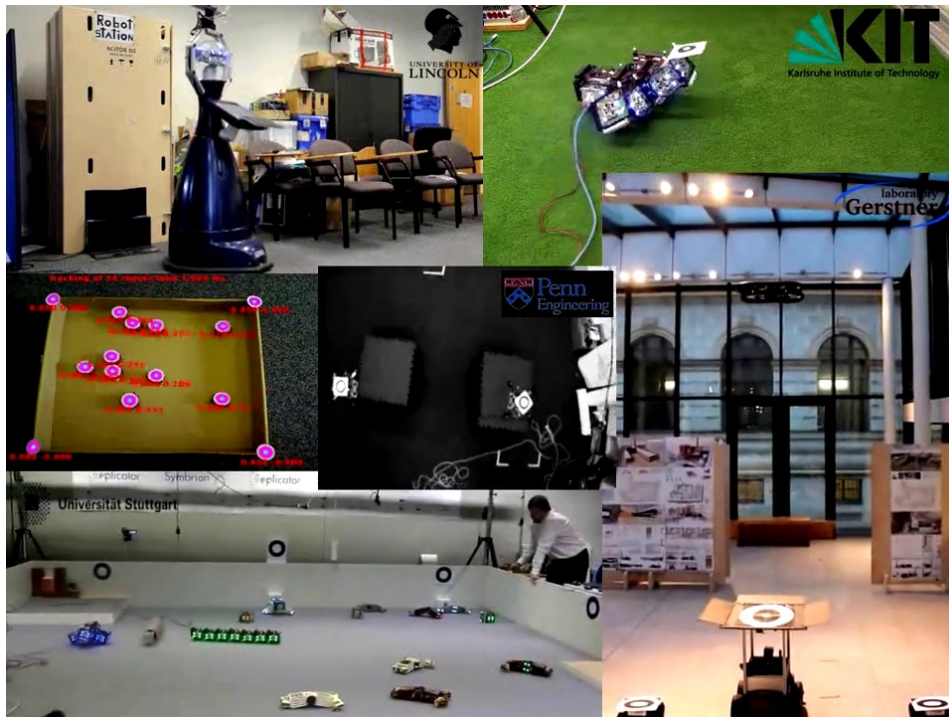


Figure 11: Examples of WhyCon localisation system applications. Top-left shows autonomous charging in EU project STRANDS - note the tree o's in the ROBOT STATION tag. Top-right shows fitness evaluation for self-evolving robots in EU project SYMBRION. Centre and bottom-right shows relative localization of UAV-UGV formations in CZ-USA project COLOS. Bottom left shows energy source localization in EU project REPLICATOR. Left part demonstrates multi-robot localization used for the perpetual swarm platform capable of long-term swarm experiments.

RESEARCH OUTREACH

The marker-based localisation method introduced in [40] became popular in the robotics community, which is reflected in citations of the articles [40],[39] and [64]. The system was used in several works that required relative localisation of UAV and mixed UAV-UGV formations in [56, 57, 58, 59, 65, 66]. Its components were used in the multi-UAV system, which achieved tremendous success at the Mohammed bin Zayed Robotic International Contest in 2017 [67, 68, 69]. Furthermore, it was extensively used for micro-robot swarms [51, 70], leading to a system capable of artificial pheromone simulation [71, 72].



Figure 12: Typical inhabitants of Lincolnshire using an interaction card to get a selfie by the STRANDS robot.

CONCLUSIONS

This thesis provides an overview of the research I had the luck to participate on during the last 10 years. The purpose of the research was to bring robots closer to being really useful by allowing their longer autonomous operation in natural environments. We tried to achieve this goal by improvements of the methods that are essential for the task of autonomous navigation and localisation. Our focus was to make these methods robust to the effects of gradual environment changes that negatively affected contemporary robotic systems. However, we realised that rather by trying to fight the change, robots should embrace it, for change is the essential process of all existence [105].

Thus, we developed a method that allows a mobile robot to have a basic understanding of the environment changes from a long-term perspective. The method endows robots with a basic comprehension of temporal structure of the world, which is partially driven by naturally-cyclic processes. Understanding the world's temporal structure allows the robots to predict what is more likely to happen when and use this knowledge to their advantage. In a series of experiments lasting several months, we demonstrated that the method improves mobile robot mapping [14, 16, 24], localization [17, 20, 23, 24], path planning [25], robotic search [18], activity recognition [19], patrolling [31], exploration [33, 34], task scheduling [35] and human-robot interaction [36]. Moreover, it allows for temporal-context-based novelty and anomaly detection [14, 24]. The method outperformed other temporal models (e.g. Gaussian processes) both in terms of prediction accuracy and computational efficiency [18, 19, 26, 31, 34, 38].

In the future, I would like to extend the spatio-temporal modeling approach so that it would be able to take into account sensor noise in way similar to standard Bayesian update applied to states with fixed probabilistic values. Furthermore, I would like to integrate the spatiotemporal modeling technique into a continuously operating outdoor robot, thus joining the research streams on robust visual navigation and spatio-temporal models.

PUBLISHED WORKS REGARDING LONG-TERM AUTONOMY

- [1] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, pp. 511–533, 2010.
- [2] Tomáš Krajník, *Large-scale Mobile Robot Navigation and Map Building*, Ph.D. thesis, Czech Technical University in Prague, 2012.
- [3] T. Krajník, S. Pedre, and L. Přeučil, "Monocular navigation for long-term autonomy," in *International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–6.
- [4] J. Faigl, T. Krajník, V. Vonásek, and L. Přeučil, "On localization uncertainty in an autonomous inspection," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 1119–1124.
- [5] M. Nitsche, T. Pire, T. Krajník, M. Kulich, and M. Mejail, "Monte carlo localization for teach-and-repeat feature-based navigation," in *Towards Autonomous Robotic Systems (TAROS)*. Springer International Publishing, 2014, pp. 13–24.
- [6] P. Cristóforis, M. Nitsche, T. Krajník, and M. Mejail, "Real-time monocular image-based path detection," *Journal of Real-Time Image Processing*, vol. 11, no. 2, pp. 335–348, 2016.
- [7] T. Krajník, J. Blažíček, and J.M. Santos, "Visual road following using intrinsic images," in *European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.
- [8] P. Cristóforis, M. Nitsche, T. Krajník, T. Pire, and M. Mejail, "Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments," *Pattern Recognition Letters*, vol. 53, pp. 118–128, 2015.
- [9] T. Krajník, F. Majer, L. Halodová, and T. Vintr, "Navigation without localisation: reliable teach and repeat based on the convergence theorem," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018, In review.
- [10] T. Krajník, P. Cristóforis, L. Přeučil, and M. Mejail, "Image features for long-term mobile robot autonomy," in *ICRA Workshop on Long-Term Autonomy*, 2013.
- [11] T. Krajník, P. Cristóforis, M. Nitsche, K. Kusumam, and T. Duckett, "Image features and seasons revisited," in *European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–7.
- [12] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, vol. 88, pp. 127–141, 2017.
- [13] T. Duckett, M. Hanheide, T. Krajník, J.P. Fentanes, and C. Dondrup, "Spatio-temporal representation for cognitive control in long-term scenarios," in *International IEEE/EPSSRC Workshop on Autonomous Cognitive Robotics*, 2014.
- [14] T. Krajník, J.P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3706–3711.

- [15] T. Duckett and T. Krajník, "A frequency-based approach to long-term robotic mapping," in *ICRA Workshop on Long-term Autonomy*. IEEE, 2014, Invited talk.
- [16] Tomáš Krajník, Joao Santos, Bianca Seemann, and Tom Duckett, "Proctomap: An efficient spatio-temporal environment representation," in *Advances in Autonomous Robotics Systems*, 2014, p. 269.
- [17] Tomáš Krajník, Jaime P Fentanes, Oscar M Mozos, Tom Duckett, Johan Ekekrantz, and Marc Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 4537–4542.
- [18] Tomáš Krajník, Miroslav Kulich, Lenka Mudrová, Rares Ambrus, and Tom Duckett, "Where's waldo at time t? using spatio-temporal models for mobile robot search," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2140–2146.
- [19] Claudio Coppola, Tomáš Krajník, Tom Duckett, and Nicola Bellotto, "Learning temporal context for activity recognition," in *22nd European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2016, vol. 285, p. 107.
- [20] Tomáš Krajník, Jaime Pulido Fentanes, João Santos, Keerthy Kusumam, and Tom Duckett, "FreMEN: Frequency map enhancement for long-term mobile robot autonomy in changing environments," in *ICRA Workshop on Visual Localization in Changing Environments*, 2015.
- [21] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *Robotics and Automation Letters*, 2018, in review.
- [22] Tomas Krajnik, Jaime Pulido Fentanes, Joao Santos, and Tom Duckett, "Frequency map enhancement: Introducing dynamics into static environment models," in *ICRA 2016 Workshop on AI for Long-term Autonomy*, 2016.
- [23] Tomáš Krajník, Jaime Pulido Fentanes, Marc Hanheide, and Tom Duckett, "Persistent localization and life-long mapping in changing environments using the frequency map enhancement," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4558–4563.
- [24] Tomáš Krajník, Jaime P Fentanes, Joao M Santos, and Tom Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, 2017.
- [25] Jaime Pulido Fentanes, Bruno Lacerda, Tomáš Krajník, Nick Hawes, Marc Hanheide, et al., "Now or later? predicting and maximising success of navigation actions from long-term experience," in *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, IEEE/RAS.
- [26] Ferdian Jovan, Jeremy Wyatt, Nick Hawes, and Tomáš Krajník, "A poisson-spectral model for modelling temporal patterns in human data observed by a robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.
- [27] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, L. Beyer, A. Hermans, B. Leibe, A. Aldoma, T. Faulhammer, M. Zillich, M. Vincze, E. Chinellato, M. Al-Omari, P. Duckworth, Y. Gatsoulis, D. C. Hogg, A. G. Cohn, C. Dondrup, J. Pulido Fentanes, T. Krajník, J. M. Santos, T. Duckett, and M. Hanheide, "The strands

project: Long-term autonomy in everyday environments," *IEEE Robotics Automation Magazine*, vol. 24, no. 3, pp. 146–156, Sept 2017.

- [28] S. Molina, G. Cielniak, and T. Krajník, T. Duckett, "Modelling and predicting rhythmic flow patterns in dynamic environments," in *UK-RAS*, 2017.
- [29] S. Molina, G. Cielniak, and T. Krajník, T. Duckett, "Modelling and predicting rhythmic flow patterns in dynamic environments," in *TAROS*, 2018, in review.
- [30] Joao Santos, Tomas Krajnik, Jaime Pulido Fentanes, and Tom Duckett, "Lifelong exploration of dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA) - Late-breaking result*, 2015.
- [31] Tomáš Krajník, Joao M Santos, and Tom Duckett, "Life-long spatio-temporal exploration of dynamic environments," in *European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–8.
- [32] Joao Santos, Tomas Krajnik, Jaime Pulido Fentanes, and Tom Duckett, "A 3d simulation environment with real dynamics: a tool for benchmarking mobile robot performance in long-term deployments," in *ICRA 2016 Workshop on AI for Long-term Autonomy*, 2016.
- [33] J. M. Santos, T. Krajník, J. P. Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4-d spatio-temporal maps," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 684–691, July 2016.
- [34] João Machado Santos, Tomáš Krajník, and Tom Duckett, "Spatio-temporal exploration strategies for long-term autonomy of mobile robots," *Robotics and Autonomous Systems*, vol. 88, pp. 116–126, 2017.
- [35] Miroslav Kulich, Tomáš Krajník, Libor Přeučil, and Tom Duckett, "To explore or to exploit? learning humans' behaviour to maximize interactions with them," in *International Workshop on Modelling and Simulation for Autonomous Systems*. Springer International Publishing, 2016, pp. 48–63.
- [36] Marc Hanheide, Denise Hebesberger, and Tomáš Krajník, "The when, where, and how: An adaptive robotic info-terminal for care home residents," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, New York, NY, USA, 2017, HRI '17, pp. 341–349, ACM.
- [37] T. Vintř, S. Molina, J.P. Fentanes, G. Cielniak, T. Duckett, and T. Krajník, "Spatio-temporal models for motion planning in human populated environments," 2018, in review.
- [38] T. Krajník, T. Vintř, S. Molina, J.P. Fentanes, G. Cielniak, and T. Duckett, "Warped hypertime representations for long-term autonomy of mobile robots," *Robotics and Automation Letters*, 2018, in review.
- [39] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil, "External localization system for mobile robotics," in *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.
- [40] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail, "A practical multirobot localization system," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3-4, pp. 539–562, 2014.

- [41] Matias Nitsche, Tomas Krajník, Petr Cizek, Marta Mejail, and Tom Duckett, "Whycon: an efficient, marker-based localization system," in *IROS Workshop on Aerial Open Source Robotics*, 2015.
- [42] Peter Lightbody, Tomáš Krajník, and Marc Hanheide, "A versatile high-performance visual fiducial marker detection system with scalable identity encoding," in *Proceedings of the Symposium on Applied Computing*. 2017, Association for Computing Machinery, Best paper award.
- [43] Farshad Arvin, Simon Watson, Ali Emre Turgut, Jose Espinosa, Tomáš Krajník, and Barry Lennox, "Perpetual robot swarm: Long-term autonomy of mobile robots using on-the-fly inductive charging," *Journal of Intelligent & Robotic Systems*, Oct 2017.
- [44] Peter Lightbody, Tomáš Krajník, and Marc Hanheide, "An efficient visual fiducial localisation system," *SIGAPP Appl. Comput. Rev.*, vol. 17, no. 3, pp. 28–37, Nov. 2017.

PUBLISHED WORKS USING SYSTEMS DEVELOPED IN THIS
THESIS

- [45] Tomáš Krajník, Jan Faigl, Vojtech Vonásek, H Szucsová, Ondrej Fišer, and Libor Preucil, "A visual navigation system for robotour competition," in *First International Conference on Robotics in Education, Bratislava*, 2010, vol. 1, pp. 95–100.
- [46] T. Krajník, J. Faigl, V. Vonásek, H. Szucsová, O. Fišer, and L. Přeučil, "A Monocular Navigation System for RoboTour Competition," *AT&P journal PLUS 2*, vol. 18, no. 1, pp. 57–63, 2010.
- [47] T. Krajník and J. Faigl, "Robot z čvut vítězem robotour 2008," *Pražská technika*, vol. 8, no. 5, pp. 6, 2008.
- [48] T. Krajník, V. Vonásek, and J. Faigl, "Robotour 2009 - soutěž outdoorových robotů," *ComputerWorld - ScienceWorld*, vol. 3, no. 4, únor 2010.
- [49] T. Krajník and V. Vonásek, "Robot z čvut vítězem robotour 2009," *Tecnicall*, vol. 3, no. 4, pp. 14–15, 2009.
- [50] Karel Košnar, Tomáš Krajník, Vojtech Vonásek, and Libor Preucil, "Lama-large maps framework," in *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial*, 2009, pp. 9–16.
- [51] P Levi, E Meister, AC Van Rossum, Tomas Krajnik, V Vonasek, P Stepan, W Liu, and Fabio Caparrelli, "A cognitive architecture for modular and self-reconfigurable robots," in *Systems Conference (SysCon), 2014 8th Annual IEEE*. IEEE, 2014, pp. 465–472.
- [52] Tomáš Krajník, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl, "Ar-drone as a platform for robotic research and education," in *International Conference on Research and Education in Robotics*. Springer Berlin Heidelberg, 2011, pp. 172–186.
- [53] Tomáš Krajník, Matías Nitsche, Sol Pedre, Libor Přeučil, and Marta E Mejail, "A simple visual navigation system for an uav," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*. IEEE, 2012, pp. 1–6.
- [54] Martin Saska, Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, and Libor Přeučil, "Low cost mav platform ar-drone in experimental verifications of methods for vision based autonomous navigation," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4808–4809.
- [55] Martin Saska, Tomas Krajnik, and Libor Preucil, "Cooperative μ uav-ugv autonomous indoor surveillance," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*. IEEE, 2012, pp. 1–6.
- [56] Martin Saska, Tomáš Krajník, Vojtěch Vonásek, Zdenk Kasl, Vojtěch Spurný, and Libor Přeučil, "Fault-tolerant formation driving mechanism designed for heterogeneous mavs-ugvs groups," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 603, 2014.
- [57] Martin Saska, Vojtěch Vonásek, Tomáš Krajník, and Libor Přeučil, "Coordination and navigation of heterogeneous mav-ugv formations localized by a 'hawk-eye'-like approach under a model predictive control scheme," *The International Journal of Robotics Research*, vol. 33, no. 10, pp. 1393–1412, 2014.

- [58] Martin Saska, Vojtěch Vonásek, Tomáš Krajník, and Libor Přeučil, "Coordination and navigation of heterogeneous uavs-ugvs teams localized by a hawk-eye approach," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2166–2171.
- [59] Martin Saska, Tomáš Krajník, Vojtěch Vonásek, Petr Vaněk, and Libor Přeučil, "Navigation, localization and stabilization of formations of unmanned aerial and ground vehicles," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE, 2013, pp. 831–840.
- [60] Keerthy Kusumam, Tomáš Krajník, Simon Pearson, Tom Duckett, and Grzegorz Cielniak, "3d-vision based detection, localization, and sizing of broccoli heads in the field," *Journal of Field Robotics*, vol. 34, no. 8, pp. 1505–1518, 2014.
- [61] Keerthy Kusumam, Tomas Krajnik, Simon Pearson, Grzegorz Cielniak, and Tom Duckett, "Can you pick a broccoli? 3d-vision based detection and localisation of broccoli heads in the field," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.
- [62] Jan Svab, Tomas Krajnik, Jan Faigl, and Libor Preucil, "FPGA based speeded up robust features," in *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*. IEEE, 2009, pp. 35–41.
- [63] Tomáš Krajník, Jan Šváb, Sol Pedre, Petr Čížek, and Libor Přeučil, "FPGA-based module for SURF extraction," *Machine vision and applications*, vol. 25, no. 3, pp. 787–800, 2014.
- [64] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska, "Low-cost embedded system for relative localization in robotic swarms," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 993–998.
- [65] Martin Saska, Tomas Baca, Justin Thomas, Jan Chudoba, Libor Preucil, Tomas Krajnik, Jan Faigl, Giuseppe Loianno, and Vijay Kumar, "System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization," *Autonomous Robots*, pp. 1–26, 2016.
- [66] Abdul Basit, Waqar S Qureshi, Matthew N Dailey, and Tomáš Krajník, "Joint localization of pursuit quadcopters and target using monocular cues," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 3-4, pp. 613, 2015.
- [67] G. Loianno, V. Spurny, T. Baca, J. Thomas, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, pp. 1–1, 2018.
- [68] P. Štěpán, T. Krajník, M. Petrlík, and M. Saska, "Vision techniques for on-board detection, following and mapping of moving targets," *Journal of Field Robotics*, 2018, In revision review after major corrections.
- [69] V. Spurny, T. Baca, M. Saska, R. Penicka, T. Krajnik, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Vision techniques for on-board detection, following and mapping of moving targets," *Journal of Field Robotics*, 2018, In revision review after major corrections.
- [70] Farshad Arvin, Ali Emre Turgut, Tomáš Krajník, and Shigang Yue, "Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm," *Adaptive Behavior*, vol. 24, no. 2, pp. 102–118, 2016.

- [71] Tomas Krajník, Farshad Arvin, Ali Emre Turgut, Shigang Yue, and Tom Duckett, "Cos ϕ : Vision-based artificial pheromone system for robotic swarms," in *IEEE International Conference on Robotics and Automation (ICRA) - Late-breaking result*. IEEE, 2015.
- [72] Farshad Arvin, Tomáš Krajník, Ali Emre Turgut, and Shigang Yue, "Cos ϕ : artificial pheromone system for robotic swarms research," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 407–412.

OTHER REFERENCES

- [73] David Austin, Luke Fletcher, and Alexander Zelinsky, "Mobile robotics in the long term-exploring the fourth dimension," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. IEEE, 2001, vol. 2, pp. 613–618.
- [74] G D Finlayson and S D Hordley, "Color constancy at a pixel," *Journal of the Optical Society of America: Optics, image science, and vision*, vol. 18, no. 2, pp. 253–64, Feb 2001.
- [75] Will Maddern, Alexander D Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," in *Proc. of Workshop on Visual Place Recognition in Changing Environments*, 2014.
- [76] C. McManus, W. Churchill, W. Maddern, A.D. Stewart, and P. Newman, "Shady dealings: Robust, long-term visual localisation using illumination invariance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 901–906.
- [77] K. MacTavish, M. Paton, and T. Barfoot, "Beyond a shadow of a doubt: Place recognition with colour-constant images," in *Proceedings of Field and Service Robotics (FSR)*, June 2015.
- [78] M. Paton, K. MacTavish, C.J. Ostafew, and T. Barfoot, "It's not easy seeing green: Lighting-resistant stereo visual teach-and-repeat using color-constant images," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.
- [79] Christoffer Valgren and Achim J. Lilienthal, "SIFT, SURF & seasons: Appearance-based long-term localization in outdoor environments," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 157–165, February 28 2010.
- [80] Feras Dayoub and Tom Duckett, "An adaptive appearance-based map for long-term topological localization of mobile robots," in *Proc. of Int. Conference on Intelligent Robots and Systems (IROS)*, 2008.
- [81] David M Rosen, Julian Mason, and John J Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, to appear.
- [82] Nicholas Carlevaris-Bianco and Ryan M Eustice, "Learning visual feature descriptors for dynamic lighting conditions," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [83] Stephanie Lowry, Gordon Wyeth, and Michael Milford, "Unsupervised online learning of condition-invariant images for place recognition," *Australasian Conference on Robotics and Automation 2014*, 2014.
- [84] Winston Churchill and Paul Newman, "Practice makes perfect? managing and leveraging visual experiences for lifelong navigation," in *ICRA*, 2012.
- [85] Colin McManus, Ben Upcroft, and Paul Newmann, "Scene signatures: localised and point-less features for localisation," in *Robotics: Science and Systems (RSS)*, 2014.

- [86] Peer Neubert and Peter Protzel, "Local region detector+ CNN based landmarks for practical place recognition in changing environments," in *ECMR*. IEEE, 2015, pp. 1–6.
- [87] Niko Sunderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford, "Place recognition with ConvNet landmarks: Viewpoint-robust, condition-robust, training-free," *Proceedings of Robotics: Science and Systems XII*, 2015.
- [88] Niko Sünderhauf, Feras Dayoub, Sareh Shirazi, Ben Upcroft, and Michael Milford, "On the performance of ConvNet features for place recognition," *arXiv preprint arXiv:1501.04158*, 2015.
- [89] Niko Sunderhauf and Peter Corke, "Visual place recognition in changing environments (VPRiCE)," 2015, <https://roboticvision.atlassian.net/wiki/pages/viewpage.action?pageId=14188617>.
- [90] Dmytro Mishkin, Michal Perdoch, and Jiri Matas, "Place recognition with WxBS retrieval," in *CVPR 2015 Workshop on Visual Place Recognition in Changing Environments*, 2015.
- [91] Stephanie Lowry, Michael Milford, and Gordon Wyeth, "Transforming morning to afternoon using linear regression techniques," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3950–3955.
- [92] Peer Neubert, Niko Sünderhauf, and Peter Protzel, "Appearance change prediction for long-term navigation across seasons," in *ECMR*, 2013.
- [93] Niko Sünderhauf, Peer Neubert, and Peter Protzel, "Predicting the change—a step towards life-long operation in everyday environments," *Robotics Challenges and Vision (RCV2013)*, 2014.
- [94] Gian Diego Tipaldi et al., "Lifelong localization in changing environments," *The International Journal of Robotics Research*, 2013.
- [95] Peter Biber and Tom Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proc. of Robotics: Science and Systems*, 2005, pp. 17–24.
- [96] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.
- [97] Tomasz Kucner, Jari Saarinen, Martin Magnusson, and Achim J Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.
- [98] Rares Ambrus, Nils Bore, John Folkesson, and Patric Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [99] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "Brief: Binary robust independent elementary features," in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 778–792. Springer, 2010.
- [100] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [101] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, Software available at <http://octomap.github.com>.

- [102] Diane J Cook, "Learning setting-generalized activity models for smart spaces," *IEEE Intelligent Systems*, , no. 99, pp. 1, 2010.
- [103] Tomasz Piotr Kucner, Martin Magnusson, Erik Schaffernicht, Victor Hernandez Bennetts, and Achim J Lilienthal, "Enabling flow awareness for mobile robots in partially observable environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1093–1100, 2017.
- [104] Harold Fredricksen and James Maiorana, "Necklaces of beads in k colors and k-ary de bruijn sequences," *Discrete Mathematics*, vol. 23, no. 3, pp. 207–210, 1978.
- [105] Mr. Spock, "Let that be your last battlefield," in *Star Trek*, 1969.



KEY ARTICLE [1] - JOURNAL OF FIELD ROBOTICS 2010

©[2017] Wiley Periodicals. This is an article published in Journal of Field Robotics: Krajník et al.: Simple Yet Stable Bearing-Only Navigation, 2010.

Simple Yet Stable Bearing-Only Navigation

Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přebil

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague,

121 35 Praha 2, Karlovo náměstí 13, Czech Republic

e-mail: tkrajnik@labe.felk.cvut.cz, xfaigl@labe.felk.cvut.cz, vonasek@labe.felk.cvut.cz, kosnar@labe.felk.cvut.cz, kulich@labe.felk.cvut.cz, preucil@labe.felk.cvut.cz

Received 8 January 2010; accepted 7 June 2010

This article describes a simple monocular navigation system for a mobile robot based on the map-and-replay technique. The presented method is robust and easy to implement and does not require sensor calibration or structured environment, and its computational complexity is independent of the environment size. The method can navigate a robot while sensing only one landmark at a time, making it more robust than other monocular approaches. The aforementioned properties of the method allow even low-cost robots to effectively act in large outdoor and indoor environments with natural landmarks only. The basic idea is to utilize a monocular vision to correct only the robot's heading, leaving distance measurements to the odometry. The heading correction itself can suppress the odometric error and prevent the overall position error from diverging. The influence of a map-based heading estimation and odometric errors on the overall position uncertainty is examined. A claim is stated that for closed polygonal trajectories, the position error of this type of navigation does not diverge. The claim is defended mathematically and experimentally. The method has been experimentally tested in a set of indoor and outdoor experiments, during which the average position errors have been lower than 0.3 m for paths more than 1 km long. © 2010 Wiley Periodicals, Inc.

1. INTRODUCTION

The fundamental problem of mobile robotics is to autonomously navigate a mobile robot along a given path. To fulfill this task efficiently, a robot should maintain some knowledge about its surrounding environment, especially its position relative to the path or desired destination. Such knowledge may be represented in the form of a map, which can be used to estimate the robot position as well as for motion planning. The map is either known a priori and the robot performs localization or is created online and the mobile robot performs so-called simultaneous localization and mapping (SLAM).

The solid mathematical background of the Kalman filter (Kalman, 1960) allowed the research community to establish a sufficient theoretical framework for extended Kalman filter (EKF)-based SLAM. Proof of EKF convergence (Dissanayake, Newman, Clark, Durrant-Whyte, & Csorba, 2001) and lower bounds (Gibbens, Dissanayake, & Durrant-Whyte, 2000) on robot position uncertainty have been formulated. Upper bounds are discussed in the paper by Mourikis and Roumeliotis (2004), where the authors emphasize the importance of robot heading precision during the mapping process. To our knowledge, there is no other paper concerning upper bounds of EKF position estimation. Unfortunately, optimality of the Kalman filter is proven only for linear systems and therefore the main

weakness of EKF methods lies in the linearization. The papers by Julier and Uhlmann (2001) and Martinelli, Tomatis, and Siegwart (2005) indicate that due to errors introduced in linearization, EKF methods might provide inconsistent results. Although the linearization process poses a significant threat to the consistency of the position estimation, it can be elegantly avoided using the inverse depth representation (Civera, Davison, & Montiel, 2008; Montiel, Civera, & Davison, 2006).

1.1. Vision-Based Navigation

The theoretical solutions of bearing-only SLAM have gained importance as the computational power of today's computers allows real-time image processing. The nature of visual information allows us to build sparse maps from well-distinguishable landmarks (Lowe, 1999, 2004), which are relatively easy to register. However, the range information is not provided directly by standard cameras. Some bearing-only methods use stereovision in order to obtain immediate range information (Kidono, Miura, & Shirai, 2000). Other methods substitute stereovision by motion and use a single monocular camera (Davison, Reid, Molton, & Stasse, 2007; Holmes, Klein, & Murray, 2008; Montiel et al., 2006).

Most monocular approaches are computationally complex and achieve low operational speeds when mapping large-scale environments. This problem can be solved by dividing a large global map into smaller maps with mutual position information (Bosse, Newman, Leonard, &

A multimedia file may be found in the online version of this article.

Teller, 2004; Clemente, Davison, Reid, Neira, & Tardós, 2007; Estrada, Neira, & Tardós, 2005; Williams, Cummins, Neira, Newman, Reid, et al., 2009).

A different approach is to build an environment map in advance and then use the map for localization (Blanc, Mezouar, & Martinet, 2005; Chen & Birchfield, 2009; Matsumoto, Inaba, & Inoue, 1996; Royer, Lhuillier, Dhome, & Lavest, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2007). In Royer et al. (2007), a monocular camera is carried through an environment and a video is recorded. The recorded video is then processed (in a matter of several minutes) and subsequently used to guide a mobile robot along the same trajectory. Chen and Birchfield (2006, 2009) present an even simpler form of navigation in a learned map. Their method utilizes a map consisting of salient image features remembered during a teleoperated drive. The map is divided into several conjoined segments, each associated with a set of visual features detected along it and a milestone image indicating the segment end. When a robot navigates a segment, its steering commands are calculated from positions of currently recognized and remembered features. The robot using this method moves forward with a constant speed and steers right or left with a constant velocity or does not steer at all. In Chen and Birchfield (2006), the segment end was detected by means of comparing the milestone image with the current view. The improved version (Chen & Birchfield, 2009) of the qualitative navigation uses a more sophisticated method to determine the segment end. The method takes into account the odometry, the current heading, and the similarity of the current and the milestone images. Still, the authors mention some problems with detection of the segment end. We claim that the segment end can be detected solely by the odometry and the comparison with the milestone image is not always necessary. Comparison with the milestone image increases robustness of the navigation method in cases of wheel slippage and other odometry errors.

The map-and-replay approach is closely related to visual servoing, in which the control of a robot is based on visual measurements (Chaumette & Hutchinson, 2006, 2007). Control inputs are either computed directly by a comparison of the current and reference images (Remazeilles & Chaumette, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2009) or by a computation of camera coordinates in the world reference frame (DeMenthon & Davis, 1992; Wilson, Hulls, & Bell, 1996). Normally, the visual servoing relies on a geometrical approach to calculate relations of map landmarks to the current image salient points (Segvic et al., 2009). These relations are used to calculate an interaction matrix (Chaumette & Hutchinson, 2006), which links observations to control inputs of the robot. The robot's control input can be computed from the Jacobian (Burschka & Hager, 2001), which relates world and image points, or from homography or fundamental matrices (Guerrero, Martinez-Cantin, & Sagüés, 2005; Remazeilles & Chaumette, 2007),

which relate coordinates between actual and reference images. A strong reliance on the geometrical representation requires either camera calibration (Burschka & Hager, 2001; Remazeilles & Chaumette, 2007; Segvic et al., 2009) or structured environment (Guerrero et al., 2005; Remazeilles & Chaumette, 2007). A more detailed overview of visual servoing approaches related to the field of mobile robotics is presented in Chen and Birchfield (2009) and Segvic et al. (2009). Contrary to the visual servoing approach, our method does not require a calibrated camera and does not rely on the environment structure.

1.2. Motivation

The target of our efforts is to create a system that would be able to reliably navigate a mobile robot in an unstructured environment of any size. To achieve this challenging goal, we have decided that the navigation system should have the following properties:

- Scalability: Its computational complexity should be independent of the environment size.
- Simplicity: The method should be as simple as possible, because complex systems are more likely to contain errors.
- Swiftiness: It has to satisfy real-time constraints.
- Standardness: It should use off-the-shelf equipment.
- Stability: The position uncertainty should not diverge with time.

The basic idea of the map-and-replay technique is similar to that of the industrial practice of programming stationary robots. One of the basic methods to program a stationary robot is by means of (tele)operation. A skilled operator guides the tip of the robot arm in order to perform a certain task (e.g., painting, welding). The robot records signals from its built-in receptors—typically incremental rotation sensors at its joints. During the robot operation, the recorded sequences serve as inputs for the robot's controllers. Though well established and efficient, this method is not applicable to mobile robots in unstructured environments due to the uncertainty in the robot-environment interaction. A typical example would be the use of odometry in a mobile robot localization—the uncertainty caused by wheel slippages tends to accumulate, which does not make odometry suitable for long-term localization and navigation. To effectively cope with the uncertainty in the robot's position, a mobile robot must use exteroceptors to sense the surrounding environment. A mobile robot position and its heading can be estimated through measurements of the surrounding environment.

Several authors of SLAM algorithms acknowledge the fact that the uncertainty of robot heading is a crucial factor affecting the quality of the map and subsequently the quality of position estimation in the localization step. The influence of the heading estimation has been evaluated both

theoretically and practically (Frese, 2006). We extend the idea of heading estimation importance and claim that for long-term mobile robot localization it is sufficient to use exteroceptive sensors for heading estimation and the Cartesian coordinate estimation can be based just on proprioceptive sensors.

1.3. Paper Overview

A minimalistic approach to monocular localization and mapping is presented in this paper. We claim that for the navigation in a known environment, a robot needs a map just to estimate its heading and can measure its position by odometry. Formulating this particular instance of the navigation mathematically, we provide a formal proof of this claim. Furthermore, several large outdoor experiments confirm the expected system performance. We think that the most important contribution of our paper is not the presented method but the convergence proof presented in Section 3. The proof would apply to several other methods (Chen & Birchfield, 2009; Guerrero et al., 2005; Zhang & Kleeman, 2009) that use vision to correct heading and lateral position errors.

The rest of this paper is organized as follows. The proposed minimalistic navigation method is described in the next section. A mathematical model of this navigation method is outlined and its properties are examined in Section 3. A theorem that claims that this method prevents position uncertainty divergence is formulated and proven in the same section. The theoretical analysis is followed by a discussion on the practical issues of the navigation method. Experimental results verifying whether the system retains the expected properties are described in Section 5. A conclusion briefly discusses the properties of the proposed navigation method and outlines possible future improvements.

2. NAVIGATION SYSTEM DESCRIPTION

The proposed navigation procedure is based on the map-and-replay technique. The idea is simple: a robot is manually driven through an environment and creates a map of its surrounding environment. After that, the map is used for autonomous navigation. A similar technique for autonomous navigation based on computation of a robot steering from positions of remembered features has been described in Chen and Birchfield (2006), Royer et al. (2007), and Zhang and Kleeman (2009). To minimize the robot sensor equipment and to satisfy the “standardness” property mentioned in Section 1.2, we consider the most available sensors. The fundamental navigation property of a mobile vehicle is the traveled distance, which can be estimated by odometry. The odometric error is cumulative and therefore can be considered precise only in the short term. Another standard available sensor that does not require additional infrastructure is a compass. A fusion of data from the com-

pass and odometry can provide position estimation but is still unsuitable for long-term navigation, because it lacks sufficient feedback from the robot’s surrounding environment. To increase robot ability to sense the environment, one of the most advantageous sensors is a camera, which can provide lots of information.

Using these three main sensors, we have proposed the following simple navigation strategy:

- The robot is navigated along a sequence of straight line segments.
- At each segment start, the robot is turned to a direction according to the compass value.
- The steering control along the straight segment is computed from matched visual features providing a so-called visual compass.
- The end of each segment is recognized according to the traveled distance, which is measured by odometry.

The crucial component of the proposed navigation procedure is a map, which is created by guiding the robot along a path consisting of straight-line segments. Each segment has its own landmark map L_i , consisting of salient features detected in images captured by the robot’s forward-looking camera, the initial robot orientation α and the segment length s . Once the map is created, the robot can travel autonomously within the mapped environment. During the navigation along a segment, the robot establishes correspondences of the currently seen and previously mapped landmarks and computes differences in the expected and recognized positions for each such correspondence. The robot steers in a direction that reduces those differences while moving straight at a constant speed until its odometry indicates that the current segment has been traversed. At the end of the segment, the robot switches to the next learned segment, turns to a direction of the initial orientation of the segment, and traverses the segment while keeping its direction according to matched features.

The next section describes the robot equipment and image processing. The algorithm for the map creation during the learning phase is described in Section 2.2, and the navigation algorithm is depicted in Section 2.3.

2.1. Robot Equipment

The proposed method has been verified on the P3AT robot with the Unibrain Fire-i601c camera, the TCM2 compass, and the HP 8710p laptop; see Figure 1(a). At first, the camera was equipped with a 7-mm objective with an electronically driven iris to prevent sunlight dazzle. The objective was replaced by a new one with a 4.5-mm focus length in 2008. At the same time, the electronically driven iris was substituted by a software exposure control. The laptop has Core2 Duo CPU running at 2.00 GHz and 1 GB of memory. Image processing is computationally demanding, and therefore the additional UPC70 battery had been used for longer experiments. To increase the robot action



(a) Robot configuration for experiments



(b) Image captured by robot camera and detected SURF positions



(c) Robot GUI with navigation phase data

Figure 1. Robot platform, detected features, and navigation graphical user interface (GUI).

radius, the three original batteries (connected in parallel) were replaced by one high-capacity battery and the robot's internal PC was disabled. The navigation system was implemented in C/C++ as a stand-alone Linux application.

The image processing algorithm is a critical component of the navigation system. The vision system must provide enough information to steer the robot in the right direction. Furthermore, it should be robust to real-world conditions, i.e., changing illumination, minor environment changes, and partial occlusions, and of course its performance should allow for a real-time response.

We have decided to use the speeded up robust features (SURF) (Bay, Tuytelaars, & Van Gool, 2006) method to identify landmarks in the image. The algorithm provides image coordinates of the salient features together with their descriptions. The SURF method is reported to perform better than most SIFT (Lowe, 1999) implementations in terms of speed and robustness to viewpoint and illumination changes. To achieve an additional speedup, the CPU implementation of the algorithm was adjusted to use both processor cores for parallel image processing. The captured image is horizontally divided, and the parts are processed in parallel. Later, we switched to the GPU (Cornelis & Van Gool, 2008) version of the algorithm. The GPU version has better real-time performance but is less distinctive than the CPU implementation (Svab, Krajcnik, Faigl, & Preucil, 2009). Nor-

mally, the recognition of a $1,024 \times 768$ grayscale image provides descriptors of 150–300 features and takes 100–500 ms. The outdoor environment is usually richer in detected features, and image processing tends to be slower than indoors. A typical outdoor processed image with highlighted feature positions is shown in Figure 1(b).

2.2. Learning Phase

In the learning phase, the robot is manually guided through an environment in a turn-move manner and creates a map consisting of several straight segments. Each segment is described by its length s , its azimuth α , and a set of detected landmarks L . A landmark $l \in L$ is described by the tuple $(\mathbf{e}, k, \mathbf{u}, \mathbf{v}, f, g)$, where \mathbf{e} is the SURF descriptor and k indicates the number of images in which the landmark was detected. Vectors \mathbf{u} and \mathbf{v} denote positions of the landmark in the captured image at the moment of its first and last detection, and f and g are the distances of the robot from the segment start in these moments.

The procedure that creates a map of one segment is shown in Algorithm 1. Before the robot starts to learn a segment, it reads compass data to establish the segment azimuth α and resets its odometric counters. After that, the robot starts to move forward, tracks detected features, and inserts them to the set L until the operator requests

Algorithm 1. Learn one segment

Input: α – an initial robot orientation (compass value)
Output: (α, s, L) – the data associated to the segment, where s is the traveled distance and L is a set of landmarks, a landmark is the tuple (k, e, u, v, f, g) , where e is the SURF descriptor, k is a counter of feature detection, u and v are positions of the features in the image (at the moment of their first, resp. last occurrence), f and g denote distance from the segment start according to u , resp. v .

```

 $L \leftarrow \emptyset$  // a set of learned landmarks
 $T \leftarrow \emptyset$  // a set of tracked landmarks
 $\alpha \leftarrow$  compass value // a robot orientation at the beginning of segment learning
repeat
   $d \leftarrow$  current distance from the segment start
   $S \leftarrow$  extracted features with associated image position,  $(u, e) \in S, u$  position,  $e$  feature descriptor
  foreach  $t_i = (e_i, k_i, u_i, v_i, f_i, g_i) \in T$  do
     $(u_a, e_a) \leftarrow \operatorname{argmin}\{\|e_i, e(s)\| \mid s \in S\}$  // select the best matching descriptor from  $S$  to  $e_i$ 
     $(u_b, e_b) \leftarrow \operatorname{argmin}\{\|e_i, e(s)\| \mid s \in S \setminus \{(u_a, e_a)\}\}$  // select the next best matching descriptor
    if  $\|(e_i, e_a)\| \ll \|(e_i, e_b)\|$  then
       $t_i \leftarrow (e_i, k_i + 1, u_i, u_a, f_i, d)$  // update matched landmark
       $S \leftarrow S \setminus \{(u_a, e_a)\}$  // remove matched feature from the current set of detected features
    else
       $T \leftarrow T \setminus \{t_i\}$  // remove  $t_i$  from the set of tracked landmarks
       $L \leftarrow L \cup \{t_i\}$  // add  $t_i$  to the set of learned landmarks
  foreach  $(u, e) \in S$  do
     $T \leftarrow T \cup \{(e, 1, u, u, d, d)\}$  // add new feature to the set of tracked landmarks
until operator terminates learning mode
 $s \leftarrow d$  // the total traveled distance along the segment
 $L \leftarrow L \cup T$  // add the current tracked landmarks to the set of learned landmarks

```

to stop. Images are continuously captured and processed during the movement. For each currently tracked landmark t_i (from the set T), two of the best matching features from the set of new features are found. If these two pairs are distinguishable enough (Bay et al., 2006), the best matching feature is associated to the tracked landmark, which is updated (values k, v, g). Each new feature is added to the set of tracked landmarks T , and its u and v are set to the value of the current distance from the segment start and the counter of the feature detection k is set to one. The segment description is saved at the end of the segment, and the operator can turn the robot to another direction and initiate mapping of a new segment. The format of the file, which stores the segment description, is shown in Table I.

2.3. Autonomous Navigation Mode

In the autonomous navigation mode, an operator enters a sequence of segments and indicates whether the robot should travel repeatedly. The robot is placed at the start of the first segment, loads the description of the segment, and turns itself to the segment azimuth and starts moving forward. The navigation procedure is shown in Algorithm 2. The relevant landmarks for the current robot position (i.e., according to the distance from the segment start) are selected from the set of the learned landmarks L . Correspondences between the mapped and the currently detected

landmarks are established in the same way as in the learning phase. A difference in horizontal image coordinates of the features is computed for each such couple. A modus of those differences is estimated by the histogram voting method. The modus is converted to a correction value of the movement direction, which is reported to the robot's steering controller. After the robot travels a distance greater than or equal to the length of the given segment, the next segment description is loaded and the procedure is repeated. During the navigation, the robot displays the relevant states (mapped and recognized landmarks, recognition success ratio, etc.) on its graphical interface; see Figure 1(c).

An important aspect of this navigation algorithm is the fact that it does not need to explicitly localize the robot or to create a three-dimensional map of detected landmarks. It should also be noted that the proposed method is able to work in real time. Even though the camera readings are utilized only to correct the robot direction and the distance is measured by the imprecise odometry, the position uncertainty does not accumulate if the robot changes direction often enough. The stability of the proposed navigation method is discussed in the next section.

3. STABILITY OF BEARING-ONLY NAVIGATION

First, we describe in an informal way how the robot position uncertainty is changed as the robot travels a closed

Table I. A part of a segment map in a text file.

Record	Value	Meaning
Initial azimuth and length	2.13, 7.03	α, s
Landmark 0		
First position	760.74, 163.29	\mathbf{u}_{l_0}
Last position	894.58, 54.44	\mathbf{v}_{l_0}
Max visibility	128	k_{l_0}
First and last visible distance	0.00, 4.25	f_{l_0}, g_{l_0}
Descriptor	1, 0.116727, -0.000254, 0.000499, 0.000352, ...	\mathbf{e}_{l_0}
Landmark 1		
First position	593.32, 381.17	\mathbf{u}_{l_1}
Last position	689.89, 377.23	\mathbf{v}_{l_1}
Max visibility	125	k_{l_1}
First and last visible distance	0.00, 6.73	f_{l_1}, g_{l_1}
Descriptor	-1, 0.070294, -0.006383, 0.012498, 0.006383, ...	\mathbf{e}_{l_1}

Algorithm 2. Traverse one segment

Input: (α, s, L) – the data associated to the segment, where α is an initial angle of the robot orientation at the segment start, s is the traveled distance and L is a set of landmarks, a landmark is the tuple (e, k, u, v, f, g) , where e is a SURF descriptor, k is a counter of feature detection, u and v are positions of feature in the image (at the moment of the first, resp. last, occurrence), f and g denote distances from the segment start according to u , resp. v .

Output: ω – a steering speed

```

turn( $\alpha$ ) // turn robot in the direction  $\alpha$ 
 $d \leftarrow$  current distance from the segment start
while  $d < s$  do
   $T \leftarrow \emptyset$  // a set of current tracked landmarks
   $H \leftarrow \emptyset$  // a set of differences (horizontal position in the image) of matched features
   $d \leftarrow$  current distance from the segment start
   $S \leftarrow$  extracted features with associated image position,  $(u, e) \in S, u$  position,  $e$  feature descriptor
  foreach  $l_i = (e_i, k_i, u_i, v_i, f_i, g_i) \in L$  do
    if  $f_i \geq d \geq g_i$  then
       $T \leftarrow T \cup \{l_i\}$  // add landmark to the tracked landmarks according to the traveled distance
  while  $|T| > 0$  do
     $(e_i, k_i, u_i, v_i, f_i, g_i) \leftarrow \text{argmax}_{t \in T} k(t)$  // get landmark with maximal number of occurrences  $k$ 
     $(u_a, e_a) \leftarrow \text{argmin}\{\|e_i, e(s)\| \mid s \in S\}$  // select the best matching descriptor from  $S$  to  $e_i$ 
     $(u_b, e_b) \leftarrow \text{argmin}\{\|e_i, e(s)\| \mid s \in S \setminus \{(u_a, e_a)\}\}$  // select the next best matching descriptor
    if  $\|(e_i, e_a)\| \ll \|(e_i, e_b)\|$  then
       $p \leftarrow (v_i - u_i)(d - f_i)/(g_i - f_i) + u_i - u_a$  // estimate angle to the matched landmark
       $H \leftarrow H \cup \{p_x\}$  // add horizontal difference to set of differences
     $T \leftarrow T \setminus \{(e_i, k_i, u_i, v_i, f_i, g_i)\}$  // discard used landmark
   $\omega \leftarrow \text{modus}(H)$  // determine new robot steering velocity
  report  $\omega$  to steering controller

```

path. This should help to interpret the mathematical formalism describing the robot position uncertainty in geometrical terms and make the rest of this section more comprehensible. After that, we lay down a formal description of the proposed navigation method and analyze its stability. We outline a model of the robot movement and depict

equations allowing the computation of the robot position uncertainty. Next, we use these equations to compute the robot position uncertainty for a closed path. Finally, we examine the properties of the proposed model and establish conditions ensuring that the robot position error does not diverge.

3.1. Geometrical Interpretation

Suppose that the learned path is a square and the robot has to travel it repeatedly. The robot is placed at a random [two-dimensional (2D) Gaussian distribution with zero mean] position near the first segment start; see Figure 2. The initial position uncertainty can therefore be displayed as a circle in which the robot is found with 90% probability. The navigation procedure is executed, and the robot starts to move along the first segment. Because it senses landmarks along the segment and corrects its heading, its lateral position deviation is decreased. However, owing to the odometric error, the longitudinal position error increases. At the end of the segment, the circle denoting position uncertainty becomes an ellipse, with a shorter axis perpendicular to the segment. Heading corrections are dependent on the value of the lateral deviation (see Section 3.2): the greater the deviation, the stronger the effect of heading corrections and therefore the lateral error decreases by a factor h for every traversed segment. The odometry error is independent of the current position deviation and is affected only by the length of the traversed segment, and therefore it can be modeled as an additive error o .

After the segment is traversed, the robot turns by 90 deg and starts to move along the next segment. The uncertainty changes again, but because of the direction change, the longer ellipse axis shrinks and the shorter is elongated due to the odometry error. This repeats for every traversed segment; the size of the uncertainty ellipse converges to a finite value. Because this particular trajectory is symmetric, axis lengths a , b of the “final” ellipse can be

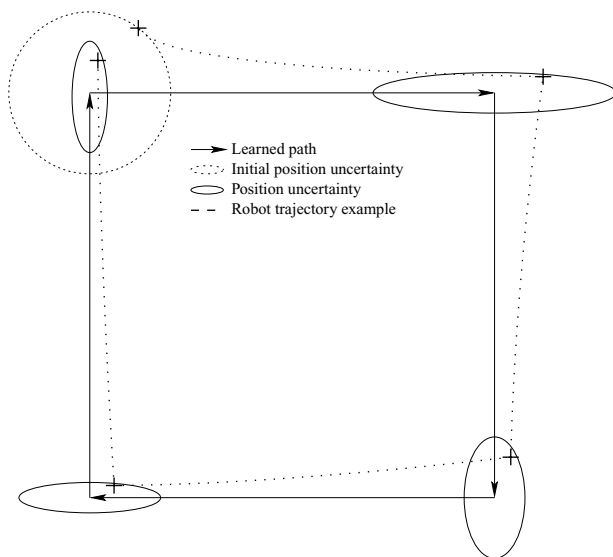


Figure 2. Position uncertainty evolution for a simple symmetric path.

easily computed by the equations

$$\begin{aligned} a &= hb, \\ b &= a + o, \end{aligned} \quad (1)$$

where h is the coefficient of the lateral error reduction and o is the odometric error. The position error for $o = 1$ and $h = 0.25$ is shown in Figure 2. Though simple, this particular symmetric case gives us a basic insight into the problem. Now we will derive a broader mathematical model of the navigation, examine its properties, and show that the uncertainty does not diverge for nonsymmetrical trajectories as well.

3.2. Navigation

The proposed navigation method is based on the following assumptions:

- The robot moves in a plane.
- The map already exists in the form of a sequence of conjoined linear segments with landmark description.
- At least two segments of the mapped path are not collinear.
- The robot can recognize and associate a nonempty subset of mapped landmarks and determine their bearing.
- The robot can (imprecisely) measure the traveled distance by odometry.
- The camera is aimed forward, i.e., in the direction of the robot movement.

The path P consists of a sequence of linear segments p_i . The robot moves in a plane, i.e., its state vector is (x, y, φ) . The robot we consider has a differential, nonholonomic drive, and therefore $\dot{x} = v \cos(\varphi)$ and $\dot{y} = v \sin(\varphi)$. For each segment p_i , there exists a nonempty subset of landmarks and a mapping between the robot position and the expected bearing of each landmark is established. At the start of each segment, the robot resets its odometry counter and turns approximately toward the segment end to sense at least one of the segment landmarks. The robot establishes correspondences of seen and mapped landmarks and computes differences in expected and recognized bearings. The robot steers in a direction that reduces these differences while moving forward until its odometry indicates that the current segment has been traversed.

Definition 1 (Closed-path stability property). Assume that a robot navigates a closed path several times. Furthermore the robot is using an environment map only for heading corrections and measuring the distance by odometry. Then a path for which the robot position uncertainty does not diverge has the closed-path stability property.

Theorem 1. A path consisting of several conjoined segments retains the closed-path stability property if the assumptions in Section 3.2 are satisfied.

3.3. Movement along One Segment

First, let us examine how a robot moves along one segment. We will focus on the position before and after traversing one segment and establish mapping from the robot position error at the segment start to the robot position error at the segment end.

To keep the model simple, we assume that the robot as well as the landmarks are positioned in a plane. We will consider having a map consisting of a single segment of length s with d landmarks, with positions represented as vectors \mathbf{u}_i . Because the robot is equipped with a forward-heading camera, learned landmark positions \mathbf{u}_i are not assumed to be distributed uniformly along the path but rather shifted in the direction of the robot movement by a distance ρ . We can assume that $\rho \approx \frac{1}{d} \sum_{i=0}^{d-1} u_{xi}$, where d is the number of mapped landmarks. Let us place the segment start at the coordinate origin and the segment end at the position $[s, 0]^T$. We designate the robot position prior to the segment traversal as $\mathbf{a} = [a_x, a_y]^T$ and the final robot position as $\mathbf{b} = [b_x, b_y]^T$; see Figure 3. Let us assume that at every moment during the segment traversal, the robot recognizes a nonempty subset \mathbf{W} of previously learned landmarks and the robot heads in a direction that minimizes the horizontal deviation of expected and recognized landmark positions. We denote the intersection of the robot heading with the learned segment axis as \mathbf{w} . At the beginning of the segment traversal, this position equals approximately $[\rho, 0]^T$ (i.e., $\mathbf{w} \approx [\rho, 0]^T$). As the robot traverses the segment, it loses sight of nearby landmarks and recognizes new ones. As new, more distant landmarks appear in the robot field of view and nearby landmarks disappear, the set \mathbf{W} changes and the point \mathbf{w} moves along the segment. It can be assumed that the point \mathbf{w} moves approximately at the speed of the robot and therefore it is always ahead of the robot by the distance ρ .

Based on these premises, the robot position $[x, y]^T$ in terms of $y = f(x)$ can be established. The robot movement can be characterized by the following differential equation:

$$\frac{dx}{dy} = \frac{\rho}{-y}. \tag{2}$$

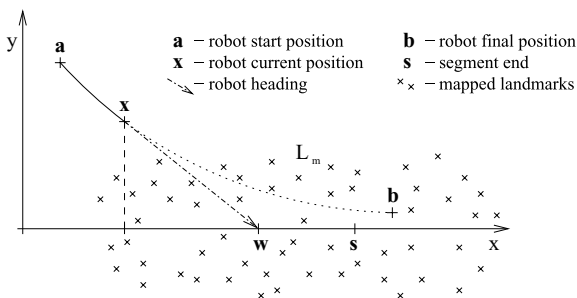


Figure 3. Robot movement model for a single path segment.

Solving Eq. (2) gives us a trajectory along which the robot moves:

$$y = ce^{-x/\rho}.$$

Considering a boundary condition $a_y = f(a_x)$, the constant c equals

$$c = \frac{a_y}{e^{-a_x/\rho}}.$$

Considering that the range of the robot's sensor is higher than the robot position uncertainty and that the segment length is higher than the robot lateral distance from the segment start (i.e., $\rho \gg a_x, s \gg |a_y|$), the constant c equals approximately a_y and the traveled distance is approximately equal to the segment length. Therefore, we can estimate the robot position after traveling a segment of length s by the following equations:

$$\begin{aligned} b_x &= a_x + s, \\ b_y &= a_y e^{-s/\rho}. \end{aligned} \tag{3}$$

We can transform Eqs. (3) to the matrix form

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s \\ 0 \end{pmatrix}. \tag{4}$$

Equation (4) is valid for an error-free odometry. If the odometry error is modeled as a multiplicative uncertainty, the equation changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + s \begin{pmatrix} 1 + v \\ 0 \end{pmatrix}, \tag{5}$$

where v is a random variable drawn from the Gaussian distribution with the zero mean and the variance ϵ . Accounting for the heading sensor noise, Eq. (5) changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s + s v \\ \xi \end{pmatrix}, \tag{6}$$

where ξ is a random variable of the Gaussian distribution with the zero mean and the variance τ . Consolidating Eq. (6), we can state

$$\mathbf{b} = \mathbf{M}\mathbf{a} + \mathbf{s}.$$

The aforementioned movement model holds for a segment aligned with the x axis. For a segment with an arbitrary orientation α , the movement model becomes

$$\mathbf{b} = \mathbf{R}^T \mathbf{M} \mathbf{R} \mathbf{a} + \mathbf{R}^T \mathbf{s}, \tag{7}$$

where

$$\mathbf{R} = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & m \end{pmatrix}.$$

Equation (7) corresponds to aligning the segment with the x axis, applying \mathbf{M} , adding the odometric and the sensor

noise, and rotating the segment back to the direction α . In the following text, we use $\mathbf{N} = \mathbf{R}^T \mathbf{M} \mathbf{R}$, which shortens Eq. (7) to

$$\mathbf{b} = \mathbf{N} \mathbf{a} + \mathbf{R}^T \mathbf{s}. \quad (8)$$

All the aforementioned assumptions about the surrounding environment (landmark shift equal to $\rho, \rho \gg a_x$, etc.) can be relaxed as long as $m < 1$ for $s > 0$.

3.4. Position Uncertainty

Now, the dependence of the robot position uncertainty at the segment end to its uncertainty at the segment start can be examined. Consider that the robot position \mathbf{a} before the segment traversal is a random variable drawn from a 2D normal distribution with the mean $\hat{\mathbf{a}}$ and the covariance matrix \mathbf{A} . To compute the robot position uncertainty after the segment traversal, we apply Eq. (8) to \mathbf{a} . Because the robot movement model in Eq. (8) has only linear and absolute terms, the robot position uncertainty after the segment traversal will constitute a normal distribution with the mean $\hat{\mathbf{b}}$ and the covariance matrix \mathbf{B} .

We denote $\mathbf{a} = \hat{\mathbf{a}} + \tilde{\mathbf{a}}$, where $\hat{\mathbf{a}}$ is the mean of \mathbf{a} and $\tilde{\mathbf{a}}$ is a random variable of a normal distribution with the zero mean and the covariance \mathbf{A} . Similarly, we can denote $\mathbf{b} = \hat{\mathbf{b}} + \tilde{\mathbf{b}}$. Thus, we can rewrite Eq. (7) as follows:

$$\tilde{\mathbf{b}} = \mathbf{R}^T \mathbf{M} \mathbf{R} \tilde{\mathbf{a}} + \mathbf{R}^T \tilde{\mathbf{s}}.$$

We can claim that

$$\tilde{\mathbf{b}} \tilde{\mathbf{b}}^T = (\mathbf{R}^T \mathbf{M} \mathbf{R} \tilde{\mathbf{a}} + \mathbf{R}^T \tilde{\mathbf{s}})(\mathbf{R}^T \mathbf{M} \mathbf{R} \tilde{\mathbf{a}} + \mathbf{R}^T \tilde{\mathbf{s}})^T.$$

Because $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{a}}$ are independent and do not correlate,

$$\tilde{\mathbf{b}} \tilde{\mathbf{b}}^T = \mathbf{R}^T \mathbf{M} \mathbf{R} \tilde{\mathbf{a}} \tilde{\mathbf{a}}^T \mathbf{R}^T \mathbf{M}^T \mathbf{R} + \mathbf{R}^T \tilde{\mathbf{s}} \tilde{\mathbf{s}}^T \mathbf{R},$$

which rewritten in terms of covariance matrices is

$$\mathbf{B} = \mathbf{R}^T \mathbf{M} \mathbf{R} \mathbf{A} \mathbf{R}^T \mathbf{M}^T \mathbf{R} + \mathbf{R}^T \mathbf{S} \mathbf{R}, \quad (9)$$

where

$$\mathbf{S} = \begin{pmatrix} s^2 \epsilon^2 & 0 \\ 0 & \tau^2 \end{pmatrix}.$$

Equation (9) allows us to compute the robot position uncertainty after traversing one segment.

3.5. Traversing Multiple Segments

Let us consider a path consisting of n chained segments denoted by $i \in \{0, \dots, n-1\}$ with the end of the last segment equal to the start of the first segment, i.e., the considered path is closed. We denote length and orientation of the i th segment as s_i and α_i . The robot position before and after traversing the i th segment is noted as \mathbf{a}_i and \mathbf{b}_i . Because the robot position at the end of the i th segment equals its start position at the segment $i+1$, we can state that $\mathbf{a}_{i+1} = \mathbf{b}_i$.

The movement model (9) for the i th traveled segment is

$$\mathbf{A}_{i+1} = \mathbf{B}_i = \mathbf{R}_i^T \mathbf{M}_i \mathbf{R}_i \mathbf{A}_i \mathbf{R}_i^T \mathbf{M}_i^T \mathbf{R}_i + \mathbf{R}_i^T \mathbf{S}_i \mathbf{R}_i. \quad (10)$$

Considering $\mathbf{N}_i = \mathbf{R}_i^T \mathbf{M}_i \mathbf{R}_i$ and defining $\mathbf{T}_i = \mathbf{R}_i^T \mathbf{S}_i \mathbf{R}_i$, we can rewrite Eq. (10) as

$$\mathbf{A}_{i+1} = \mathbf{N}_i \mathbf{A}_i \mathbf{N}_i^T + \mathbf{T}_i.$$

One can compute the robot position uncertainty in terms of the covariance matrix after traversing i path segments in the following terms:

$$\begin{aligned} \mathbf{A}_i = & \left(\prod_{j=i-1}^0 \mathbf{N}_j \right) \mathbf{A}_0 \left(\prod_{j=0}^{i-1} \mathbf{N}_j^T \right) \\ & + \sum_{j=0}^{i-1} \left[\left(\prod_{k=i-1}^j \mathbf{N}_k \right) \mathbf{N}_j^{-1} \mathbf{T}_j (\mathbf{N}_j^T)^{-1} \left(\prod_{k=j}^{i-1} \mathbf{N}_k^T \right) \right]. \end{aligned} \quad (11)$$

To examine how the robot position uncertainty changes after the robot travels the entire learned path i times, we define $\mathbf{C}_i = \mathbf{A}_{in}$ (e.g., $\mathbf{C}_1 = \mathbf{A}_n$). Moreover, we denote

$$\check{\mathbf{N}} = \prod_{j=n-1}^0 \mathbf{N}_j$$

and

$$\check{\mathbf{T}} = \sum_{j=0}^{n-1} \left[\left(\prod_{k=n-1}^j \mathbf{N}_k \right) \mathbf{N}_j^{-1} \mathbf{T}_j (\mathbf{N}_j^T)^{-1} \left(\prod_{k=j}^{n-1} \mathbf{N}_k^T \right) \right] \quad (12)$$

and rewrite Eq. (11) as

$$\mathbf{C}_{i+1} = \check{\mathbf{N}} \mathbf{C}_i \check{\mathbf{N}}^T + \check{\mathbf{T}}. \quad (13)$$

By proving that \mathbf{C}_i converges to a finite matrix as i grows to infinity, we prove Theorem 1.

3.6. Convergence Conditions

Expression (13) is the Lyapunov discrete equation (Lyapunov, 1992). If all eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle and $\check{\mathbf{T}}$ is symmetric, then $\lim_{i \rightarrow \infty} \mathbf{C}_i$ is finite and equal to \mathbf{C}_∞ , which can be obtained by solution of the algebraic equation

$$\mathbf{C}_\infty = \check{\mathbf{N}} \mathbf{C}_\infty \check{\mathbf{N}}^T + \check{\mathbf{T}}. \quad (14)$$

Because the matrix \mathbf{S}_i is symmetric, $\mathbf{T}_i = \mathbf{R}_i^T \mathbf{S}_i \mathbf{R}_i$ is also symmetric. The product $\mathbf{X} \mathbf{T}_i \mathbf{X}^T$ is symmetric for any \mathbf{X} and therefore all addends in Eq. (12) are symmetric. Addition

preserves symmetry and therefore the matrix

$$\check{\mathbf{T}} = \sum_{j=0}^{n-1} \left[\left(\prod_{k=n-1}^j \mathbf{N}_k \right) \mathbf{N}_j^{-1} \mathbf{T}_j \left(\mathbf{N}_j^T \right)^{-1} \left(\prod_{k=j}^{n-1} \mathbf{N}_k^T \right) \right]$$

is symmetric.

To prove that the eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle, we exploit the positiveness of the matrices \mathbf{M}_i and \mathbf{R}_i . Because \mathbf{M}_i is positive, $\mathbf{N}_i = \mathbf{R}_i^T \mathbf{M}_i \mathbf{R}_i$ is also positive. Moreover, as every \mathbf{N}_i equals $\mathbf{R}_i^T \mathbf{M}_i \mathbf{R}_i$, its eigenvalues are the same as those of \mathbf{M}_i and eigenvectors are columns of \mathbf{R}_i . The eigenvalues of \mathbf{N}_i therefore correspond to one and e^{-s_i/ρ_i} . Because the product $\mathbf{X}\mathbf{Y}$ of a positive definite matrix \mathbf{X} and a symmetric positive definite matrix \mathbf{Y} is positive definite, the matrix $\check{\mathbf{N}}$ is positive definite as well. Moreover, the dominant (maximal) eigenvalue of the product $\mathbf{X}\mathbf{Y}$ is lower than or equal to xy , where x and y are dominant eigenvalues of \mathbf{X} and \mathbf{Y} . Because the dominant eigenvalue of every \mathbf{N}_i is one, all eigenvalues of $\check{\mathbf{N}}$ are smaller than or equal to one. The dominant eigenvalue of $\check{\mathbf{N}}$ is equal to one if and only if the dominant eigenvalue of the product $\mathbf{N}_{i+1}\mathbf{N}_i$ equals 1 for all i . Conditions satisfying that the eigenvalues of a product $\mathbf{N}_{i+1}\mathbf{N}_i$ are lower than one ensure the existence of a finite solution of Eq. (14). Therefore, we have to find those conditions to support the closed-path stability property.

3.7. Convergence Proof

We will exploit the fact that a product of matrix eigenvalues equals the matrix determinant and the sum of eigenvalues equals matrix trace. Let us denote eigenvalues of the matrix product $\mathbf{N}_{i+1}\mathbf{N}_i$ as $\lambda_{0,1}$ and the smaller eigenvalue of \mathbf{N}_i as n_i ($n_i = e^{-s_i/\rho_i}$). For our convenience, we denote $j = i + 1$. Therefore

$$\det(\mathbf{N}_j\mathbf{N}_i) = \det \mathbf{N}_j \det \mathbf{N}_i = \lambda_0\lambda_1 = n_in_j. \quad (15)$$

If $\lambda_{0,1} \in (0, 1)$, we can state that

$$(1 - \lambda_0)(1 - \lambda_1) \geq 0, \quad (16)$$

and therefore

$$\lambda_0\lambda_1 - \lambda_0 - \lambda_1 + 1 \geq 0. \quad (17)$$

Combining Eq. (15) and inequality (17), we obtain

$$1 + n_in_j \geq \lambda_0 + \lambda_1.$$

Considering that the sum of eigenvalues equals matrix trace, we get

$$\text{trace}(\mathbf{R}_j^T \mathbf{M}_j \mathbf{R}_j \mathbf{R}_i^T \mathbf{M}_i \mathbf{R}_i) \leq 1 + n_in_j. \quad (18)$$

Because $\text{trace}(\mathbf{A}\mathbf{B})$ is equal to $\text{trace}(\mathbf{B}\mathbf{A})$, we can rewrite inequality (18) as

$$\text{trace}(\mathbf{M}_j(\mathbf{R}_i\mathbf{R}_j^T)^T \mathbf{M}_i \mathbf{R}_i \mathbf{R}_j^T) \leq 1 + n_in_j. \quad (19)$$

Both matrices \mathbf{R}_i and \mathbf{R}_j represent rotations. The matrix \mathbf{R}_i denotes rotation by the angle α_i , and \mathbf{R}_j denotes rotation by the angle α_j . Their product $\mathbf{R}_i\mathbf{R}_j^T$ denotes rotation by $\alpha_i - \alpha_j$. If we denote $\beta = \alpha_i - \alpha_j$ and $\mathbf{R}_\beta = \mathbf{R}_i\mathbf{R}_j^T$, inequality (19) is changed to

$$\text{trace}(\mathbf{M}_j\mathbf{R}_\beta^T \mathbf{M}_i \mathbf{R}_\beta) \leq 1 + n_in_j. \quad (20)$$

By expanding matrices $\mathbf{M}_j\mathbf{R}_\beta^T$, we obtain

$$\mathbf{M}_j\mathbf{R}_\beta^T = \begin{pmatrix} \cos \beta & -n_j \sin \beta \\ \sin \beta & n_j \cos \beta \end{pmatrix}$$

and

$$\mathbf{M}_i\mathbf{R}_\beta = \begin{pmatrix} \cos \beta & n_i \sin \beta \\ -\sin \beta & n_i \cos \beta \end{pmatrix}.$$

Inequality (20) can be rewritten to

$$(1 + n_in_j) \cos^2 \beta + (n_i + n_j) \sin^2 \beta \leq 1 + n_in_j$$

and further reduced to

$$1 + n_in_j - (1 - n_i - n_j + n_in_j) \sin^2 \beta \leq 1 + n_in_j.$$

Finally, we get

$$(1 - n_i)(1 - n_j) \sin^2 \beta \geq 0. \quad (21)$$

Because $n_i = e^{-s_i/\rho_i}$, $n_i \in (0, 1)$ and $n_j \in (0, 1)$, and inequality (21) is strict for $\sin \beta \neq 0$. This fact implies that inequality (16) is strict as well, which means that both λ_0 and λ_1 are lower than one. Therefore both eigenvalues of the matrix product $(\mathbf{N}_i\mathbf{N}_j)$ are smaller than one if $\beta \neq n\pi |n \in \mathcal{N}$. The matrix $\check{\mathbf{N}}$ has both eigenvalues smaller than one if and only if at least two conjoined segments of the path form an angle different from 0 or π .

Because all eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle and $\check{\mathbf{T}}$ is symmetric, the covariance matrix \mathbf{C}_∞ denoting the robot position uncertainty at the start of the first path segment is finite and obtainable by a solution of the algebraic equation

$$\mathbf{C}_\infty = \check{\mathbf{N}}\mathbf{C}_\infty\check{\mathbf{N}}^T + \check{\mathbf{T}}.$$

□

3.8. Convergence Proof Overview

We have established Eqs. (7) describing the movement of a robot using a navigation method, which is described in Section 2. Equation (10) allowed us to examine the robot position uncertainty evolution as the robot travels through a known environment. Modifying Eq. (10) to closed trajectories, we could rewrite it as Eq. (13). By examining the conditions under which Eq. (13) has a finite solution, we have proven Theorem 1 for closed paths that have at least two noncollinear segments. The existence of a finite solution of Eq. (13) means that if a mobile robot traverses repeatedly a

closed polygonal path using our method, its position error at every point of the traversed trajectory will stabilize at a certain value.

4. PRACTICAL ISSUES

The theoretical proof of convergence stands on several assumptions, which might not always be met. In this section, we will outline possible issues that might arise from incorrect assumptions and discuss their impact on navigation stability and accuracy. Moreover, we will discuss method requirements in terms of computational power, memory, and disk storage.

4.1. Convergence Proof from an Engineer's Point of View

Though elegant and useful, mathematical models and formal proofs are often based on a simplification of reality. A good mathematical model picks out the essence of the modeled problem and concentrates on an examination of the model properties. Some properties of the real system are not included in the model and therefore are not considered. An experienced engineer must be aware of these properties and realize the difference between math and reality. This applies to the proof presented in Section 3 as well.

In practice, extremely elongated (imagine a rectangle with sides 1,000 and 0.001 m long) paths will not retain the closed-path stability property because the movement along the shorter side will not compensate odometry errors accumulated over the long side. Moreover, the model does not cover the probability that the robot will not establish enough correct correspondences because its position error would grow too high. This might easily happen in case the robot has to avoid large obstacles as well as for paths with very long segments.

Moreover, the fact that the position error does not diverge might not be really useful in real robotic systems. In practice, the real precision is more important. The precision can be estimated using Eq. (13) if the learned path shape, landmark distribution ρ , camera noise τ , and odometry noise ϵ are known. Using $\rho = 20$ (i.e., most of the sensed landmarks are 20 m in front of the robot), the sensor noise $\tau = 0.1$, and the odometry noise $\epsilon = 0.0005$ for a square, 1-km-long path, the predicted navigation repeatability (i.e., computed from eigenvalues of \mathbf{C}_∞) is 0.15 m. This value is in good accordance with the experimental results presented in Section 5, where the measured repeatability in outdoor scenarios was 0.14 m.

4.2. Reliance on Odometry

Odometry is regarded as unsuitable for long-term localization due to cumulative errors. Its error model is usually multiplicative with a precision around 1%. The error of the odometric pose estimation is caused mainly by the

fact that the robot heading cannot be properly determined. On the other side, odometry can be very precise for traveled distance measurements. Moreover, an odometric error is usually systematic, which can be solved by precise calibration. Our experience with the P3AT robot shows that repeatability of its odometric measurements of the traveled distance on paved roads is better than 0.1%. This means that in the case of precise heading estimation, the robot would be able to travel 1 km with a position error lower than 1 m.

Our approach relies on the fact that the robot changes direction often enough. If the robot would travel in a straight direction for a long distance, its position error might grow beyond an acceptable level. This might be avoided either by forcing the robot to change directions during the learning phase or by complementing the distance measurement by methods without a long-term drift. An example of such a method might be global positioning system or a vision-based localization used in methods in Chen and Birchfield (2009), Royer et al. (2007), and Zhang and Kleeman (2009).

4.3. False Correspondences

The most troublesome issue is that correct correspondences might not be established. However, our algorithm works even in cases of a large number of outliers. Consider a situation in which the system is navigating and all of its established correspondences are false. The horizontal position deviation of detected and mapped features would be basically a random variable. Therefore a histogram H , which is built in order to establish the robot turning speed, will have its bins (approximately) equally filled. The robot turning speed will therefore be random. Now consider that there are a few correctly established correspondences. Each correctly established correspondence increases the value of the bin, which corresponds to the robot's true heading deviation. Therefore the probability that the correct bin has a maximal value increases with each correct correspondence.

This is different from the work presented in Chen and Birchfield (2009) and Segvic et al. (2007), where the authors choose a mean of horizontal differences instead of the modus. The modus is more invariant to the incorrect correspondences than the mean, which makes our method more precise and robust.

In reality, we get 80%–90% correctly established correspondences if the navigation phase follows mapping immediately. As the map gets older, the ratio of correct correspondences tends to drop. The rate of “map decay” depends on the environment and is caused mainly by two factors: short-term lighting changes caused by a change in the position of the sun and the current weather conditions and long-term environment changes caused by seasonal factors. Both illumination and long-term changes are not so significant in indoor environments, because lighting is typically artificial and seasonal changes do not happen. So it

is expected that illuminations and seasonal changes would play an important role in outdoor environments.

To evaluate the system robustness to lighting changes, we made an all-day experiment in which the robot traversed a 1-km-long path in an outdoor environment; see Section 5.5. To evaluate our system robustness to seasonal environment changes, we mapped a 50-m-long path in a park. The path was autonomously navigated and then re-learned one month later. This was done in five consecutive months; see Section 5.4. The results of both experiments show that the system is robust to both long-term and short-term environment changes.

Dynamic objects and occlusions cause only a temporary and slight decrease in the ratio of correctly established correspondences. During the experiments, we did not notice any problems with moving objects in the robot field of view.

4.4. Obstacle Avoidance

The proposed navigation method itself does not include obstacle avoidance. However, it can be complemented by a collision avoidance module, which takes control of the robot whenever an obstacle in the robot's course is detected. Such a module guides the robot around the obstacle until the area between the robot and its path is clear again. After that, the visual-based navigation takes control and guides the robot by Algorithm 2.

Because obstacles have finite dimensions, the robot position error will grow by a finite value every time it passes an obstacle. From the theoretical point of view, random obstacles in the robot path can be modeled by the addition of a random vector with a zero mean to \mathbf{s} in Eq. (8). Although the addition will increase the matrix \mathbf{S} in Eq. (9), the symmetry of \mathbf{S} will be preserved. Obstacles would therefore increase the matrix $\tilde{\mathbf{T}}$ in Eqs. (13) and (14), but because $\tilde{\mathbf{T}}$ remains symmetric, Eq. (14) will have a unique solution. However, the robot position uncertainty, represented by the matrix \mathbf{C}_∞ , will increase. Therefore, obstacle avoidance would decrease the precision of the robot navigation, but it should remain stable. This assumption is experimentally verified in Section 5.3.

It is clear that there exists a size of obstacles for which the algorithm will fail, because after circumnavigating the obstacle, the robot will not find previously mapped features.

4.5. Systematic Errors

Because the navigation algorithm relies on two sensors, there are two sources of systematic errors in our algorithm: the odometry and the camera.

The systematic error of the odometry means that if the robot traverses the distance d , it will report that the traveled distance is $d(1 + \eta)$. Let us consider that the robot has 900% odometric error, i.e., $\eta = 9$. During mapping phases,

the error will cause the segment lengths s and landmark data f and g to be 10 times higher in the map than in reality. However, in the navigation phases, the odometry error will give 10 times higher values of the robot distance from the segment start, and therefore errors in the map and robot position error will suppress each other.

The systematic error of the camera might be caused by the misalignment of the camera optical axis and the robot body. This causes the positions of landmarks \mathbf{u} , \mathbf{v} in the map to be different from a case with an ideal camera. However, when the robot encounters the same location, detected landmark positions will be shifted the same way as in the learning phase. The systematic error will therefore cancel out as in the previous case.

A different case would be a change of the odometric error η or the camera angle θ between the learning and navigation phases. From a theoretical point of view, this would cause a change of the vector \mathbf{s} . Unlike in the previous case, the vector \mathbf{s} will not be modified by a random vector but a fixed one. This means that $\tilde{\mathbf{s}}$ will remain the same and matrix $\tilde{\mathbf{T}}$ will preserve symmetry. Systematic errors would cause the robot to traverse a trajectory slightly different from the learned one but should not affect navigation stability. However, the algorithm will fail to establish correct correspondences between the mapped and detected features if the systematic errors are too high.

The experimental evaluation of the influence of systematic errors on the navigation stability is described in Section 5.2. Even though the systematic errors were set to high values during the experimental evaluation, the navigation stability was preserved.

4.6. Necessity of a Compass

Relying on only a compass for heading estimation was shown to be a weak point during experiments performed in 2008 and 2009. During learning phases, the compass noise caused an incorrect azimuth estimation of some segments.

Therefore, we considered replacing the absolute azimuth measurements by relative ones. So, instead of recording α_i for the i th segment in the learning phase, the azimuth relative to the previous segment (i.e., $\Delta_i = \alpha_i - \alpha_{i-1}$) is recorded in the map. The relative azimuth Δ_i can be estimated either by odometry or by tracking of the features during transitions to the next segment. This approach is applicable in cases in which a robot is taught a path that is supposed to be traversed later. However, sometimes it is necessary to create a more complex, graph-like map; see Section 5.7.

In more complex cases, the robot creates a map of the large environment in several mapping runs and traverses the given sequence of segments in an order different from the mapped sequence. In this case, sole knowledge of the relative segment azimuths is not sufficient, because angles between nonconsecutive segments are not known to us. Of course an angle between the i th and $(i + j)$ th segments can

be estimated by summing all relative angles between segments i and $i + j$, but because every Δ_i contains a small error, the error of the sum is too large for high j .

To deal with these more complex cases, we have implemented a simple Kalman filter, which fuses data from odometry and compass. The filter suppresses the compass noise and causes the absolute heading measurements to be more reliable. However, the filter was implemented at the end of 2009, so in the previous experiments the compass noise caused trouble.

4.7. Computational and Storage Requirements

To estimate computational and storage requirements, we have used data from the experiment described in Section 5.5.

We evaluated the computational requirements in terms of required computational time spent in various stages of the algorithm. The most computationally intensive stage of the algorithm is the feature extraction, which takes 260 ms on average. About 30 ms is taken by establishing proper correspondences. The histogram voting time is less than 1 ms. During experiments, the camera image and additional parameters of the algorithm were displayed for debugging purposes. Drawing these data on a computer screen takes about 60 ms. Thus, the entire control loop takes about 350 ms.

The average landmark density is about 140 landmarks per meter of the path. The map is stored on the hard drive in a text format (see Table I), and one landmark occupies about 800 bytes. Therefore, the disk storage needed for 1 km of path is about 112 MB. Once loaded to the computer memory, a landmark is represented in binary and occupies less than 300 bytes. Thus, a segment 1 km long would take 42 MB of computer memory.

5. EXPERIMENTS

The assumptions formed in Sections 3 and 4 were verified in several real-world experiments. The experimental evaluation was performed in seven different scenarios examining the following:

1. Convergence for two types of paths—with and without the closed-path stability property
2. The impact of systematic errors to the navigation precision
3. Feasibility of complementing the method by the collision avoidance module
4. Robustness to environment changes and variable lighting conditions
5. Performance in environments with landmark deficiency
6. Navigation efficiency for long paths in an outdoor environment with diverse terrain
7. Real deployment of the navigation procedure in RoboTour 2008 and RoboTour 2009 contests (Iša & Dlouhý, 2010)

During these scenarios, the robot autonomously traversed more than 3 km of indoor and more than 25 km of outdoor paths. The P3AT platform with the configuration described in Section 2 was used in all testing scenarios.

The robot learned different closed paths and was requested to navigate these paths several times in the first six scenarios. The relative position \mathbf{c}_i of the robot to the path start was measured every time the robot completed the i th path loop. To evaluate the quality of the navigation algorithm, accuracy and repeatability values as in Chen and Birchfield (2009) were used. The accuracy ε_{acc} and the repeatability ε_{rep} are computed as the rms of the Euclidean distance or the standard deviation of the robot's final positions from the path start:

$$\varepsilon_{\text{acc}} = \sqrt{\frac{1}{n-j} \sum_{i=j}^n \|\mathbf{c}_i\|^2}, \quad \varepsilon_{\text{rep}} = \sqrt{\frac{1}{n-j} \sum_{i=j}^n \|\mathbf{c}_i - \mu\|^2}, \quad (22)$$

where \mathbf{c}_i is the robot position relative to the path start after completing the i th loop and $\mu = \sum_{i=j}^n \mathbf{c}_i / (n-j)$. In most scenarios, the initial robot position was intentionally changed to be 1.5 m apart from the learned path start. In these cases, we do not set j to 1 but wait five loops until the initial position error diminishes. Thus, the repeatability and the accuracy are computed for $j = 5, n = 20$ in the first four scenarios.

5.1. Stability of Robot Position for Different Types of Paths

The scenario examines Theorem 1 for paths with and without the closed-path stability property. The conclusions made in Section 3 indicate that paths with all collinear segments do not retain the closed-path stability property, whereas other paths do. The following paths have been considered: a path with only two collinear segments (i.e., a “back and forth” line path) and a square path. At first, the robot was taught these closed paths in an indoor hall. After that, the robot was placed either directly at the path start or 1.5 m away and requested to navigate along the learned path 20 times. The robot position \mathbf{c}_i was measured after each completed loop.

The first (degenerate) path was formed of two collinear, 5-m-long, segments. The square path was composed of four segments, each 5 m long, with the end of the last segment identical to the segment at the start of the path. The distance of the robot from the path start after each loop (i.e., $\|\mathbf{c}_i\|$) is shown in Figure 4.

The values indicate that for square trajectories, the robot was able to correct the position error that was introduced at the beginning of navigation along the learned path. Only was the error in the y coordinate (i.e., the coordinate axis normal to path segments) partially corrected for collinear trajectories, while the x coordinate remained

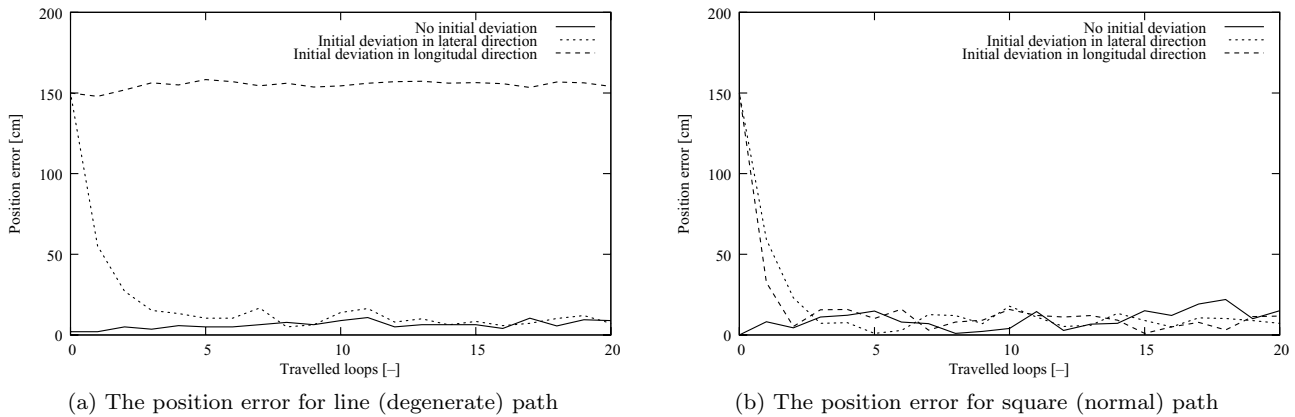


Figure 4. The position error for paths without and with the stability property.

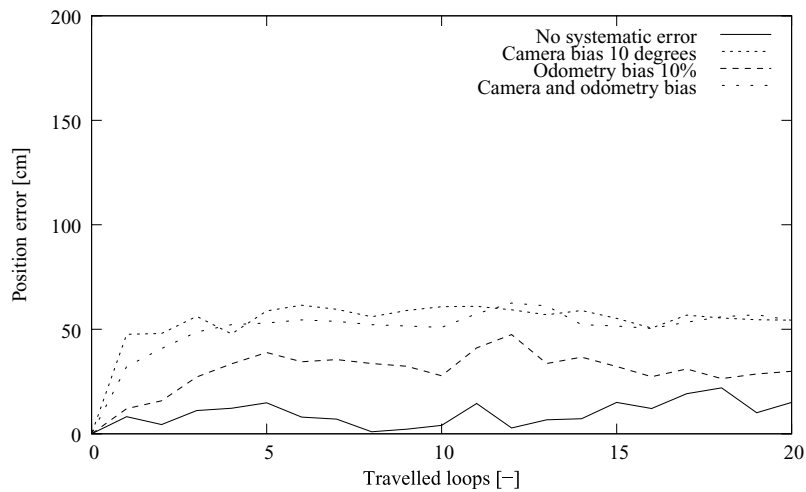


Figure 5. Systematic errors effect: the position error for different types of systematic errors.

uncorrected. These experimental results confirm the theoretical assumptions described in Section 3.8, stating that the navigation is unstable for paths with only collinear segments.

The robot traversed more than 1.8 km in this scenario. Both the accuracy and the repeatability for the 20-m-long square paths were 0.10 m.

5.2. Effect of Systematic Errors

The effect of the systematic errors on navigation precision was evaluated in this scenario. Two sources of systematic errors were considered: the camera and the odometry. An error of the camera can be caused by its optical axis deviation, and an odometric error can be caused by a tire pressure change. To show the effect of the parameter change, it is necessary to modify these parameters between the learning and the navigation phases; otherwise a path is learned

with the systematic errors, and therefore the errors do not have an effect.

The following experiments were performed to verify that small-scale systematic errors do not affect navigation stability. At first, the robot camera was panned by 10 deg, and the robot was requested to traverse the square path learned during scenario 5.1 20 times. Then, a 10% systematic odometry error was introduced.¹ Finally, the robot was requested to traverse the path 20 times with both 10% odometry and 10-deg camera bias. As in the previous cases, the robot position c_i was measured each time the robot reached the learned path start. The measured distances from the path start are shown in Figure 5.

¹This was done in software—a distance of the robot from the segment start measured by odometry was multiplied by a factor of 1.1 before it was passed to the navigation algorithm.

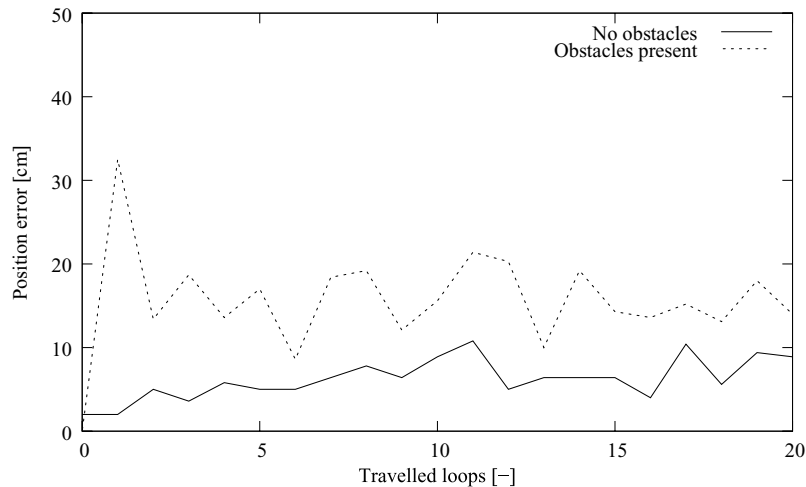


Figure 6. Obstacle avoidance experiment: the position errors with and without obstacles.

The results show that the odometric and the camera biases cause errors in robot positioning but the error does not diverge. After a few loops, the position error does not grow anymore and the system reaches a steady state. The overall accuracy is lower than without the systematic errors, but repeatability remains similar to the previous scenarios.

The robot traversed more than 1.2 km in this scenario. The average accuracy with the camera bias was 0.58 m, and the odometry bias caused the accuracy to change to 0.34 m. When both the odometry and the camera were biased, the accuracy was 0.55 m. The average repeatability was lower than in the previous scenario, i.e., 0.06 m.

5.3. Obstacle Avoidance

A simple collision avoidance module (based on the robot sonars) was activated in this scenario. The collision avoidance is based on the Tangent Bug algorithm with a finite range sensor (Choset, Lynch, Hutchinson, Kantor, Burgard, et al., 2005). When the robot detects an obstacle on its course, the visual-based navigation is suppressed and the robot starts to circumnavigate the detected obstacle. During the circumnavigation, odometry is used to determine the robot position and the sonar data are used to estimate the obstacle position. The visual navigation algorithm is resumed when the path between the robot and the end of the current segment is clear.

The robot was taught a square path similar to the one used in scenario 5.1. After that, one obstacle² was placed on each path segment, and the robot navigated the path 20 times. The robot autonomously navigated approximately 0.4 km with an accuracy of 0.16 m and a repeatability

of 0.08 m. It is clear that the position precision was affected but did not diverge. See Figure 6.

5.4. Environment and Lighting Changes

The effects of variable lighting conditions and long-term environment changes were examined in this scenario. At first, the robot was taught a closed, 50-m-long path consisting of five segments in the Stromovka park located in the city of Prague. One month later, the robot was placed 1.5 m away from the path start and was requested to navigate the path 20 times. This procedure was repeated five times, i.e., the test was done every month from November 2009 until April 2010. In each experiment, the robot used a map created in a previous month. The measured distances are shown in Figure 7.

Not only did the lighting conditions differ every time but also the environment went through seasonal changes. To document these changes, a picture from the onboard camera was stored every time the mapping was initiated; see Figure 8. There were considerably fewer correct correspondences between recognized and learned features. With a map created just before the navigation, the robot usually correctly recognizes 70%–90% of learned landmarks. Using a 1-month-old map, the ratio of the correctly recognized landmarks drops to 10%–40%. Nevertheless, the robot was able to correct its initial position error and was able to traverse the path faultlessly in all cases.

The robot autonomously navigated more than 6 km with an average accuracy of 0.24 m in this scenario. Unlike in previous scenarios, we did not measure the robot position \mathbf{c}_i after completion of each loop; we recorded the robot distance only from the path start, i.e., $\|\mathbf{c}_i\|$. Therefore, the repeatability cannot be calculated by Eqs. (22). Except for winter months, pedestrians regularly crossed the robot path and moved into the robot's field of view.

²Obstacle dimensions were approximately half of the robot size.

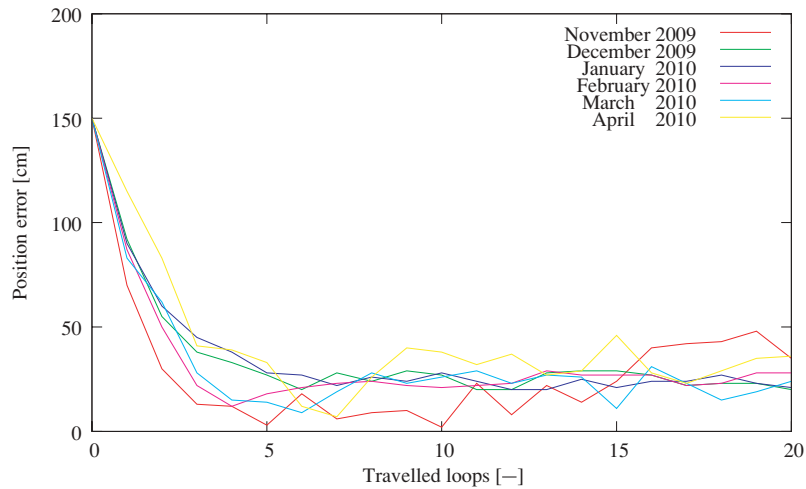


Figure 7. The position errors in different months of the long-term experiment.

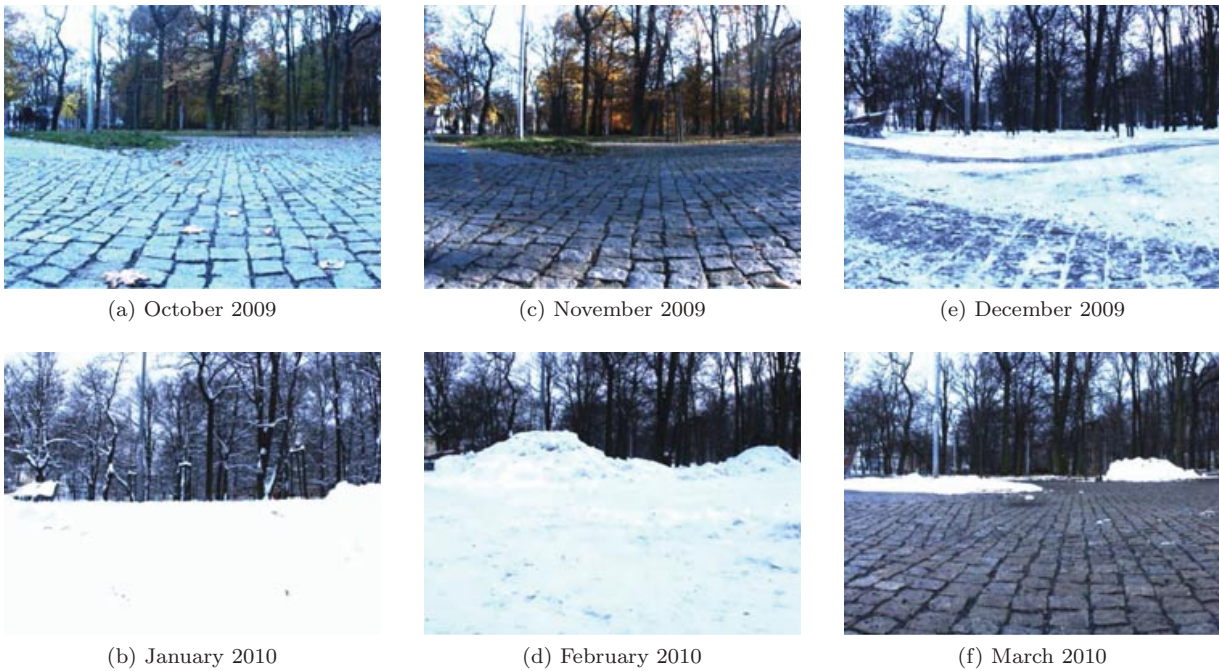


Figure 8. Long-term experiment: the view from the robot's camera at the path start in different months.

5.5. One-Day Outdoor Experiment

The system performance for long paths was evaluated in a realistic outdoor environment. The experiment was performed around the Proboštov pond³ in Proboštov, Czech Republic, at the end of March 2010. The robot was taught a 1-km-long path around the pond in the morning. The path went through a variable nonflat terrain with asphalt paths, dirt roads, footpaths, and grass terrain in an approximately equal ratio (see Figure 9). After the path was taught, the

robot was placed 1.5 m away from the path start and requested to traverse it repeatedly. Every time it reached the path start, its position was measured and its batteries replaced (the robot was not moved during the battery exchange). It took approximately 1 h for the robot to traverse the learned path, and the battery replacement took 15 min. The weather changed from cloudy/light rain to partly cloudy/sunny during the experiment. In the afternoon, a lot of pedestrians showed up and either entered the robot's field of view or crossed its path.

Nevertheless, the robot was able to complete the learned path six times before nightfall. The robot traversed

³50°39'58.716"N, 13°50'18.35"E.



Figure 9. One-day experiment: the path around Proboštov pond and the dirt road terrain example.

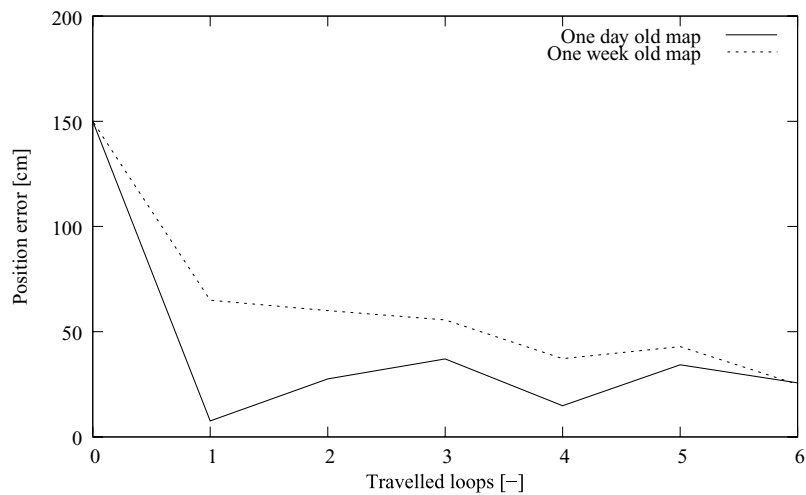


Figure 10. The position errors of the one-day experiment.

6 km with an accuracy⁴ of 0.26 m and a repeatability of 0.14 m. The experiment was repeated (without the learning phase) 1 week later, and the robot traversed the path six times with an accuracy of 0.31 m and a repeatability of 0.20 m. The measured distances are shown in Figure 10.

5.6. Landmark Deficiency Experiment

We have claimed that the system is able to operate in an environment that contains a low number of landmarks. To verify this assumption, we taught the robot an outdoor path during night and let it navigate using only street-lamp lights. The robot was taught a 0.3-km-long path on paved roads in a residential area. The onboard camera iris was fully opened, and the camera exposure time was set to 0.37 s. The path was taught at midnight, so more than

90% of the mapped landmarks were streetlamps and illuminated windows.

After the path was learned, the robot was placed 1.5 m from the path start and requested to traverse it 10 times. As opposed to in the daytime experiments, in which the robot detected typically 150–300 landmarks, during the nighttime, the typical number of landmarks was 3. The robot traversed 3 km with an accuracy⁵ of 0.32 m and a repeatability of 0.16 m. See Figure 11.

5.7. The RoboTour Outdoor Delivery Challenge

The RoboTour contest (Dlouhy & Winkler, 2009; Iša & Dlouhý, 2010) is an international autonomous robot delivery challenge organized by robotika.cz. The participating

⁴In this case, ϵ_{acc} and ϵ_{rep} were computed with $j = 3$ and $n = 6$ in Eqs. (22).

⁵In this case, ϵ_{acc} and ϵ_{rep} were computed with $j = 3$ and $n = 10$ in Eqs. (22).

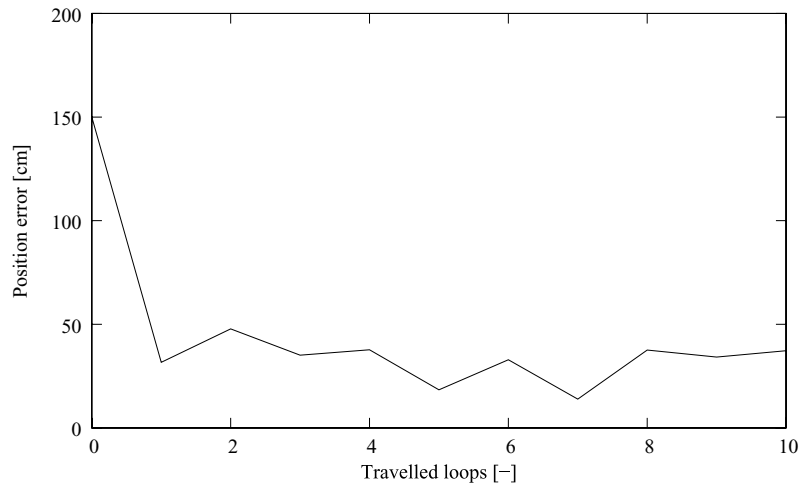
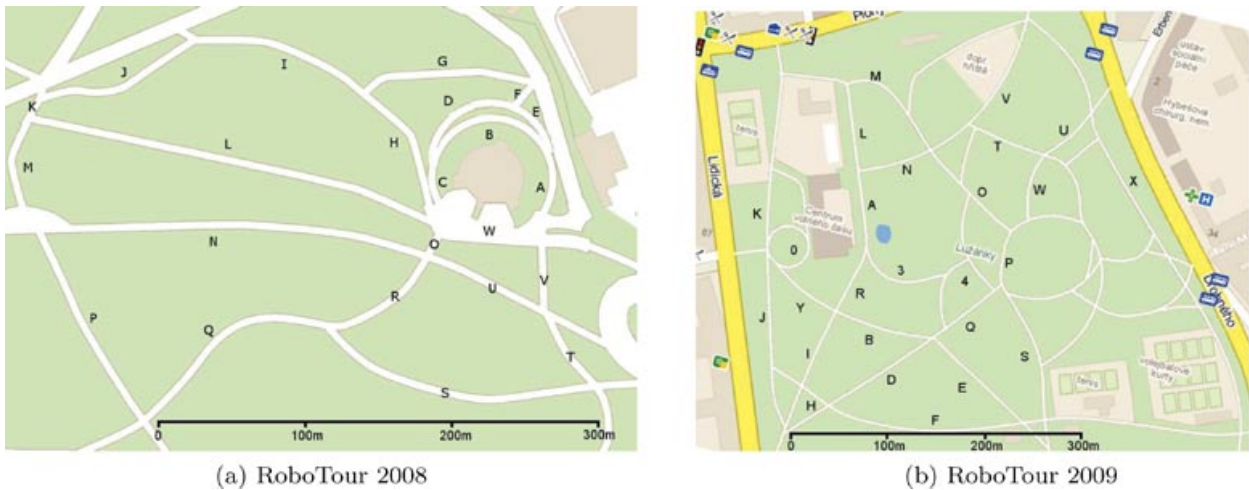


Figure 11. The position errors of the landmark deficiency (night) experiment.



(a) RoboTour 2008

(b) RoboTour 2009

Figure 12. RoboTour 2008/2009 pathway maps.

teams are mostly from Czech and Slovak universities. The competition is a perfect event for an independent verification of system functions and comparison with other navigation methods. However, not only are the navigation methods evaluated, but also the complete systems including all hardware parts are tested.

Fully autonomous robots have to travel a random path in a park, stay on the pavements, and detect randomly placed obstacles in this challenge. A map of the park with its pathways (designated by letters) is given to the teams in advance. The competition consists of several rounds, each with a different path. Thirty minutes before each round, referees choose a random closed path and announce it as a sequence of letters. Competing teams place their robots at the starting positions and execute their autonomous navigation algorithms. Robots must travel without leaving the path-

way and without colliding with any random obstacles. The robot score is determined according to its traveled distance. In 2008, the competition was held in Stromovka park⁶ in Prague, Czech Republic. One year later, the contest moved to park Lužánky⁷ in Brno, Czech Republic.

The Stromovka park pathways were mapped 2 days prior to the competition. The competition had five rounds with different pathways; see Table II and Figure 12(a). The robot completed the required path four times of these five attempts. During two of these attempts, the robot did not leave the pathway at all, and during two others, the robot had partially left (i.e., with two side wheels) the pathway. In

⁶50°6'18.778"N, 14°25'33.395"E.

⁷49°12'25.516"N, 16°36'29.81"E.

Table II. RoboTour 2008 and 2009 paths.

Pathway sequence length (m)			
RoboTour 2008		RoboTour 2009	
ABCW	250	ALK0JIR	800
ORQPMKJIH	850	BESQDGHJ0Y	1,050
ABCHGFDCW	500	34PWTMLA	750
HGFEAWOUVW	600	0YR34SFHJ	600
UTSROCEBCW	700		
Total	2,900	Total	2,200

cases when the robot left the pathway partially, it was left to continue moving (without additional scores) and reached the goal area. One failed attempt was caused by a battery failure.

The competition in Lužánky park was performed in a larger part of the environment; hence the mapping took 3 days. The total length of the mapped pathways was 8 km; the map consisted of approximately one million landmarks, which took 834 MB of disk space. The competition had four rounds; see Table II and Figure 12(b). The robot was able to complete the required paths two times. One attempt failed due to a wrong compass reading during the path learning, but after a manual correction of the robot heading, the robot caught up and reached the goal area. The other failed attempt was caused by a human factor.

Although the performance was not perfect, the robot was able to travel the required trajectory, and our team reached the first rank for both events in 2008 and 2009.

5.8. Experiment Summary

The results of the aforementioned experiments not only confirm theoretical assumptions stated in Section 3, but they also show that our method is feasible for use in real-world conditions. The proposed method is able to cope with diverse terrain, dynamic objects, obstacles, systematic errors, variable lighting conditions, and seasonal environment changes. The summary of experiments in Table III indicates that the localization precision of our method is

slightly worse than in closely related methods presented in Royer et al. (2007) and Zhang and Kleeman (2009). Lower precision is probably caused by heavy reliance on odometry and suboptimal use of visual information.

Compared to the similar method presented in Chen and Birchfield (2009), our method accuracy and repeatability is better in outdoor environments. A probable reason for this is that we used a modulus to determine robot heading. Chen and Birchfield (2009) use a mean of horizontal deviations, which is less robust to data association errors.

6. CONCLUSION

A simple navigation method based on bearing-only sensors and odometry was presented. In this method, a robot navigating a known environment uses a map of the environment and a camera input to establish its heading, while measuring the traveled distance by odometry. We claim that this kind of navigation is sufficient to keep the robot position error limited. This claim is formulated as a closed-path stability property and proved for polygonal paths with at least two noncollinear segments. The property allows us to estimate the robot position uncertainty based on the landmark density, robot odometry precision, and path shape.

The proposed method was experimentally verified by a mobile robot with a monocular camera. The robot builds a SURF-based (Bay et al., 2006; Cornelis & Van Gool, 2008) landmark map in a guided tour. After that, it uses the aforementioned method to autonomously navigate in the mapped environment.

We conducted experiments indicating that theoretical results and assumed conditions are sound.

The proposed navigation method has surprising properties different from the properties of other navigation and localization methods, mainly the following:

- The robot can perform 2D localization by heading estimation, which is a one-degree-of-freedom method.
- If the robot travels between two points, it is better to use a “zigzag” trajectory rather than a straight one.
- Traveling a closed trajectory might reduce the robot position uncertainty.

Table III. Proposed method accuracy and repeatability in various scenarios.

	Indoor		Outdoor		
	Clear	Obstacles	Long term	1 day	Night
Accuracy (m)	0.10	0.16	0.24	0.25	0.32
Repeatability (m)	0.10	0.08	N/A	0.14	0.16
Loop length (m)	20	20	50	1,040	330

We believe that the convergence proof does not apply only to our system but is valid for many other algorithms. The proof suggests that any algorithm that decreases the lateral position error of a robot is stable for closed polygonal trajectories. This might be the case for even simpler and faster methods, such as the one presented in Zhang and Kleeman (2009). However, this is merely a hypothesis, which needs to be thoroughly examined.

The fundamental limitation of our method is its reliance on odometric measurements. Other visual-based navigation methods use odometry only as an auxiliary measurement or do not require odometry at all. Therefore, these methods would perform better in scenarios in which wheel slippages have to be taken into account. Although our method is limited in application and its precision is lower compared to methods presented in Royer et al. (2007) and Zhang and Kleeman (2009), we believe that it is interesting from an academic point of view.

In the future, we would like to test our algorithm with robots that have only imprecise odometry or inaccurate dead reckoning. The preliminary tests conducted with the AR-Drone quadrotor helicopter, which estimates the traveled distance by accelerometers, seem to be promising.

7. APPENDIX A: INDEX TO MULTIMEDIA EXTENSIONS

The video is available as Supporting Information in the on-line version of this article.

Extension	Media type	Description
1	Video	Algorithm 1 implemented on a UAV

8. APPENDIX B

This Appendix presents values measured during experiments.

8.1. Stability of Robot Position for Different Types of Paths

Table B.I contains data measured during the first experimental scenario presented in Section 5.1. It shows the measured positions for the paths that do and do not retain the stability property. Note that line (degenerate) paths do not correct the longitudinal position deviation, which is in accordance with the convergence proof presented in Section 3.

Table B.I. Convergence for degenerate and normal paths: indoors.

Loop	Position relative to start (m)					
	Line (degenerate) path			Square (normal) path		
	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$
00	0.00, 0.00	0.00, 1.50	1.50, 0.00	0.00, 0.00	0.00, 1.50	1.50, 0.00
01	-0.02, 0.00	0.05, 0.55	1.46, -0.24	0.08, -0.02	0.12, 0.58	0.32, -0.06
02	-0.03, -0.04	-0.03, 0.27	1.49, -0.30	0.02, 0.04	0.02, 0.23	0.05, -0.02
03	-0.03, -0.02	-0.06, 0.14	1.53, -0.32	-0.02, 0.11	0.04, 0.06	0.10, 0.12
04	-0.05, 0.03	-0.03, 0.13	1.53, -0.25	0.10, 0.07	-0.07, -0.03	0.05, 0.15
05	-0.05, 0.00	-0.03, 0.10	1.56, -0.27	0.10, 0.11	-0.01, 0.00	0.05, 0.09
06	-0.04, 0.03	-0.03, 0.10	1.55, -0.25	0.08, 0.00	-0.02, 0.02	0.00, -0.16
07	-0.05, 0.04	-0.05, 0.16	1.53, -0.22	0.01, 0.07	0.04, -0.12	-0.03, 0.00
08	-0.05, 0.06	-0.05, 0.00	1.54, -0.25	0.00, -0.01	0.05, 0.11	0.07, 0.04
09	-0.04, 0.05	-0.06, 0.02	1.53, -0.15	-0.01, 0.02	0.05, -0.05	0.07, 0.06
10	-0.04, 0.08	-0.05, 0.13	1.53, -0.21	0.00, -0.04	-0.08, -0.16	0.09, 0.13
11	-0.06, -0.09	-0.04, 0.16	1.54, -0.25	-0.04, -0.14	0.02, -0.11	0.12, 0.02
12	-0.05, 0.01	-0.04, -0.07	1.54, -0.31	0.02, -0.02	0.05, 0.02	0.02, -0.11
13	-0.05, 0.04	-0.08, 0.06	1.55, -0.27	0.03, -0.06	0.00, 0.06	-0.01, -0.12
14	-0.04, 0.05	-0.06, 0.02	1.53, -0.31	0.06, 0.04	-0.09, -0.10	0.02, -0.09
15	-0.05, -0.04	-0.06, 0.06	1.55, -0.21	-0.01, -0.15	-0.01, 0.09	0.01, 0.00
16	-0.04, 0.00	-0.05, -0.03	1.54, -0.24	-0.02, -0.12	-0.05, 0.01	0.01, 0.05
17	-0.03, 0.10	-0.07, 0.02	1.52, -0.21	-0.03, -0.19	0.07, 0.08	0.05, -0.06
18	-0.04, 0.04	-0.09, -0.05	1.55, -0.24	0.02, -0.22	0.09, 0.05	0.01, 0.03
19	-0.03, 0.09	-0.11, 0.05	1.56, -0.10	-0.02, -0.10	0.08, 0.04	0.07, 0.09
20	-0.04, 0.08	-0.07, 0.02	1.54, -0.07	0.01, -0.15	0.02, 0.07	0.04, -0.11

Table B.II. Effect of systematic errors on navigation convergence.

Loop	Position relative to start (m) Systematic error (bias)			
	None	Camera	Odom.	Both
00	0.00, 0.00	0.00, 0.00	0.00, 0.00	0.00, 0.00
01	0.08, -0.02	-0.30, 0.37	0.08, 0.09	-0.16, 0.28
02	0.02, 0.04	-0.28, 0.39	0.09, 0.13	-0.15, 0.38
03	-0.02, 0.11	-0.35, 0.44	0.13, 0.24	-0.16, 0.46
04	0.10, 0.07	-0.29, 0.38	0.15, 0.30	-0.15, 0.50
05	0.10, 0.11	-0.31, 0.50	0.17, 0.35	-0.03, 0.53
06	0.08, 0.00	-0.33, 0.52	0.13, 0.32	-0.24, 0.49
07	0.01, 0.07	-0.34, 0.49	0.19, 0.30	-0.14, 0.52
08	0.00, -0.01	-0.32, 0.46	0.13, 0.31	-0.18, 0.49
09	-0.01, 0.02	-0.33, 0.49	0.12, 0.30	-0.13, 0.50
10	0.00, -0.04	-0.36, 0.49	0.10, 0.26	-0.14, 0.49
11	-0.04, -0.14	-0.35, 0.50	0.13, 0.39	-0.33, 0.47
12	0.02, -0.02	-0.32, 0.50	0.18, 0.44	-0.39, 0.49
13	0.03, -0.06	-0.35, 0.45	0.13, 0.31	-0.38, 0.48
14	0.06, 0.04	-0.33, 0.49	0.14, 0.34	-0.18, 0.49
15	-0.01, -0.15	-0.32, 0.45	0.14, 0.29	-0.13, 0.50
16	-0.02, -0.12	-0.31, 0.40	0.11, 0.25	-0.12, 0.49
17	-0.03, -0.19	-0.36, 0.44	0.11, 0.29	-0.16, 0.51
18	0.02, -0.22	-0.31, 0.46	0.11, 0.24	-0.15, 0.54
19	-0.02, -0.10	-0.35, 0.42	0.12, 0.26	-0.21, 0.53
20	0.01, -0.15	-0.32, 0.44	0.13, 0.27	-0.12, 0.53

8.2. Effect of Systematic Errors

Table B.II contains data from the experimental scenario in Section 5.2, in which effects of the camera and the odometry bias were measured.

8.3. Obstacle Avoidance

Table B.III contains data from the experiment scenario described in Section 5.3, in which we verified the methods ability to deal with obstacles.

8.4. Environment and Lighting Changes

Table B.IV contains data from the experiment scenario described in Section 5.4, in which the algorithm was tested in an outdoor environment with long-term environment changes.

8.5. One-Day Outdoor Experiment

Table B.V contains data from experiment scenario 5.5, in which the algorithm was tested in an outdoor environment with variable terrain.

Table B.III. Positioning errors with and without obstacles.

Loop	Position relative to start (m)	
	Clear path	Obstacles
00	0.00, 0.00	0.00, 0.00
01	0.08, -0.02	0.24, 0.22
02	0.02, 0.04	0.12, 0.06
03	-0.02, 0.11	0.17, 0.08
04	0.10, 0.07	0.11, -0.08
05	0.10, 0.11	0.15, -0.08
06	0.08, 0.00	-0.05, -0.07
07	0.01, 0.07	0.14, -0.12
08	0.00, -0.01	0.15, -0.12
09	-0.01, 0.02	0.02, -0.12
10	0.00, -0.04	0.14, -0.07
11	-0.04, -0.14	0.17, -0.13
12	0.02, -0.02	0.20, -0.04
13	0.03, -0.06	0.08, -0.06
14	0.06, 0.04	0.19, -0.03
15	-0.01, -0.15	0.14, -0.03
16	-0.02, -0.12	0.13, 0.04
17	-0.03, -0.19	0.15, -0.03
18	0.02, -0.22	0.02, -0.13
19	-0.02, -0.10	0.17, -0.06
20	0.01, -0.15	0.14, -0.01

Table B.IV. Long-term algorithm reliability.

Loop	Distance to path start (m)					
	Nov	Dec	Jan	Feb	Mar	Apr
00	1.50	1.50	1.50	1.50	1.50	1.50
01	0.70	0.92	0.90	0.87	0.83	1.15
02	0.30	0.55	0.60	0.50	0.62	0.83
03	0.13	0.38	0.45	0.22	0.28	0.41
04	0.12	0.33	0.38	0.12	0.15	0.39
05	0.03	0.27	0.28	0.18	0.14	0.33
06	0.18	0.20	0.27	0.21	0.09	0.12
07	0.06	0.28	0.22	0.23	0.19	0.07
08	0.09	0.24	0.26	0.24	0.28	0.26
09	0.10	0.29	0.24	0.22	0.23	0.40
10	0.02	0.27	0.28	0.21	0.26	0.38
11	0.23	0.20	0.24	0.22	0.29	0.32
12	0.08	0.20	0.20	0.23	0.23	0.37
13	0.22	0.28	0.20	0.29	0.27	0.27
14	0.14	0.29	0.25	0.27	0.26	0.29
15	0.24	0.29	0.21	0.27	0.11	0.46
16	0.40	0.27	0.24	0.27	0.31	0.28
17	0.42	0.22	0.24	0.22	0.23	0.23
18	0.43	0.23	0.27	0.23	0.15	0.29
19	0.48	0.23	0.23	0.28	0.19	0.35
20	0.45	0.20	0.21	0.28	0.24	0.36

Table B.V. One-day outdoor experiments.

Loop	Position relative to start (m) by map age	
	Up to date	1 week
00	0.00, 1.50	0.00, 1.50
01	-0.07, 0.03	0.63, 0.15
02	-0.21, 0.18	0.59, 0.11
03	-0.34, 0.15	0.53, 0.17
04	-0.05, -0.14	0.32, 0.19
05	0.34, 0.05	-0.09, 0.22
06	-0.25, 0.06	0.15, 0.20

Table B.VI. Landmark deficiency experiment.

Loop	Position relative to start (m)
00	0.00, 1.50
01	-0.13, -0.29
02	0.21, -0.43
03	-0.09, -0.34
04	-0.32, -0.20
05	0.12, -0.14
06	-0.08, -0.32
07	-0.05, -0.13
08	-0.29, -0.24
09	-0.09, -0.33
10	-0.10, -0.36

8.6. Landmark Deficiency Experiment

Table B.VI contains data from the experiment scenario described in Section 5.6, in which the algorithm was tested during night in low-visibility conditions.

ACKNOWLEDGMENTS

We would like to thank our colleagues for valuable remarks and friends for help with the outdoor tests. The work presented in this paper has been supported by the Ministry of Education of the Czech Republic under projects MSM 6840770038 and 2C06005, by CTU research grant SGS10/185/OHK3/2T/13, and by the FP7-ICT project "REPLICATOR" No. 216240.

REFERENCES

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). SURF: Speeded up robust features. In *Proceedings of the Ninth European Conference on Computer Vision, Graz, Austria*.
 Blanc, G., Mezouar, Y., & Martinet, P. (2005, April). Indoor navigation of a wheeled mobile robot along visual routes. In *Proceedings of International Conference on Robotics and Automation, Barcelona, Spain*.

Bosse, M., Newman, P., Leonard, J. J., & Teller, S. (2004). SLAM in large-scale cyclic environments using the Atlas framework. *International Journal of Robotics Research*, 23(12), 1113–1139.
 Burschka, D., & Hager, G. (2001, May). Vision-based control of mobile robots. In *Proceedings International Conference on Robotics and Automation, Seoul, Korea*.
 Chaumette, F., & Hutchinson, S. (2006). Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4), 82–90.
 Chaumette, F., & Hutchinson, S. (2007). Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1), 109–118.
 Chen, Z., & Birchfield, S. T. (2006, May). Qualitative vision-based mobile robot navigation. In *2006 (ICRA), IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida (pp. 2686–2692)*.
 Chen, Z., & Birchfield, S. T. (2009). Qualitative vision-based path following. *IEEE Transactions on Robotics and Automation*, 25(3), 749–754.
 Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). *Principles of robot motion: Theory, algorithms, and implementations*. Cambridge, MA: MIT Press.
 Civera, J., Davison, A. J., & Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5), 932–945.
 Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., & Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In W. Burgard, O. Brock, and C. Stachniss (Eds.), *Robotics: Science and systems*. Cambridge, MA: MIT Press.
 Cornelis, N., & Van Gool, L. (2008, June). Fast scale invariant feature detection and matching on programmable graphics hardware. In *CVPR 2008 Workshop (June 27th), Anchorage AK*.
 Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 1052–1067.
 DeMenthon, D., & Davis, L. S. (1992, May). Model-based object pose in 25 lines of code. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision, Santa Margherita Ligure, Italy*.
 Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3), 229–241.
 Dlouhy, M., & Winkler, Z. (2009). Robotour outdoor delivery challenge. <http://robotika.cz/competitions/en>. Accessed June 26, 2010.
 Estrada, C., Neira, J., & Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4), 588–596.
 Frese, U. (2006). A discussion of simultaneous localization and mapping. *Autonomous Robots*, 20(1), 25–42.
 Gibbens, P., Dissanayake, G., & Durrant-Whyte, H. (2000, December). A closed form solution to the single degree

- of freedom simultaneous localisation and map building (SLAM) problem. In Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia (pp. 191–196).
- Guerrero, J. J., Martínez-Cantin, R., & Sagüés, C. (2005). Visual map-less navigation based on homographies. *Journal of Robotic Systems*, 22(10), 569–581.
- Holmes, S., Klein, G., & Murray, D. W. (2008, May). A square root unscented Kalman filter for visual monoSLAM. In 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA (pp. 3710–3716).
- Iša, J., & Dlouhý, M. (2010, September). Robotour—Robotika.cz outdoor delivery challenge. In Proceedings of the 1st Slovak–Austrian International Conference on Robotics in Education, Bratislava, Slovakia (to appear).
- Julier, S., & Uhlmann, J. (2001, May). A counter example to the theory of simultaneous localization and map building. In 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea (pp. 4238–4243).
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D), 35–45.
- Kidono, K., Miura, J., & Shirai, Y. (2000, July). Autonomous visual navigation of a mobile robot using a human-guided experience. In Proceedings of 6th International Conference on Intelligent Autonomous Systems, Venice, Italy (pp. 620–627).
- Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In ICCV '99: Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece (p. 1150).
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91–110.
- Lyapunov, A. (1992). The general problem of stability in motion. *International Journal of Control*, 55(3), 531–773.
- Martinelli, A., Tomatis, N., & Siegwart, R. (2005, August). Some results on SLAM and the closing the loop problem. In International Conference on Intelligent Robots and Systems, Edmonton, Canada (pp. 334–339).
- Matsumoto, Y., Inaba, M., & Inoue, H. (1996, April). Visual navigation using view-sequenced route representation. In Proceedings of the International Conference on Robotics and Automation, Minneapolis, MN.
- Montiel, J., Civera, J., & Davison, A. (2006, August). Unified inverse depth parametrization for monocular SLAM. In Proceedings of Robotics: Science and Systems, Philadelphia, PA.
- Mourikis, A., & Roumeliotis, S. I. (2004, September). Analysis of positioning uncertainty in simultaneous localization and mapping (SLAM). In Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS), Sendai, Japan.
- Remazeilles, A., & Chaumette, F. (2007). Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4), 345–356.
- Royer, E., Lhuillier, M., Dhome, M., & Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3), 237–260.
- Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2007, June). Large scale vision based navigation without an accurate global reconstruction. In IEEE International Conference on Computer Vision and Pattern Recognition, Minneapolis, MN.
- Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2009). A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2), 172–187.
- Svab, J., Krajník, T., Faigl, J., & Preucil, L. (2009, November). FPGA-based speeded up robust features. In 2009 IEEE International Conference on Technologies for Practical Robot Applications, Boston, MA.
- Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12), 1188–1197.
- Wilson, W., Hulls, C., & Bell, G. (1996). Relative end-effector control using Cartesian position based visual servoing. *Robotics and Automation*, 12(5), 684–696.
- Zhang, A. M., & Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. *International Journal of Robotics Research*, 28(3), 331–356.

B

KEY ARTICLE [9] - IROS 2018 (IN REVIEW)

©[2017] IEEE. Reprinted, with permission, from Tomáš Krajník, Navigation without localisation: reliable teach and repeat based on the convergence theorem, 2017.

Navigation without localisation: reliable teach and repeat based on the convergence theorem

Tomáš Krajník, Filip Majer, Lucie Halodová, Tomáš Vintr

Abstract— We present a novel concept for teach-and-repeat visual navigation. The proposed concept is based on a mathematical model, which indicates that in teach-and-repeat navigation scenarios, mobile robots do not need to perform explicit localisation. Rather than that, a mobile robot which repeats a previously taught path can simply “replay” the learned velocities, while using its camera information only to correct its heading relatively to the intended path. To support our claim, we establish a position error model of a robot, which traverses a taught path by only correcting its heading. Then, we outline a mathematical proof which shows that this position error does not diverge over time. Based on the insights from the model, we present a simple monocular teach-and-repeat navigation method. The method is computationally efficient, it does not require camera calibration and it can learn and autonomously traverse arbitrarily-shaped paths. In a series of experiments, we demonstrate that the method can reliably guide mobile robots in realistic indoor and outdoor conditions, and can cope with imperfect odometry, landmark deficiency, illumination variations and naturally-occurring environment changes. Furthermore, we provide the navigation system and the datasets gathered at www.github.com/gestom/stroll_bearnav.

I. INTRODUCTION

A considerable progress in visual-based systems capable of autonomous navigation of long routes was achieved during the last decade. According to [1], [2], vision-based navigation systems can be divided in map-less, map-based, and map-building based. Map-less navigation systems such as [3], [4], [5] aim to recognise traversable structures (e.g. roads, pathways, field rows, etc.) and use these to directly calculate motion commands. Map-based navigation systems rely on environment models that are known apriori [6]. Map-building-based systems rely on maps for localisation and navigation as well, but they can build these maps themselves. Some of these vision-based methods can build maps and localise the robot at the same time and – these are referred to as visual SLAM (Simultaneous Localisation and Mapping).

One of the most known visual SLAM systems, Monoslam [7], processes an image stream from an unconstrained-motion monocular camera in real-time, obtaining the trajectory of the camera and a 3D map of salient visual features [7]. Another method, the ORB-SLAM [8], [9], allows exploiting stereo and depth information to build both sparse and dense maps of the environment while estimating the camera motion in 6D. Unlike the aforementioned systems, LSD-SLAM [10] and DSO [11] do not rely on

image feature extraction but create dense, large-scale maps by directly processing the intensities of the image pixels. A recent, comprehensive review of SLAM systems is presented in [12]. While being an important component of many navigation systems, SLAM by itself does not control the mobile robot motion, and thus it does not navigate robots per se. Rather than that, it provides an environment map and a robot position estimate to the motion planning modules, which then guide the robot towards the desired goal.

Thus, one of the typical use of SLAM methods in practice is ‘teach-and-repeat’, where a robot uses SLAM during a teleoperated drive, creating a map of the environment and use this map later on to repeat the taught path [13], [14], [15]. This technique is analogous to a popular practice in industrial robotics, where an operator teaches a robot to perform some task simply by guiding its arm along the desired path. The systems that use SLAM methods within the teach-and-repeat paradigm were extended by techniques like experience-based localisation [16], feature selection [17] or intrinsic image [18], enabling their long-term deployment in environments that are challenging due to appearance changes [19], [20], [21] or difficult illumination conditions [22].

In the long-term deployments, it’s assumed that the robots start and end their forays at their recharging stations with a known position, and occasional loss of localisation is solved by request for human intervention [27]. Thus, teach-and-repeat methods employed in long-term scenarios do not typically address the kidnapped robot problem.

Some of the teach-and-repeat systems do not rely on SLAM-build 3D maps of the environment. For example [23] creates a visual path, which is a set of images along the human-guided route, and then employs visual servoing to guide the robot across the locations these images were captured at. Similarly, [24] represents the path as consecutive nodes, each containing a set of salient visual features, and uses local feature tracking to determine the robot steering to guide it to the next node. The authors of [25] extract salient features from the video feed on-the-fly and associate these with different segments of the teleoperated path. When navigating a given segment, their robot moves forward and steers left or right based on the positions of the currently recognised and already mapped features. The segment end is detected by means of comparing the mapped segment’s last image with the current view. The same navigation principle was recently deployed on micro aerial vehicles in [26].

At the time when the original SLAM-based teach-and-repeat framework was published [14], another article [28] mathematically proved that (while being useful) explicit

Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University tomas.krajnik@fel.cvut.cz
The work has been supported by the Czech Science Foundation projects 17-27006Y.

localisation is not necessary for teach-and-repeat scenarios. The results of [28] indicate that to repeat a taught path, camera input needs to be used only to correct the robot heading, leaving the position estimation to odometry. The mathematical proof showed, that for polygonal paths, the heading corrections suppress odometric errors, preventing the overall position error of the robot to diverge. The proof was supported by several long-term experiments [28], [29], where a robot repeatedly traversed long paths in natural environments over the period of one year. Thus, this SLAM-less method showed good robustness to environment changes even without using experience-based or feature-preselection techniques [29]. However, the mathematical proof in [28] was limited to paths consisting of straight segments. Thus, the system based on [28] could be taught only polygonal paths in a turn-move manner, and even a slight change of the movement direction during the teaching phase required to stop and turn the robot, which made the teaching tedious and deployment of the system rather impractical.

In this paper, we reformulate the problem presented in [28] in a continuous rather than a discrete domain. This allows us to simplify and extend the mathematical proof of [28] to any continuous trajectory, not only polygonal paths as in [28]. A navigation system based on this extended formulation can thus be taught smooth, curved paths, which makes its teaching faster and navigation more efficient.

The main contribution of this paper is the aforementioned mathematical proof which indicates that in teach-and-repeat scenarios, a robot can use its camera information only to correct its heading and it does not have to build metric maps or perform explicit localisation. Based on this principle, we implement a teach-and-repeat navigation system and use it to verify the aforementioned hypothesis in realistic conditions. In a series of experiments, we compare the behaviour of the system with the proposed mathematical model and demonstrate the system's ability to reliably guide robots along the taught paths in adverse illumination conditions including night. Furthermore, we present the system as an open-source, ROS-based package and accompany the software with the datasets gathered during the experiments performed [30].

II. NAVIGATION STABILITY

In this section, we analyse how heading correction influences the overall position error of a robot as it travels along a taught path. At first, we establish a model of the robot movement along the desired path and we outline a model of the robot position error. Then, we examine conditions under which the robot position error does not diverge.

A. Paths with non-zero curvature

Let us assume that a human operator placed a robot at an initial position $x_p(0), y_p(0)$ and then she drove the robot by controlling its forward v and angular ω velocities. Let the robot record its v and ω velocities together with the features detected in its camera image and let the robot index the features and velocities by the distance travelled. Thus, the taught path \mathcal{P} is defined by the initial point $x_p(0), y_p(0)$,

velocity functions $v(d)$, and $\omega(d)$, where d represents the length of the path from $x_p(0), y_p(0)$ to the current position. Some locations (again indexed by d) on the trajectory are also associated with image coordinates and descriptors of the image features detected in the robot camera image.

Then, assume that a robot is placed at a position $x_r(0), y_r(0)$ and it is supposed to traverse the path \mathcal{P} . The robot, having no information about its position or orientation, has to assume that it is at the start of the taught path, and thus its position estimate is $x_p(0), y_p(0)$. Therefore, the robot sets its forward and angular velocity to $v(0)$ and $\omega(0)$, respectively. The robot also retrieves the image features it saw at $d = 0$, matches them to the features in its current view and adjust its angular velocity in a way, which would decrease the horizontal distances (in the image coordinates) of the matched feature pairs. As the robot moves forwards, it calculates the distance travelled d and sets its forward velocity to $v(d)$ and angular velocity to $\omega(d) - \alpha\kappa$, where κ is the most frequent horizontal displacement of the matched feature pairs.

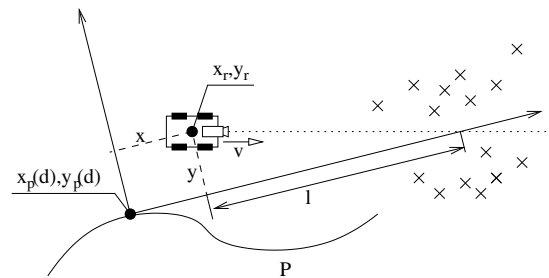


Fig. 1: Robot position error chart. The robot at a position x_r, y_r uses a local feature map of the taught path \mathcal{P} at the position $x_p(d), y_p(d)$.

Since the robot camera was facing in the direction of the robot movement during the teaching phase, each feature that is in the current map lies in the vicinity of the tangent to the path at some finite distance, see Figure 1. Assuming that the robot is able to turn fast enough to keep κ low (i.e., it is able to turn so that the horizontal distances of the mapped/detected feature pairs are low), the direction of its current movement intersects the aforementioned tangent at a certain distance l . The distance l is given by the spatial distribution of the features in the traversed environment – low l is typical for cluttered environments and high l occurs mostly in large open areas.

Figure 1 illustrates that the error of the robot position estimate $x(t), y(t)$ can be defined as its position in a local coordinate frame defined by the location and orientation of the local map, which the robot uses to determine its velocities. In other words, the robot position error $x(t), y(t)$ is defined by its position $(x_r(t), y_r(t))$ relatively to $x_p(d), y_p(d)$. In order to analyse the evolution of the position error during the robot navigation, we form a differential equation

describing $(\dot{x} = dx(t)/dt, \dot{y} = dy(t)/dt)$:

$$\begin{aligned} \dot{x} &= +\omega y - v_r + v + s_x \\ \dot{y} &= -\omega x - vyl^{-1} + s_y, \end{aligned} \quad (1)$$

where the terms ωy , ωx and $-v_r$ are caused by the rotation and translation of the local coordinate system as the point $x_p(d), y_p(d)$ moves along the intended path, the term $+v$ is the movement of the robot along the path, vyl^{-1} reflects the influence of the visual-based heading correction, and s_x and s_y represent perturbations caused by image processing imperfections, odometric errors and control system inaccuracies. Since the errors s_x and s_y directly influence the velocities \dot{x} , \dot{y} of the robot in our model, they encompass not only additive errors, such as occasional wheel slippage or imperfect image feature localisation, but also systematic errors, such as miscalibration of the odometry causing the v_r and v to be different, or camera misalignment causing an offset in robot heading corrections. Assuming that v_r and v are almost identical, and that their difference is included in the perturbation s_x , we can rewrite (1) in a matrix form:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & +\omega \\ -\omega & -vyl^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} s_x \\ s_y \end{pmatrix}, \quad (2)$$

which is a system of linear differential equations.

In general, a linear continuous system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{s}$ is stable, i.e., the \mathbf{x} does not diverge, if the real components of all eigenvalues of the matrix \mathbf{A} are smaller than 0. Thus, the robot position error x, y does not diverge if the matrix \mathbf{A} from (2) has all real components of its eigenvalues negative. Eigenvalues of a 2×2 matrix are obtainable by solving a quadratic equation

$$\lambda(\lambda + v/l) + \omega^2 = 0, \quad (3)$$

$$\lambda_{1,2} = \frac{-v/l \pm \sqrt{(v/l)^2 - 4\omega^2}}{2}. \quad (4)$$

In cases when $(v/l)^2 < 4\omega^2$, the expression $\sqrt{(v/l)^2 - 4\omega^2}$ is an imaginary number, and $\lambda_{1,2}$ are complex conjugates with the real component equal to $-v/(2l)$. Since v and l are always positive, $-v/(2l)$ is always negative, which ensures the system stability and non-divergence of the robot position error x, y .

In the case of $v/l \geq 4\omega^2$, the result of the square root is positive and the real part of the λ_2 eigenvalue

$$\lambda_2 = \frac{-v/l - \sqrt{(v/l)^2 - 4\omega^2}}{2} \quad (5)$$

is always negative. The eigenvalue λ_1 , which is calculated as

$$\lambda_1 = \frac{-v/l + \sqrt{(v/l)^2 - 4\omega^2}}{2}, \quad (6)$$

is non-negative only if ω equals to 0. *This means, that if the robot does not move along a straight line, both eigenvalues of \mathbf{A} are lower than 0, and therefore, both longitudinal (x) and lateral (y) components of the robot position error do not diverge even if the robot uses its exteroceptive sensors only to correct its heading.* \square

B. Paths containing straight segments

According to (1) and (2), if a robot travels along a straight line, its longitudinal position error, represented by x in our model, gradually grows due to the perturbances s_x , which are caused primarily by odometric drift. Let assume that the taught path consists of straight segments conjoined by segments with non-zero curvature. Let assume that a robot started to traverse a straight segment at time t_0 and ended its traversal at t_1 . Its error after the segment traversal is obtainable by integrating (2) over time as:

$$\begin{pmatrix} x(t_1) \\ y(t_1) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-\frac{v}{l}(t_1-t_0)} \end{pmatrix} \begin{pmatrix} x(t_0) \\ y(t_0) \end{pmatrix} + \begin{pmatrix} b_{x0} \\ b_{y0} \end{pmatrix}. \quad (7)$$

Now, let assume that from the time t_1 to the time t_2 , the robot traverses a segment with non-zero curvature. Regardless of the segment shape and curvature, the magnitude of x and y decreases (see (2)), and thus we can state that

$$\begin{pmatrix} x(t_2) \\ y(t_2) \end{pmatrix} = \mathbf{N}_1 \begin{pmatrix} x(t_1) \\ y(t_1) \end{pmatrix} + \begin{pmatrix} b_{x1} \\ b_{y1} \end{pmatrix}, \quad (8)$$

where the eigenvalues of the matrix \mathbf{N}_1 are lower than one. Rewriting (7) in a compact form as $\mathbf{x}_1 = \mathbf{N}_0\mathbf{x}_0 + \mathbf{b}_0$, and (8) as $\mathbf{x}_2 = \mathbf{N}_1\mathbf{x}_1 + \mathbf{b}_1$, and substituting (7) into (8) results in:

$$\mathbf{x}_2 = \mathbf{N}_1(\mathbf{N}_0\mathbf{x}_0 + \mathbf{b}_0) + \mathbf{b}_1 = \mathbf{N}_1\mathbf{N}_0\mathbf{x}_0 + (\mathbf{N}_1\mathbf{b}_0 + \mathbf{b}_1). \quad (9)$$

Equation 9 represents a discrete system, which allows to estimate the robot position error after traversing a straight segment followed by a curved one. Since the largest eigenvalue of \mathbf{N}_0 equals to 1 (see (7)) and both eigenvalues of \mathbf{N}_1 are smaller than 1, the eigenvalues of their product $\mathbf{N}_1\mathbf{N}_0$ are also smaller than 1. This means that the discrete system (9) is stable. *Thus, position error of a robot, which repeatedly traverses a path formed of conjoined straight and curved segments does not diverge even if it is using its exteroceptive sensors only to correct its heading.* \square

C. Convergence basin

The model that we established assumes that during the repeat phase, the robot perceives at least some image features that it saw during the teaching. Thus, if the robot's initial position in the repeat phase is too far from the origin of the teaching step, or if the robot deviates from the path too much, the mapped features will not be in its field of view and the navigation will fail. The actual position error that would cause the navigation to fail depends on robot's camera, path shape and feature distance. In our experiments, we started the repeat phase with an initial position error exceeding 1 m, which is approximately an order of magnitude higher than the accuracy of the navigation, see Figures 5, 7, and 9. Thus, the typical navigation inaccuracy caused by s_x and s_y in (2) is very unlikely to deviate from its path beyond the point where it can correct its position error. In practice, a robot can monitor the consistency of the feature matching and when there are not enough correspondences, it can request human intervention.

III. NAVIGATION METHOD DESCRIPTION

The considered navigation system works in two steps: teach and repeat. In the teaching phase, a robot is guided by an operator along a path, which is the robot supposed to autonomously navigate in the repeat phase. During the teaching, the robot extracts salient features from its onboard camera image and stores its current travelled distance and velocity. During the autonomous navigation, the robot sets its velocity according to the travelled distance and compares the image coordinates of the currently detected and previously mapped features to correct its heading.

A. Image processing

The feature extraction method which detects salient objects in the onboard camera image is a critical component of the navigation system because it is the only mechanism which the robot employs to reduce its position error. Based on the results from our previous work on image feature stability in changing outdoor environments [31], we decided to use the Speeded Up Robust Features (SURF) [32] and a combination of the AGAST [33] and BRIEF [34] methods.

The feature extraction is composed of two steps, detection of keypoints and description of their vicinity. The keypoint detection indicates points in the image, which have sufficient contrast that makes them easy to localise and track. In the case of SURF, the keypoint detection is based on the approximation of Hessian matrix determinant [32], while AGAST [33] uses an optimised pixel brightness testing scheme around the keypoint candidate. To form the description of a particular keypoint, BRIEF [34] calculates a binary descriptor by comparing brightnesses of randomly-chosen pixel pairs around the keypoint. The advantage of this descriptor is an efficient calculation, low memory requirements, and rapid matching. The SURF-based descriptor is based on image intensity gradients near the keypoint [32]. While being slower to calculate and match, it is more resistant to large viewpoint changes.

Once the keypoints are detected and described, they can be matched to the keypoints stored in a map and the associations can be used to correct the robot heading. The quality of the features depends on the quality of the input image stream, which, in outdoor environments, suffers from varying illumination. To compensate the illumination instability, we select the exposure and brightness of the robot camera based on the results from [35].

B. Teaching (mapping) phase

During the phase, the robot is driven through the environment by a human operator. The robot continuously measures the distance it travelled and whenever the operator changes the forward or angular velocity, the robot saves the current distance and the updated velocity values – we refer to this sequence as to a “path profile”. Additionally, the robot continuously extracts image features from its onboard camera image and every 0.2 m, it saves the currently detected image features in a local map, which is indexed by the current distance the robot travelled.

C. Repeat (navigation) phase

At the start of this phase, the robot loads the path profile and creates a distance-indexed list of the local maps containing the image features. Then, it sets its forward and angular velocities according to the first entry of the path profile and it loads the first local map containing data about image features visible at the start of the path. As the robot moves forwards, it extracts image features from its onboard camera image and matches them to the ones loaded from the local map. The differences of the horizontal image coordinates of the matched feature pairs (i.e., the positions of the features in the camera image relative to the positions of the features in the preloaded map) are processed by a histogram voting method. The maximum of the histogram indicates the most frequent difference in the horizontal positions of the features, which corresponds to the shift of the image that was acquired during the mapping phase relative to the image that is currently visible from the onboard camera. This difference is used to calculate a corrective steering speed, which is added to the speed from the velocity profile.

If the histogram voting results are inconclusive due to the low number of features extracted, e.g., when the robot faces a featureless wall, the camera image is over- or under-exposed, etc., the corrective angular speed is not added to the one from the velocity profile. In a case the visual information is not sufficient to determine the heading, the robot simply steers according to the path profile data. As the robot proceeds forwards along the taught path, it loads local maps and path profile data that correspond to the distance travelled and repeats the steps described above. Thus, the path profile allows the robot to steer approximately in the same way as during the teaching phase, and the image matching corrects the robot heading whenever it deviates from the intended path.

The principal advantage of the system is its robustness to feature deficiency and uneven feature distribution in the camera image. This makes the system robust to environment appearance changes and adverse lighting conditions. The histogram voting-based heading corrections are demonstrated in videos at [30].

D. System implementation

The navigation system was implemented in the Robotic Operating System (ROS), in particular the version Kinetic. The system structure is shown in Figure 2. The *feature extraction* node extracts image features from the robot camera and passes them to the *mapper* and *navigator* nodes. The *odometry monitor* node receives data from robot odometry and measures travelled distance. It also sends special messages every time the robot passes a given distance (e.g., 0.2 m), which is used by the *mapper node*, see Section III-B. The *mapper* node receives features from the *feature extraction* node and saves them into the local map when it receives the aforementioned message from the *odometry monitor* node. It also saves the path profile. The *map preprocessor* node loads all local maps and path profile, and then sends them to the *navigator* node based on the travelled

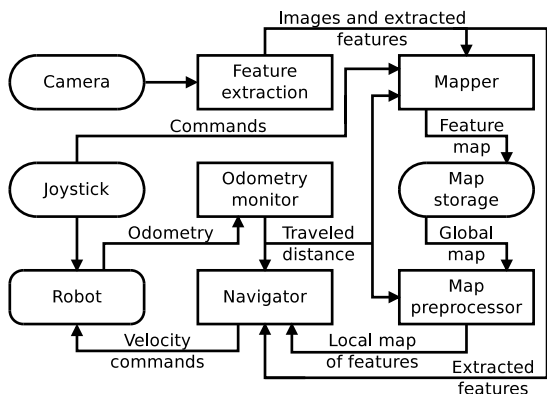


Fig. 2: Software structure of the presented system

distance received from the *odometry monitor*. The *navigator* node receives the velocity profile and local feature maps and it matches the features from the maps to the currently visible features from the *feature extraction* node. It performs the histogram voting described in Section III-C, calculates the robot velocities and steers it along the path.

All the aforementioned modules were implemented as ROS action servers with dynamically reconfigurable parameters. Therefore, the robot operator can change their parameters during runtime or activate and deactivate the modules in case they are not necessary to run during the teaching or replay phase. Action servers also provide the operator with feedback that shows the current state of the system. Thus, the operator can see the currently used map, path profile data, number of visible image features, results of the histogram voting method etc. All the aforementioned modules are available as C++ open source code at [30].

IV. EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed navigation system consists of 3 different experiments performed with 2 different robotic platforms. The aim of the first experiment was to demonstrate that without the visual feedback or path profile information, the robot is not able to repeat the taught path. The second experiment, which is performed under controlled conditions and an external localisation system, demonstrates the accuracy of the motion model established in Section II and the ability of a minimalistic robotic platform to converge to the taught trajectory during multiple traversals of the intended path. The third experiment demonstrates the trajectory convergence in challenging outdoor conditions including night.

A. Platforms

For indoor experiments, we used a lightweight robot based on an MMP-5 platform made by TheMachineLab 3. Its base dimensions are $0.3 \times 0.3 \times 0.1$ m, its mass is 2.2 kg, and it can carry additional 2 kg payload while achieving speeds over 1.2 ms^{-1} . The robot has a four-wheel differential drive with a control unit which allows setting PWM signal duty on individual motors by a simple serial protocol. Since the

platform does not provide any odometric information, we estimate the travelled distance simply by the time and motors' PWM duty. Its vision system is based on a single

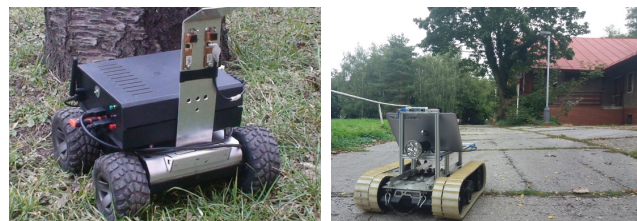


Fig. 3: MMP-5 and Cameleon ECA used in our experiments.

USB camera, which provides 320×240 color images with a $45^\circ \times 35^\circ$ field of view. The computer is based on an AT3IONT-I miniATX board with an Intel Atom 330 CPU running at 1.6GHz with a 2GB RAM. Due to its computational constraints, this robot is using the AGAST/BRIEF features.

For outdoor experiments, we used a heavy duty, tracked platform, called CAMELEON ECA, which is equipped with onboard PC and two cameras as primary sensors. Unfortunately, the cameras are positioned very low, which causes problems in grassy terrains and the onboard PC is too slow. Thus, we equipped the robot with a superstructure with several equipment mounts, where we placed the TARA USB stereo camera (we use only the left camera image in our experiments), Intel i3 laptop with 8GB RAM and the Fenix 4000 lumen torch, see Figure 3.

B. Experiment I: Proof-of-concept

To evaluate the system's ability to repeat the taught path and to correct position errors that might arise during the navigation, we have taught the CAMELEON platform a closed, approximately 25 m long path in the outdoor environment in the Czech Technical University campus. The shape of the trajectory is a closed, smooth, oval-shaped curve. After mapping, we let the robot to drive along the taught path repeatedly. Every time the robot completed a path loop, we measured its distance relative to the path end/start. In this way, we quantitatively assess the robot's ability to adhere to the path it has been taught. The experiments were performed during the evening, and therefore, the lighting conditions changed from high to low illumination, which made the image-based navigation particularly difficult. Facing changes in environment and lighting conditions is inevitable for long-term navigation, that is why we chose this particular setup.

To demonstrate the interplay between the path profile velocity setting and vision-based heading correction, we let the robot to drive the path autonomously using only the velocities remembered from the teaching phase (i.e. path profile), then only using visual information and then the combination of these. In the first test, we have deactivated the vision-based heading corrections and we let the robot move according to the path profile only – this corresponds

to the term $v_y l^{-1}$ in (2) being equal to 0. The model (2) predicts that the errors of s_x and s_y will gradually accumulate, drifting the robot off the taught path. As predicted by the model, during this trial the robot slowly diverged from the path because of the inaccurate odometry.

During the second trial, we have let the robot run with the method described in [28]. Thus, the robot did not use the path profile information, but it moved forward with a constant speed and steered its heading according to the results of the image feature matching and histogram voting. In this case, the robot diverged from the taught path as soon as it was supposed to perform a sharper turn, because the visual heading correction by itself could not perform sharp turns.

The final trial used both path profile and visual feedback as described in Section III. To verify if the robot can correct position errors that arise during navigation, we have started the autonomous navigation, not at the path start, but 1.2 m away. The reason for the robot displacement is to demonstrate that unlike in the previous two cases, where the position error diverged, a combination of the vision and path profile information would allow the robot to suppress the error and adhere to the taught trajectory. As expected, each time the robot completed the taught path, its position error decreased, which confirms the assumptions stated in Section II.

C. Experiment II: Position error model verification

The indoor experiments were meant to compare the real system behaviour with the model of the robot movement. In these experimental trials, we used the MMP-5 robot platform equipped with a circular marker, which allows for an accurate tracking of the robot position. In the first trial, we guided the robot along a 10 m long oval-shaped path consisting of two half-circle segments connected with two straight lines, see Figure 4. Then, we displaced the robot 1 m away from the path start and let it traverse along the path autonomously 20 times while recording its position with the external localisation system [36].

Figure 4 shows the robot trajectory during the autonomous traversals, which slowly converges to the mapped path. Each time the robot finished one path traversal, we measured its position relative to the path start. Fig. 5 shows that the position error diminished after two loops, stabilising at 7 cm.

The convergence of the robot position during the autonomous travels is visible in Figure 4 and the robot position error evolution along the first path traversal in Figure 5. The error evolution confirms the mathematical model presented in Section II – one can observe that during the first 5 meters, where the path is curved, the position error diminished rapidly. Once the robot starts to traverse the straight segment, the position error stabilises and it starts to diminish once again when the robot starts to traverse the second semi-circular path segment. Since the robot is nonholonomic, skid-steer drive, the convergence of the position error implies that its orientation conforms to the model (1).

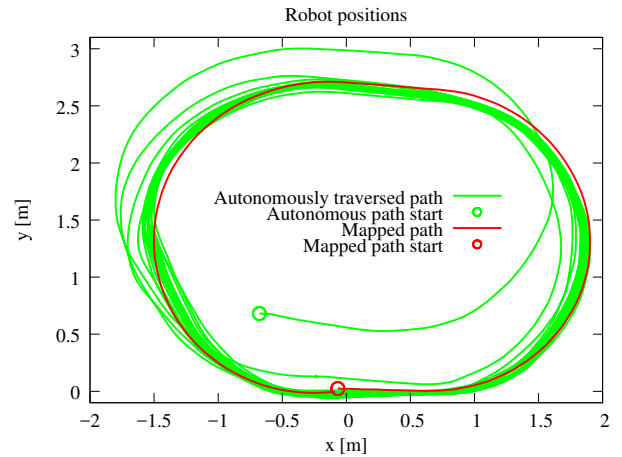


Fig. 4: Indoor trial I: Robot path during teach and repeat.

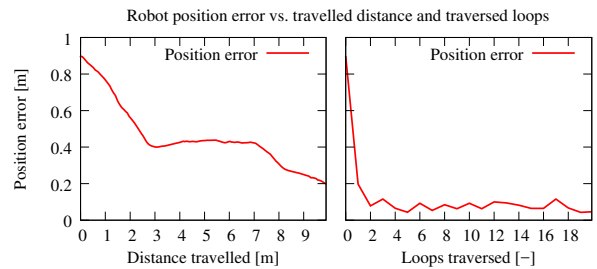


Fig. 5: Indoor trial I: Robot position error during the first autonomous path traversal (left) and during the 20 autonomous path repeats (right).

In the second trial, we guided the robot along a 17 m long eight-shaped path consisting of four half-circle segments connected with three straight ones and let it traverse the path 19 times, see Figure 6. This trial was performed in a larger

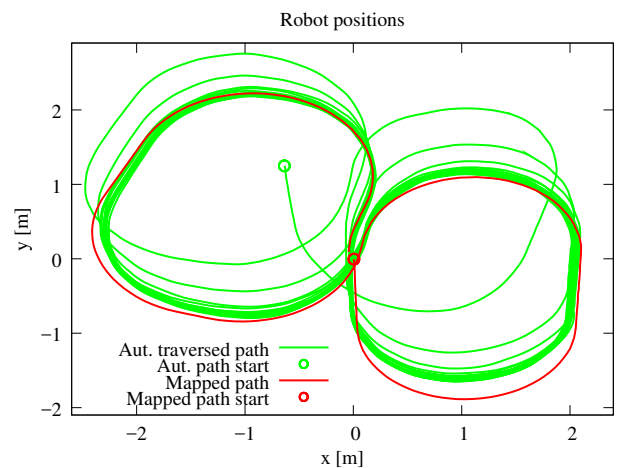


Fig. 6: Indoor trial II: Robot path during teach and repeat.

hall than the previous one, and therefore, the average distance of landmarks is bigger than in the previous case, causing

slower convergence of the robot trajectory, see Figure 7. Furthermore, the start and end of the taught path are about 0.15 m apart, which causes the robot to start with 0.15 m error during subsequent path traversals, which is notable in Figure 6, where the first part of the repeated path is slightly displaced from the taught one. Similarly to the

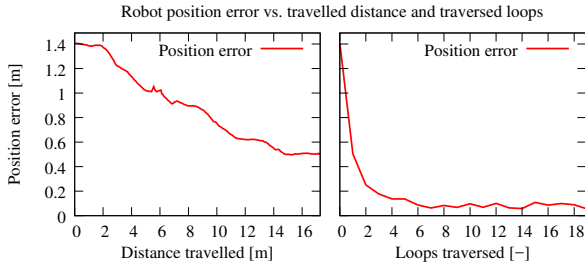


Fig. 7: Indoor trial II: Robot position error during the first autonomous path traversal (left) and during the 19 autonomous path repeats (right).

previous case, the error evolution shown in the left part of Figure 6 conforms with the mathematical model presented in Section II – the error reduction is slower during traversal of the straight segments of the taught path. In these experiments, a robot without any odometric system, equipped with a low resolution, uncalibrated camera which suffered from motion blur (see [30]) reliably traversed over 700 m, reducing the initial ~ 1 m position errors below ~ 0.1 m.

D. Experiment III: System robustness

The purpose of final outdoor experiments is to demonstrate the ability of the system to cope with uneven terrain and variable illumination. The first trial was performed during a day, where a robot was supposed to traverse a 60 m long path at the Hostibejk hill in Kralupy nad Vltavou, see Figure 3. The second and third trials were performed at night at the same location. During the first trial, we created the map in a clear sky weather and let the robot traverse the path 7 times one month later during a cloudy day.

In the second trial, which was performed at night, we attached a 4000 lumen searchlight to the robot’s superstructure. The location of the trial was free of any artificial light sources; so, the most of the robot path, it saw only parts of the scene, which it illuminated itself, see Figure 8. After creating the local map, we displaced the robot by 1.5 m and let it traverse the path 7 times (the number of traversals is limited by the capacity of the searchlight batteries).

The third trial was performed at the same location, which at this time was partially illuminated, because three of the local street lamps were repaired in the meantime. Again, after teaching the robot a 60 m long path, we displaced the robot by 1.5 m and let it traverse the path 7 times. During these trials, we initiated the autonomous run 1.5 m from the taught path start and then measured the robot displacement from the path start every time it completed the taught path. Figure 9 shows that the initial position error quickly diminished to values around 0.2–0.3 m. In these experiments,



Fig. 8: Outdoor experiment: Hostibejk site at night and day from the robot perspective.

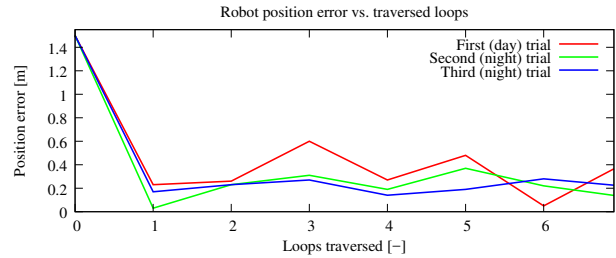


Fig. 9: Outdoor experiments: Robot position error relatively to the path start after traversing the path n times.

a tracked robot with imperfect odometry, equipped with an uncalibrated camera working in low-visibility conditions (see [30]) reliably traversed over 1.2 km, reducing the initial ~ 1.5 m position errors to ~ 0.2 – 0.3 m. Furthermore, we did 10 days of additional tests at the same site over a period of one month. In these tests, the robot successfully traversed the taught path 66 times, 21 during the day, 25 during the night and 20 during the sunset and covered distance over 4 km.

To compare the system’s robustness to the state-of-the-art SLAM-based methods, we processed the data gathered during these trials by the ORB-SLAM [8], which we modified to take into account odometric information when initialising camera position estimation. To do so, we had to calibrate the camera and process the gathered data (in the form of ROSbags) by the ORB-SLAM. In our tests, we had to replay the ROSbags from the day trial $2.5\times$ slower than in the original experiment several times because of occasional loss of tracking and subsequent breakdown of the ORB-SLAM method. Despite trying several settings of ORB-SLAM, we could not build a consistent map from the night data due to feature deficiency and motion blur. This comparison indicates the proposed method’s robustness to difficult illumination conditions.

V. CONCLUSION

We formulated a mathematical proof which indicates that in teach-and-repeat scenarios, explicit localisation of a mobile robot along the taught route is not necessary. Rather, a robot navigating through a known environment can use an environment map and visual feed to correct only its heading, while measuring the travelled distance only by its odometry. Based on this principle, we designed and implemented a simple teach-and-repeat navigation system and evaluated its performance in a series of experimental trials. These

experiments confirmed the validity of the aforementioned proof, showing that this kind of simple navigation is sufficient to keep the position error of the robot limited. The requirement to establish only the robot heading simplifies the visual processing and makes it particularly robust to situations, where the detected and mapped features cannot be associated reliably. This makes our system prone to difficult lighting conditions and environmental changes, which is demonstrated by its comparison to ORB-SLAM2. Compared to the previous work [28], the mathematical proof of bearing-only navigation presented here is simpler, shorter and is not limited to polygonal routes only. Thus, unlike in [28], where the robot could only learn polygonal routes in a turn move manner, our navigation system allows to learn arbitrarily-shaped routes, making its real-world deployment more feasible. Furthermore, we show that the robot position error is asymptotically stable, whereas in our previous work [28] we only proved its Lyapunov stability. In future, we aim to extend the system by its ability to improve its performance by exploiting the experience gathered during autonomous traversals.

REFERENCES

- [1] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.
- [2] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, 2008.
- [3] J. M. A. Alvarez and A. M. Lopez, "Road detection based on illuminant invariance," *IEEE Transactions on Intelligent Transportation Systems*, 2011.
- [4] B. Wang, V. Frémont, and S. A. Rodríguez, "Color-based road detection and its evaluation on the KITTI road benchmark," in *Intelligent Vehicles Symposium*, 2014.
- [5] B. Åstrand and A.-J. Baerveldt, "A vision based row-following system for agricultural field machinery," *Mechatronics*, 2005.
- [6] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *CVGIP: Image understanding*, vol. 56, no. 3, pp. 271–329, 1992.
- [7] S. Holmes, G. Klein, and D. W. Murray, "A Square Root Unscented Kalman Filter for visual monoSLAM," in *International Conference on Robotics and Automation (ICRA)*, 2008, pp. 3710–3716.
- [8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, 2017.
- [10] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.
- [11] R. Wang, M. Schwörer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017.
- [12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, 2016.
- [13] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human-guided experience," in *Proceedings of 6th International Conference on Intelligent Autonomous Systems*, 2000.
- [14] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010. [Online]. Available: <http://dx.doi.org/10.1002/rob.20342>
- [15] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, Sep 2007.
- [16] W. S. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *IJRR*, 2013.
- [17] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, 2011.
- [18] G. D. Finlayson, S. D. Hordley, C. Lu, and M. S. Drew, "On the removal of shadows from images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 59–68, Jan 2006.
- [19] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.
- [20] M. Paton, K. MacTavish, C. J. Ostafew, and T. D. Barfoot, "It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1519–1526.
- [21] K. MacTavish, M. Paton, and T. D. Barfoot, "Visual triage: A bag-of-words experience selector for long-term visual route following," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2065–2072.
- [22] M. Paton, F. Pomerleau, and T. D. Barfoot, "In the dead of winter: Challenging vision-based path following in extreme conditions," in *Field and Service Robotics*. Springer International Publishing, 2016, pp. 563–576.
- [23] G. Blanc, Y. Mezouar, and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005.
- [24] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "Large scale vision based navigation without an accurate global reconstruction," in *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR '07*, Minneapolis, Minnesota, 2007, pp. 1–8.
- [25] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics and Automation*, 2009.
- [26] T. Nguyen, G. K. I. Mann, R. G. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *Journal of Intelligent & Robotic Systems*, 2016.
- [27] N. Hawes *et al.*, "The strands project: Long-term autonomy in everyday environments," *IEEE Robotics and Automation Magazine*, 2016.
- [28] T. Krajník, J. Faigl, V. Vonásek *et al.*, "Simple, yet Stable Bearing-only Navigation," *Journal of Field Robotics*, 2010.
- [29] T. Krajník, S. Pedre, and L. Přeučil, "Monocular navigation for long-term autonomy," in *International Conference on Advanced Robotics (ICAR)*, Nov 2013, pp. 1–6.
- [30] F. Majer, L. Halodová, and T. Krajník, "Source codes: Bearing-only navigation." [Online]. Available: <https://github.com/gestom/stroll.bearnav>
- [31] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, vol. 88, pp. 127–141, 2017.
- [32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, 2008.
- [33] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *European conference on Computer vision*, 2010.
- [34] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Proceedings of the ICCV*, 2010.
- [35] L. Halodová and T. Krajník, "Exposure setting for visual navigation of mobile robots," in *Student Conference on Planning in Artificial Intelligence and Robotics (PAIR)*, 2017.
- [36] T. Krajník *et al.*, "A practical multirobot localization system," *Journal of Intelligent and Robotic Systems*, 2014.



KEY ARTICLE [12] - ROBOTICS AND AUTONOMOUS SYSTEMS
2017

©[2017] Elsevier. This manuscript version is made available under the CC-BY-NC-ND 4.0 license <http://creativecommons.org/licenses/by-nc-nd/4.0/>.



Contents lists available at ScienceDirect

Robotics and Autonomous Systems

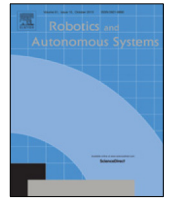
journal homepage: www.elsevier.com/locate/robot

Image features for visual teach-and-repeat navigation in changing environments



Tomáš Krajník^{a,*}, Pablo Cristóforis^b, Keerthy Kusumam^{a,d}, Peer Neubert^c, Tom Duckett^a

^a Lincoln Centre for Autonomous Systems, University of Lincoln, UK

^b Laboratory of Robotics and Embedded Systems, University of Buenos Aires, Argentina

^c Department of Electrical Engineering and Information Technology, Technische Universität Chemnitz, Germany

^d School of Computer Science, University of Nottingham, UK

HIGHLIGHTS

- We investigate long-term visual navigation of robots in outdoor environments.
- Robustness of image features to seasonal appearance variations is evaluated.
- The evaluation is based on five datasets gathered over the course of one year.
- A computationally efficient, trainable feature descriptor, called GRIEF, is proposed.
- Best performing image features are SpG/CNN and STAR/GRIEF.

ARTICLE INFO

Article history:

Available online 22 November 2016

Keywords:

Visual navigation
Mobile robotics
Long-term autonomy

ABSTRACT

We present an evaluation of standard image features in the context of long-term visual teach-and-repeat navigation of mobile robots, where the environment exhibits significant changes in appearance caused by seasonal weather variations and daily illumination changes. We argue that for long-term autonomous navigation, the viewpoint-, scale- and rotation- invariance of the standard feature extractors is less important than their robustness to the mid- and long-term environment appearance changes. Therefore, we focus our evaluation on the robustness of image registration to variable lighting and naturally-occurring seasonal changes. We combine detection and description components of different image extractors and evaluate their performance on five datasets collected by mobile vehicles in three different outdoor environments over the course of one year. Moreover, we propose a trainable feature descriptor based on a combination of evolutionary algorithms and Binary Robust Independent Elementary Features, which we call GRIEF (Generated BRIEF). In terms of robustness to seasonal changes, the most promising results were achieved by the SpG/CNN and the STAR/GRIEF feature, which was slightly less robust, but faster to calculate.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Cameras are becoming a de-facto standard in sensory equipment for mobile robotic systems including field robots. While being affordable, small and light, they can provide high resolution data in real time and virtually unlimited measurement ranges. Moreover, they are passive and do not pose any interference problems even when deployed in the same environment in large numbers. Most importantly, the computational requirements of

most machine vision techniques are no longer a significant issue due to the availability of powerful computational hardware. Hence, on-board cameras are often used as the primary sensors to gather information about the robot's surroundings.

Many visual robot navigation and visual SLAM methods rely on local image features [1] that allow to create quantitatively sparse, but information-rich image descriptions. These methods consist of a detection and a description step, which extract salient points from the captured images and describe the local neighborhood of the detected points. Local features are meant to be detected repeatedly in a sequence of images and matched using their descriptors, despite variations in the viewpoint or illumination. Regarding the quality of feature extractors, a key paper of Mikolajczyk and Schmid [2] introduced a methodology for evaluation of feature invariance to image scale, rotation, exposure and camera

* Corresponding author.

E-mail addresses: tkrajnik@lincoln.ac.uk (T. Krajník), pdecris@gmail.com (P. Cristóforis), kkusumam88@gmail.com (K. Kusumam), peer.neubert@etit.tu-chemnitz.de (P. Neubert), tduckett@lincoln.ac.uk (T. Duckett).

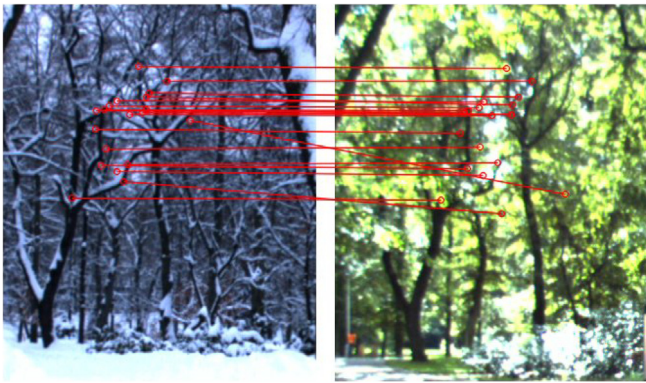


Fig. 1. Examples of tentative matches of the GRIEF image features across seasonal changes.

viewpoint changes. Mukherjee et al. [3] evaluated a wide range of image feature detectors and descriptors, confirming the superior performance of the SIFT algorithm [4]. Other comparisons were aimed at the quality of features for visual odometry [5] or visual Simultaneous Localization and Mapping (SLAM) [6]. Unlike the aforementioned works, we focus our evaluation on navigational aspects, especially to achieve long-term autonomy under seasonal changes.

Although the problem of long-term autonomy in changing environments has received considerable attention during the last few years [7], the main efforts were aimed at place recognition [8] and metric localization [9]. Unlike these works, we focus on the image processing aspect of long-term navigation in the context of teach-and-repeat systems [10], where a key issue is robust estimation of the robot heading [11,12].

Let us consider a scenario where a mobile robot navigates along a previously mapped path using vision as the main sensory modality. Typically, the robot would keep close to the previously learned path and it will not be necessary to use image features that are highly invariant to significant viewpoint changes. One can also assume that the surface in the path vicinity will be locally planar, which means that rotational invariance of the image features is not important either. On the other hand, the appearance of outdoor environments changes over time due to illumination variations, weather conditions and seasonal factors [13]. After some time, the environment appearance might differ significantly from its pre-recorded map, making long-term map-based visual navigation a difficult problem (see Fig. 1).

We hypothesize that for the purpose of teach-and-repeat visual navigation, the invariance of the image features to scale, rotation and viewpoint change is less important than their robustness to seasonal and illumination variations. These considerations motivate us to analyze available feature detector and descriptor algorithms in terms of their long-term performance in autonomous navigation based on a teach-and-repeat principle, e.g., as used in [10–12,14].

In this work, we present an image feature evaluation methodology which is tailored for teach-and-repeat navigation in long-term scenarios. We show the results achieved using combinations of open-source feature detectors and descriptors such as BRIEF [15], (root)-SIFT [4], ORB [16] and BRISK [17]. Moreover, we evaluate a feature based on a Convolutional Neural Network (CNN) descriptor and a Superpixel Grid detector (SpG) [18]. We also propose a trainable feature descriptor based on evolutionary methods and binary comparison tests and show that this algorithm, called GRIEF (Generated BRIEF), and the SpG/CNN feature outperform the engineered image feature extractors in their ability to deal with naturally-occurring seasonal changes and lighting variations [19].

This adaptive approach allows to automatically generate visual feature descriptors that are more robust to environment changes than standard hand-designed features.

The work presented here broadens our previously-published analysis [19] by including new datasets ('Nordland' [18]), image features (SpG/CNN) and feature training schemes. In particular, we separate the influence of the detector and descriptor phases on the robustness of the feature extractors to appearance changes and demonstrate that combination of detection and description phases of different features can result in feature extractors that are more robust to seasonal variations. Moreover, we perform a comparative analysis of training schemes, leading to computationally-efficient image features that can deal with naturally-occurring environment changes. We apply our evaluation on a new dataset, which became available only recently [20]. Finally, we provide the aforementioned benchmarking framework and the GRIEF training method as a documented, open-source software package [21].

2. Visual navigation in changing environments

The problem of vision-based localization and mapping has received considerable attention during the last decades and nowadays robots can create precise maps of very large environments and use these maps to determine their position with high accuracy. Localization itself was typically studied in the context of Simultaneous Localization and Mapping (SLAM), where the position estimate was based on a map that was built on-the-fly and, therefore, the effects of environment changes had only marginal importance. However, as the operation time of the robots increased, they have to face the problem that cameras are inherently passive and their perception of the environment is heavily influenced by illumination factors which tend to change throughout the day.

This issue motivated research into methods that are able to suppress the effects of naturally-changing outdoor illumination. One of the popular methods [22] calculates illumination-invariant images by exploiting the fact that the wavelength distribution of the main outdoor illuminant, the sun, is known. This method improves robot localization and navigation in outdoor environments [23–26], but can cope only with changes caused by varying outdoor illumination during the day. A recent work by Mount and Milford also reported that low-light cameras [27] can provide images that allow reliable day/night localization.

However, appearance changes are not caused just by varying illumination, but also by the fact that the environment itself changes over time. Valgren and Lilienthal [13] addressed the question of environment change in vision-based localization by studying the robustness of SIFT and SURF image features to seasonal variations. The paper indicated that as robots are gradually becoming able to operate for longer and longer time periods, their navigation systems will have to address the fact that environment itself, not only the illumination, is subject to constant, albeit typically slow, changes.

Some approaches aimed at solving the problem by using long-term observations to identify which environment features are more stable. Dayoub and Duckett [28] presented a method that continuously adapts the environment model by identifying stable image features and forgetting the unstable ones. Rosen et al. [29] used Bayesian-based survivability analysis to predict which features will still be visible after some time and which features will disappear. Carlevaris et al. [30] proposed to learn visual features that are robust to the appearance changes and showed that the learned features outperform the SIFT and SURF feature extractors. Lowry et al. [31] used principal component analysis to determine which aspects of a given location appearance are influenced by seasonal factors and presented a method that can calculate 'condition-invariant' images. Cieslewski et al. [32] show that a sparse 3D



Fig. 2. Examples of the seasonal variations at location II of the Planetarium dataset.



Fig. 3. View from the robot camera at three different locations of the Planetarium dataset.



Fig. 4. View from the robot camera at two locations of the Stromovka dataset.



Fig. 5. Examples of the seasonal variations at location II of the Michigan dataset.

environment description obtained through structure-from-motion approaches is robust to seasonal changes as well.

Some works use the long-term observations to build models that can predict the appearance of a given location at a particular time. Lowry et al. [33] applied linear regression techniques directly to the image space in order to predict the visual appearance of different locations in various conditions. Sünderhauf and Neubert [34,35] mined a dictionary of superpixel-based visual-terms from long-term data and used this dictionary to translate between the appearance of given locations across seasons. Krajník et al. [36] used Fourier analysis to identify the cyclical changes of the environment states and showed that predicting these states for a particular time improves long-term localization [37].

Another group of approaches proposes to use multiple, condition-dependent representations of the environment. For example, Churchill and Newman [9] clustered different observations of the same place to form “experiences” that characterize the place appearance in particular conditions. McManus et al. [38] used dead reckoning to predict which place the vehicle is close to, loaded a bank of Support Vector Machine classifiers associated with that place and used these to obtain a metric pose estimate. Krajník et al. [39] proposed to maintain maps gathered over an entire year and select the most relevant map based on its mutual information with the current observation.

Methods based on deep learning, which has had a big impact on the field of computer vision, were also applied to the problem of persistent navigation. Neubert and Protzel [18] showed that image descriptors based on Convolutional Neural Networks

(CNN) outperformed the best holistic place recognition methods while being able to handle large viewpoint changes. Sünderhauf et al. [8,40] also demonstrated impressive results with CNN-based methods. However, the recent outcome of the Visual Place Recognition in Changing Environments, or VPRiCE Challenge [41] indicated that novel, yet classic-feature-based approaches, such as [42] performed better than the CNN-based methods.

Most of the aforementioned approaches were aimed at place recognition [7] and metric localization [9]. Unlike these works, we focus on the image processing aspect of long-term navigation in the context of teach-and-repeat systems [10], where a key issue is robust estimation of the robot heading [11,12].

3. Local image feature extractors

Local image features provide a sparse, but distinctive representation of images so that these can be retrieved, matched or registered efficiently. The feature extraction process consists of two successive phases: feature detection and feature description. The detector identifies a salient area in an image, e.g. a corner, blob or edge, which is treated as a keypoint. The descriptor creates a vector that characterizes the neighborhood of the detected keypoint, typically in a scale-affine invariant way. Typical descriptors capture various properties of the image region like texture, edges, intensity gradients, etc.

The features are meant to be repeatably extracted from different images of the same scene even under conditions of unstable illumination or changing viewpoints. In this paper, we evaluate several image feature extraction algorithms for the purpose of long-term robot navigation. Most of these algorithms are included in the Open Source Computer Vision (OpenCV) software library (version 2.4.3), which was used to generate the results presented in this paper.

3.1. Feature detectors

3.1.1. LoG/DoG (SIFT)

The SIFT feature [4] uses a Difference-of-Gaussians detector to find scale-invariant keypoint locations. The feature detection process first generates a scale space of the image by convolving it with Gaussian kernels of different sizes. The DoG detector then searches for local extrema in the images obtained by the difference of two adjacent scales in the Gaussian image pyramid. This gives an approximation of the Laplacian of Gaussian (LoG) function where local extrema correspond to the locations of blob-like structures. A local extremum is found by comparing the DoG values of each point with its 8 pixel neighborhood and 9 other neighbors in the two adjacent scale levels. This type of keypoint localization allows to detect blobs at multiple scales, resulting in scale invariance of the features. To achieve rotation invariance, SIFT assigns a dominant orientation to the detected keypoint obtained by binning the gradient orientations of its neighborhood pixels.

3.1.2. Hessian–Laplace region (SURF)

The Hessian keypoint detector finds interest points that vary in the two orthogonal directions [43]. It computes the second derivatives for each image location and finds the points for which the determinant of the Hessian matrix is maximal. The Hessian–Laplace detector combines the Hessian detector that returns corner-like structures along with a LoG detector. The Hessian detector returns interest points at each scale in the scale space and the Laplacian of Gaussian (LoG) detector searches for the extremum on these interest locations. The SURF detection scheme speeds up the process by approximating the Gaussian scale pyramid using box filters.

3.1.3. Maximally Stable Extremal Regions—MSER

The MSER method finds regions that remain invariant under varying conditions of image transformations [44]. The algorithm applies a watershed segmentation algorithm with a large number of thresholds and finds the regions that remain stable across these thresholds. These regions are affine-covariant and can be reliably extracted from an image irrespective of large viewpoint or affine transformations. Since segmentation is used, the regions can have different contours or an elliptical contour can be fitted to the region.

3.1.4. Features from Accelerated Segment Test—FAST

The FAST detector compares intensities of pixels lying on a 7-pixel diameter circle to the brightness of the circle's central pixel [45]. The 16 pixels of the circle are first marked as bright, neutral or dark depending on their brightness relative to the central pixel. The central pixel is considered as a keypoint if the circle contains a contiguous sequence of at least n bright or dark pixels (a typical value of n is 12). In order to quickly reject candidate edges, the detector uses an iterative scheme to sample the circle's pixels. For example, the first two examined pixels are the top and bottom one—if they do not have the same brightness, a contiguous sequence of 12 pixels cannot exist and the candidate edge is rejected. This fast rejection scheme causes the FAST detector to be computationally efficient.

3.1.5. Oriented FAST and Rotated BRIEF—ORB

The ORB feature extractor combines a FAST detector with an orientation component (called oFAST) [16]. The keypoints are identified by the FAST detector and ordered by the Harris corner measure, then the best N keypoints are chosen. The original FAST detector is not scale invariant, hence the ORB detector uses a scale space to identify interest points. Then, the orientation of the feature is calculated using the intensity centroid. The direction of the vector between the intensity centroid and the corner's center gives the orientation of the point.

3.1.6. Binary robust invariant scalable keypoints—BRISK

The BRISK feature detector is scale and rotation invariant [17]. To identify the keypoint locations, BRISK uses the AGAST [46] feature detector, which is an accelerated variant of FAST. The scale invariance of BRISK is achieved by detecting keypoints on a scale pyramid [17]. The points are chosen by ordering them according to the FAST scores for saliency.

3.1.7. Center surround extremas—STAR

The STAR feature detector is a variant of the Center Surround Extrema (CenSurE) detector [47]. The authors of CenSurE argue that the keypoint localization precision of the multi-scale detectors like SIFT and SURF becomes low because of the interpolation used at higher levels of the scale space. The CenSurE detector circumvents this issue as it searches for keypoints as extrema of the center surround filters at multiple scales. Thus, the scale space is generated by using masks of different sizes rather than interpolation, which has a negative impact on detection precision. While CenSurE uses polygons to approximate the circular filter mask, the STAR feature approximates it by using two square masks (one upright and one rotated at 45 degrees). Similarly to SURF, this scheme allows for efficient box filter response calculation at multiple scales, resulting in the computational efficiency of STAR.

3.1.8. Superpixel-Grids—SpG

The above detectors are designed to extract a sparse set of salient image locations from the image. In contrast, the recently published Superpixel-Grid detector (SpG) [48] provides a dense set of local regions based on superpixel segmentations. A superpixel segmentation is an oversegmentation of an image. To obtain SpG regions, the image is segmented at multiple scales and neighboring segments are combined to create a set of overlapping regions. These SpG regions are better adapted to the image content than fixed patches and were successfully used in combination with ConvNet descriptors for place recognition in changing environments. Since there is only a tentative Matlab implementation available [48], we include only a partial evaluation in the experiments section, where we extract around 100, 240 or 740 regions per image.

3.2. Feature descriptors

3.2.1. Scale Invariant Feature Transform—SIFT

The Scale Invariant Feature Transform (SIFT) is probably the most popular local feature extractor [4] due to its scale and rotation invariance and robustness to lighting and viewpoint variations. The SIFT descriptor is based on gradient orientation histograms. It is formed by sampling the image gradient magnitudes and orientations of the region around the keypoint while taking into account the scale and rotation calculated in the previous steps. The interest region is sampled around a keypoint, at a given scale, at 16×16 pixels. This region is divided into 4×4 grid of pixels and the gradient orientations and magnitudes are calculated. Each grid is accumulated into an 8-bin histogram of gradient orientations, which is weighted by the gradient magnitude of given pixel. It results in a high-dimensional vector of size 128, which contributes to the distinctiveness of the descriptor. Further steps include normalization of the resulting feature vector and clipping of the feature values to 0.2. This provides robustness against illumination variations. While being precise, distinctive and repeatable, calculation of the SIFT feature extractor is computationally demanding. Arandjelović and Zisserman [49] showed that simple normalization (called Root-SIFT) improves SIFT performance in object retrieval scenarios.

3.2.2. Speeded Up Robust Features—SURF

Inspired by SIFT, the Speeded Up Robust Feature (SURF) extractor was first introduced by Bay et al. [50]. The main advantage of SURF is its speed—the experiments presented in [50] show that it is significantly faster than SIFT, with no considerable performance drop in terms of invariance to viewpoint, rotation and scale changes. The speedup is achieved through the use of integral images that allow to calculate the response of arbitrarily-sized 2D box filters in constant time. The box filters are used both in the detection step and the description phase for spatial binning, similarly to SIFT. The (rather inefficient) rotation estimation step can be omitted from the SURF algorithm, resulting in 'Upright SURF', which is not rotation invariant. This might be beneficial in some applications, for example, Valgren and Lilienthal [13] showed that U-SURF outperforms SURF in long-term outdoor localization.

3.2.3. Binary robust independent elementary features—BRIEF

The BRIEF feature descriptor uses binary strings as features, which makes its construction, matching and storage highly efficient [15]. The binary string is computed by using pairwise comparisons between pixel intensities in an image patch that is first smoothed by a Gaussian kernel to suppress noise. In particular, the value of the i th bit in the string is set to 1 if the intensity value of a pixel in position x_i, y_i is greater than the intensity of a pixel at position x'_i, y'_i . Since the sequence of test locations of the comparisons $\delta_i = (x_i, y_i, x'_i, y'_i)$ can be chosen arbitrarily, Calonder et al. [15] compared several schemes for generating δ_i and determined the

best distribution to draw δ_i from. The binary strings are matched using Hamming distance, which is faster than using the Euclidean distance as in SIFT or SURF. In [15], the authors consider binary string sizes of 128, 256 and 512 referred to as BRIEF-16, BRIEF-32, BRIEF-64 respectively.

3.2.4. Oriented FAST and Rotated BRIEF—ORB

The ORB feature extractor combines the FAST detector with orientation component (called oFAST) and the steered BRIEF (rBRIEF) descriptor [16]. The goal of ORB is to obtain robust, fast and rotation-invariant image features meant for object recognition and structure-from-motion applications. ORB uses a rotated/steered variant of BRIEF features where the coordinates of the pair of points for comparison are rotated according to the orientation computed for each keypoint. The comparisons are then performed. However, the rotation invariance introduced in ORB has a negative impact on its distinctiveness. Thus, the authors of ORB employed machine learning techniques to generate the comparison points so that the variance of the comparisons are maximized and their correlation minimized.

3.2.5. Binary robust invariant scalable keypoints—BRISK

The descriptor of BRISK is a binary string that is based on binary point-wise brightness comparisons similar to BRIEF [17]. Unlike BRIEF or ORB, which use a random or learned comparison pattern, BRISK's comparison pattern is centrally symmetric. The sample points are distributed over concentric circles surrounding the feature point and Gaussian smoothing with a standard deviation proportional to the distance between the points is applied. While the outermost points of the comparison pattern are used to determine the feature orientation, the comparisons of the inner points form the BRISK binary descriptor. The orientation is computed using the local gradients between the long distance pairs and the short distance comparisons are rotated based on this orientation. The BRISK descriptor is formed by taking the binary comparisons of the rotated short distance pairs with a feature length of 512.

3.2.6. Fast Retina Keypoint—FREAK

FREAK is a binary descriptor similar to BRIEF, BRISK and ORB, which uses a sampling pattern inspired by the human retina [51]. FREAK also uses a circular pattern for sampling points, although the density of the points is higher towards the center of the pattern, similar to the human retina. It uses different Gaussian kernels that overlap for smoothing the points following the distribution of the receptive fields in the retina. FREAK uses a coarse-to-fine approach for the comparisons to form the final binary string descriptor.

3.2.7. Convolutional Neural Networks—CNN

In recent years, Deep Learning methods were successfully applied to many computer vision tasks. This inspired the application of descriptors computed from the output of general purpose Convolutional Neural Networks (CNN) for place recognition in changing environments [8,18,40]. CNNs are a class of feed-forward artificial (neural) networks whose lower convolutional layers were shown to be robust against environmental changes like different seasons, illumination, or weather conditions. In our experiments we follow [18] and use the conv3-layer of the VGG-M network [52]. Due to the high computational efforts for computing the CNN descriptor, we evaluated its CPU and GPU implementations.

4. GRIEF: Generated BRIEF sequence

The standard BRIEF descriptor is a binary string that is calculated by 256 intensity comparisons of pixels in a 48×48 image region surrounding the keypoint provided by a detector. In principle, the locations of the pixel pairs to be compared can be

chosen arbitrarily, but have to remain static after this choice has been made. Realizing that the choice of the comparison locations determines the descriptor performance, the authors of BRIEF and ORB attempted to find the best comparison sequences. While the authors of the original BRIEF algorithm proposed to select the sequences randomly from a two-dimensional Gaussian distribution, the authors of ORB chose the locations so that the variance of the comparisons is high, but their correlation is low.

We propose a simple method that allows to adapt the BRIEF comparison sequence for a given dataset. The proposed method exploits the fact that the similarity of the BRIEF features are calculated by means of Hamming distance of the binary descriptors and, therefore, the contribution of each comparison pair to the descriptor distinctiveness can be evaluated separately. This allows to rate the individual comparison locations that constitute the BRIEF descriptor.

Given an image \mathbf{I} , a BRIEF descriptor $\mathbf{b}(\mathbf{I}, c_x, c_y)$ of an interest point c_x, c_y (detected by the STAR algorithm) is a vector consisting of 256 binary numbers $b_i(\mathbf{I}, c_x, c_y)$ calculated as

$$b_i(\mathbf{I}, c_x, c_y) = \mathbf{I}(x_i + c_x, y_i + c_y) > \mathbf{I}_j(x'_i + c_x, y'_i + c_y). \quad (1)$$

Since the position c_x, c_y is provided by the feature detector, the BRIEF descriptor calculation is defined by a sequence Δ of 256 vectors $\delta_i = (x_i, y_i, x'_i, y'_i)$ that define pixel positions for the individual comparisons. Thus, the BRIEF method calculates the dissimilarity of interest point \mathbf{a} with coordinates (a_x, a_y) in image \mathbf{I}_a and interest point \mathbf{b} with coordinates (b_x, b_y) in image \mathbf{I}_b by the Hamming distance of their binary descriptor vectors $\mathbf{b}(\mathbf{I}_a, a_x, a_y)$ and $\mathbf{b}(\mathbf{I}_b, b_x, b_y)$. Formally, the dissimilarity $d(\mathbf{a}, \mathbf{b})$ between points \mathbf{a} and \mathbf{b} is

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{255} d_i(\mathbf{a}, \mathbf{b}), \quad (2)$$

where $d_i(\mathbf{a}, \mathbf{b})$ are the differences of the individual comparisons δ_i calculated as

$$d_i(\mathbf{a}, \mathbf{b}) = |b_i(\mathbf{I}_a, a_x, a_y) - b_i(\mathbf{I}_b, b_x, b_y)|. \quad (3)$$

Let us assume that the BRIEF method has been used to establish tentative correspondences of points in two images, producing a set \mathcal{P} of point pairs $\mathbf{p}_k = (\mathbf{a}_k, \mathbf{b}_k)$. Now, let us assume that the tentative correspondences were marked as either 'correct' or 'false', e.g. by RANSAC-based geometrical verification [53], or by histogram voting scheme [11]. This allows to split \mathcal{P} into a set of correct correspondence pairs \mathcal{P}_C and a set of incorrectly established pairs \mathcal{P}_F . This allows to calculate the fitness $f(\delta_i, \mathcal{P}_C, \mathcal{P}_F)$ of each individual comparison δ_i as

$$f(\delta_i, \mathcal{P}_C, \mathcal{P}_F) = \sum_{\mathbf{p} \in \mathcal{P}_C} (1 - 2 d_i(\mathbf{p})) + \sum_{\mathbf{p} \in \mathcal{P}_F} (2 d_i(\mathbf{p}) - 1). \quad (4)$$

The first term of Eq. (4) penalizes the comparisons δ_i that increase the Hamming distance of correctly established correspondences and increases the fitness of comparisons that do not contribute to the Hamming distance. The second term of Eq. (4) improves the fitness of comparisons that indicate the differences of incorrectly established correspondences, while penalizing those comparisons that do not increase the Hamming distance. The fitness function $f(\delta_i)$ allows to rank the comparisons according to their contribution to the descriptor's distinctiveness.

The sets \mathcal{P}_C and \mathcal{P}_F , which serve as positive and negative training samples, can contain correspondences from several image pairs, which allows to calculate the fitness $f(\delta_i)$ for larger datasets. The fitness evaluation of the individual components (comparisons) of the descriptor allows to train GRIEF for a given dataset through an iterative procedure that repeatedly evaluates the contribution

of the individual comparisons δ_i to the feature's distinctiveness and substitutes the 'weak' comparisons by random vectors, see Algorithm 1.

At first, the training method extracts positions of the interest points of all training images, calculates the descriptors of these keypoints using the latest comparison sequence Δ and establishes tentative correspondences between the features of relevant image pairs. Then, a histogram of horizontal (in pixels) distances of the corresponding points is built for each image pair from the same location. The highest bin of this histogram contains correspondences consistent with the relative rotation of the robot when capturing the two images –these correspondences are added to the set \mathcal{P}_C , while the rest of the tentative correspondences are added to set \mathcal{P}_F . After that, Eq. (4) is used to rank the individual pixel-wise comparisons δ_i . Then, the algorithm discards the 10 comparisons with the lowest fitness and generates new ones by drawing (x_i, y_i, x'_i, y'_i) from a uniform distribution. The aforementioned procedure is repeated several (n_g) times. The resulting comparison sequence Δ is better tuned for the given dataset. Except for the locations of pixels to be compared, the working principle of the GRIEF feature is identical to BRIEF and the time required for computation and matching is the same.

Algorithm 1: GRIEF comparison sequence training

Input: \mathcal{I} – a set of images for GRIEF training,
 Δ_0 – initial comparison sequence – BRIEF
 n_g – number of iterations

Output: Δ – improved compar. sequence – GRIEF

```

// calculate keypoints in all images
foreach  $\mathbf{I} \in \mathcal{I}$  do
   $\mathcal{C}_1 \leftarrow \text{STAR}(\mathbf{I})$ 
// start GRIEF training
while  $n < n_g$  do
  // extract GRIEF features
  foreach  $\mathbf{I} \in \mathcal{I}$  do
     $\mathcal{B}_1 \leftarrow \emptyset$  // clear descriptor set
    foreach  $(c_x, c_y) \in \mathcal{C}_1$  do
       $\mathcal{B}_1 \leftarrow \{\mathcal{B}_1 \cup \text{GRIEF}(\mathbf{I}, c_x, c_y)\}$ 
    // generate training samples
     $\mathcal{P}_C, \mathcal{P}_F \leftarrow \emptyset$  // initialize sample sets
  foreach  $\mathbf{I}, \mathbf{J} \in \mathcal{I}$  do
    // calculate correspondences
    if  $\mathbf{I} \neq \mathbf{J}$  then
      // tentative correspondences
       $\mathcal{P} \leftarrow \text{match}\{\mathcal{B}_1, \mathcal{B}_j\}$ 
      // geometric constraints
       $(\mathcal{P}'_C, \mathcal{P}'_F) \leftarrow \text{histogram voting}(\mathcal{P})$ 
      // add results to sample sets
       $\mathcal{P}_C \leftarrow \{\mathcal{P}_C \cup \mathcal{P}'_C\}$ 
       $\mathcal{P}_F \leftarrow \{\mathcal{P}_F \cup \mathcal{P}'_F\}$ 
    // establish fitness of  $\delta_i$  by (4)
  for  $i \in 0..255$  do
     $f(\delta_i) \leftarrow \sum_{\mathcal{P}_C} (1 - 2 d_i(\cdot)) + \sum_{\mathcal{P}_F} (2 d_i(\cdot) - 1)$ 
  // increment iteration number
   $n \leftarrow n + 1$ 
  // replace 10 least-fit comparisons
  for  $i \in 0..9$  do
     $\delta_w \leftarrow \arg \min_{\delta \in \Delta} (f(\delta))$  // least fit  $\delta$ 
     $\Delta \leftarrow \{\Delta \setminus \delta_w\}$  // gets replaced
     $\Delta \leftarrow \{\Delta \cup \text{random } \delta_i\}$  // by a random  $\delta$ 

```

5. Evaluation datasets

The feature evaluation was performed on five different datasets collected by mobile vehicles over the course of several months. The Planetarium dataset was gathered on a monthly basis in a small forest area near Prague's planetarium in the Czech Republic during the years of 2009 and 2010 [11]. The Stromovka dataset comprises of 1000 images captured during two 1.3 km long tele-operated runs in the Stromovka forest park in Prague during summer and winter 2011 [54]. The third and fourth datasets, called 'Michigan' and 'North Campus', were gathered around the University of Michigan North Campus during 2012 and 2013 [20]. Similarly to the datasets gathered in Prague, the Michigan set covers seasonal changes in a few locations over one year and the North Campus dataset consists of two challenging image sequences captured in winter and summer. The fifth dataset, called 'Nordland', consists of more than 1000 images organized in two sequences gathered during winter and summer on a ~ 20 km long train ride in northern Norway [18]. The datasets that we used for our evaluation are publicly available at [21].

5.1. The planetarium dataset

The Planetarium dataset was obtained by a P3-AT mobile robot with a Unibrain Fire-i601c color camera. At first, the mobile robot was manually driven through a 50 m long path and created a topological-landmark map, where each topological edge was associated with a local map consisting of image features. On the following month, the robot used a robust navigation technique [11] to repeat the same path using the map from the previous month. During each autonomous run, the robot recorded images from its on-board camera and created a new map. Data collection was repeated every month from September 2009 until the end of 2010, resulting in 16 different image sequences [54].

Although the path started at an identical location every time, the imprecision of the autonomous navigation system caused slight variations in the robot position when traversing the path. Therefore, the first image of each traversed path is taken from exactly the same position, while the positions of the other pictures may vary by up to ± 0.8 m.

Although the original data contains thousands of images, we have selected imagery only from 5 different locations in 12 different months, see Figs. 2 and 3.

Six independent persons were asked to register the images and to establish their relative horizontal displacement, which corresponds to the relative robot orientation at the times the images were taken. The resulting displacements were checked for outliers (these were removed) and the averaged estimations were used as ground truth.

5.2. The stromovka dataset

The Stromovka dataset was gathered by the same robot as the Planetarium dataset. It consists of four image sequences captured in different seasons along a 1.3 km long path through diverse terrain of the Stromovka park in Prague. The appearance of the environment between the two sequences changes significantly (see Fig. 4), which makes the Stromovka dataset especially challenging. The magnitude of the appearance change should allow for better evaluation of the feature extractors' robustness to environment variations. Unlike the Planetarium dataset, where the robot used a precise navigation technique, the Stromovka data collection was tele-operated and the recorded trajectories are sometimes more than 2 m apart. The Stromovka dataset exhibits not only seasonal variations, but also permanent changes, e.g. some trees were cut down, see Fig. 4.

5.3. The Michigan dataset

The Michigan Dataset was collected by a research team at the University of Michigan for their work on image features for dynamic lighting conditions [30]. The dataset was gathered during 27 data-collection sessions performed over 15 months around the North University Campus in Ann Arbor, comprising 1232×1616 color images captured from 5 different locations.

Since this dataset was not captured on an exactly regular basis and some months were missing, we selected 12 images of each place in a way that would favor their uniform distribution throughout a year. Then, we removed the uppermost and bottom parts of the images that contain ground plane or sky and resized the rest to 1024×386 pixels while maintaining the same aspect ratio, see Figs. 5 and 6. The resulting dataset has the same format as the Planetarium one and was evaluated in exactly the same way.

However, the Michigan dataset was gathered around a university campus and it contains less foliage and more buildings than the Planetarium and Stromovka datasets. Moreover, seasonal weather variations in Ann Arbor are less extreme than the ones in Prague. Therefore, the appearance of the environment captured in the Michigan dataset is less influenced by the naturally occurring seasonal changes.

5.4. The North Campus dataset

The team of the Michigan university carried on with their data collection efforts and made their 'North Campus Long-Term Dataset' publicly available [20]. This large-scale, long-term dataset consists of omnidirectional imagery, 3D lidar, planar lidar, and proprioceptive sensory data and ground truth poses, which makes it a very useful dataset for research regarding long-term autonomous navigation. The dataset's 27 sessions, which are spread over 15 months, capture the university campus, both indoors and outdoors, on varying trajectories, and at different times of the day across all four seasons. We selected two outdoor sequences captured by the robot's front camera during February and August 2012 and processed them in exactly the same way as the images from the Michigan dataset. Thus, we obtained two challenging image sequences in a format similar to the Stromovka dataset, see Fig. 7.

5.5. The Nordland dataset

Similarly to the North Campus and Stromovka, the 'Nordland' dataset consists of two challenging sequences captured during winter and summer. However, this dataset was not gathered by a mobile robot, but by a train-mounted camera that recorded the spectacular landscape between Trondheim and Bodø in four different seasons. Since the original footage contains four ten-hour videos with more than 3 million images captured from the same viewpoint and angle, we had to adapt the dataset for our purposes. First, we selected 1000 images covering 20 km of the train ride in winter and summer. To emulate camera viewpoint variation, we shifted and cropped the winter images, so that the winter/summer image pairs would overlap only by $\sim 85\%$, see Fig. 8. Unlike in [18], where the images are shifted by a fixed number of pixels, we used a variable shift in both horizontal and vertical directions.

6. Evaluation

The goal of our evaluation is to test the suitability of various image features for long-term visual teach-and-repeat in changing environments. Our evaluation assumes that the robot's navigation is based on a teach-and-repeat method that uses the visual data to correct the robot's orientation in order to keep it on the path it has been taught previously [10–12,14]. Since these methods



Fig. 6. View from the robot camera at different locations of the Michigan dataset.



Fig. 7. View from the robot camera at two locations of the North Campus dataset.



Fig. 8. Example images from the Nordland dataset. Notice the horizontal shift between the winter/summer image pairs.

do not require full six degree-of-freedom global localization, we evaluate the feature extraction and matching algorithms in terms of their ability to establish the correct orientation of the robot under environment and lighting variations. Since the proposed evaluation is based on a measure of the feature extractor's ability to establish the robot heading, we calculate its 'error rate' as the ratio of incorrect to total heading estimates. In our evaluation, we select image pairs from the same locations but different times, extract and match their features and estimate the (relative) robot orientation from the established correspondences. We consider an orientation estimate as correct if it does not differ from the ground truth by more than 35 pixels, which roughly corresponds to 1 degree.

To determine the best features for the considered scenario, we evaluate not only their invariance to seasonal changes, but also

their computational complexity. Moreover, our evaluation also requires to select the other components of the processing pipeline, which estimates the robot heading based on the input images. In particular, we need to choose how to match the currently perceived features to the mapped ones, how to determine the robot orientation based on these matches and what training scheme to use for the GRIEF feature.

6.1. Feature matching schemes

To determine the best strategy for feature matching, we compared the performance of two different matching schemes, which attempt to establish pairs between the feature sets \mathcal{A} and \mathcal{B} extracted from the two images. The first scheme, called a 'ratio test', searches the descriptor space for two nearest neighbors $\mathbf{b}_0, \mathbf{b}_1 \in \mathcal{B}$

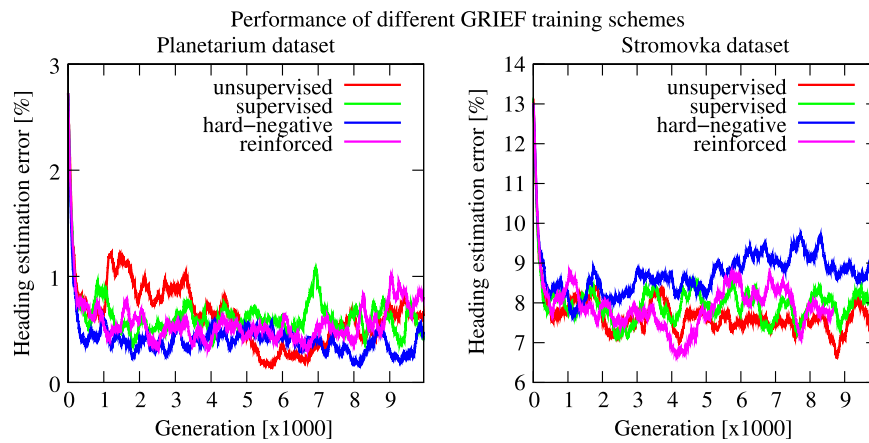


Fig. 9. The performance of different training schemes of GRIEF: Evolution of position estimation errors on the Stromovka and Planetarium datasets (smoothed).

of a given feature $\mathbf{a}_0 \in \mathcal{A}$. A match is considered correct if $|\mathbf{a}_0 - \mathbf{b}_0| < r |\mathbf{a} - \mathbf{b}_1|$, where r is typically chosen between 0.5 and 0.9 [4]. The second scheme, called a 'symmetric match', considers \mathbf{a}_0 and \mathbf{b}_0 a pair if \mathbf{b}_0 is the nearest neighbor of \mathbf{a}_0 in the set \mathcal{B} and vice versa [55]. In our experiments, we evaluated the performance of the 'ratio test' matching with the r coefficient set to 10 different values between 0.5 and 1.0. However, the 'symmetric match' performed better and thus, the following results presented use the 'symmetric' matching strategy.

6.2. Heading estimation

We also considered two different methods for determining the relative rotation of the camera. The first method closely follows the classical approach used in computer vision where known camera parameters and correspondences between extracted and mapped features are used to calculate the essential matrix, which is factored to obtain the robot rotation. An alternative method used in [12,11] calculates a histogram of horizontal (in image coordinates) distances of the tentative correspondences and calculates the robot orientation from the highest-counted bin. In other words, the robot orientation is established from the mode of horizontal distances of the corresponding pairs by means of histogram voting. The latter method is less general, because it cannot cope with large viewpoint changes, but was reported to perform better than the essential-matrix-based method in teach-and-repeat scenarios [56]. Our observations confirm the findings presented in [56], and thus we chose to use the histogram voting method in our evaluations.

We hypothesize that better performance of the histogram voting method is caused by the fact that unlike the essential-matrix-based estimation, it does not assume rigid scenes. Thus, it is more robust to object deformations caused by snow, temperature variations or vegetation growth.

6.3. GRIEF feature training

Before the actual evaluations, we tested four different training schemes for the GRIEF feature. We evaluated how much a GRIEF feature trained on a specific location improves its performance across locations in different environments and how many iterations of the training Algorithm 1 are required. Four training schemes were considered:

Unsupervised, where the matched pairs are divided into positive \mathcal{P}_C and negative \mathcal{P}_F training samples (see Algorithm 1) by histogram voting, i.e. the pairs that belong in the highest-rated bin constitute the set \mathcal{P}_C and the others go to \mathcal{P}_F .

Supervised, where the division into \mathcal{P}_C and \mathcal{P}_F is based on the ground-truth provided with the dataset.

Hard-negative, which performs the GRIEF training only on the image pairs that were registered incorrectly, i.e. the results of the histogram voting method do not match the ground truth.

Reinforced, where the incorrectly-matched image pairs influence the evaluation of the individual comparisons $10\times$ more strongly than correctly-registered image pairs.

The advantage of the first training scheme is that it only needs to know which images were taken at the same locations, while the latter three schemes require the dataset to be ground-truthed. We performed 10 000 iterations of each training scheme on the Planetarium dataset and evaluated the performance of each generation on the Stromovka datasets. The results shown in Fig. 9 indicate that at first, the 'supervised' and 'hard-negative' training schemes outperform the 'unsupervised' one on the training dataset, but the situation is reversed when the trained feature is tested on images from another environment. Moreover, we can see that although the heading estimation error rate decreases quickly during the first ~ 500 training iterations, further training improves the feature performance at a slower rate.

We trained the GRIEF feature by running 10 000 iterations of the 'unsupervised' training scheme on the Planetarium dataset and validating its performance on 50 images of the Stromovka dataset. Based on this validation we selected the 8612th GRIEF generation for the rest of our experiments. The evolution of the GRIEF fitness and its performance improvement (i.e. heading estimation error relative to the BRIEF feature) are shown in Fig. 10. One iteration of the training algorithm on the Planetarium dataset takes approximately 10 s on an i7 machine. Thus, training the GRIEF sequence by iterating the algorithm 10 000 times took approximately one day.

6.4. Evaluation procedure

First, the feature correspondences between each pair of images from the same location were established by the 'symmetric match' scheme. Then, the corresponding feature pairs with significantly different vertical image coordinates were removed. After that, we build a histogram of horizontal distances of the corresponding pairs and find the most prominent bin. The average distance of all pairs that belong to this bin are used as an estimate of the relative orientations of the robot at the time instants when the particular images were captured. These estimates are then compared with the ground truth and the overall error is calculated as the ratio of

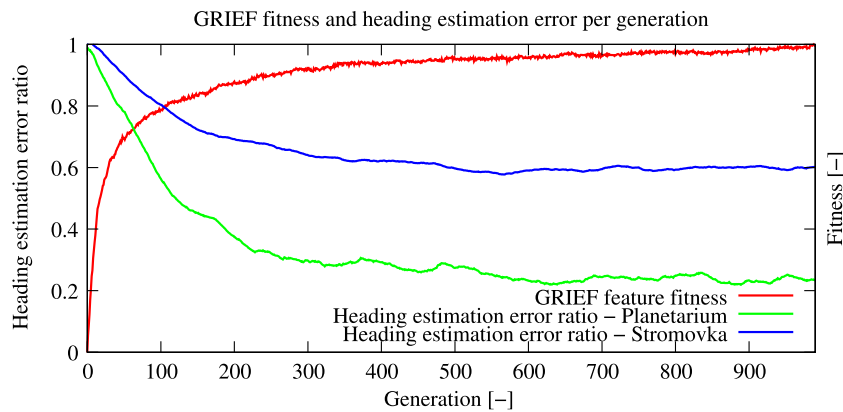


Fig. 10. GRIEF training process: GRIEF fitness and position estimation error improvement on the Stromovka and Planetarium dataset. The error is calculated relative to the heading estimation error of the BRIEF feature that is used to initialize the positions of the binary comparisons of the GRIEF. Error rates are smoothed by sliding average.

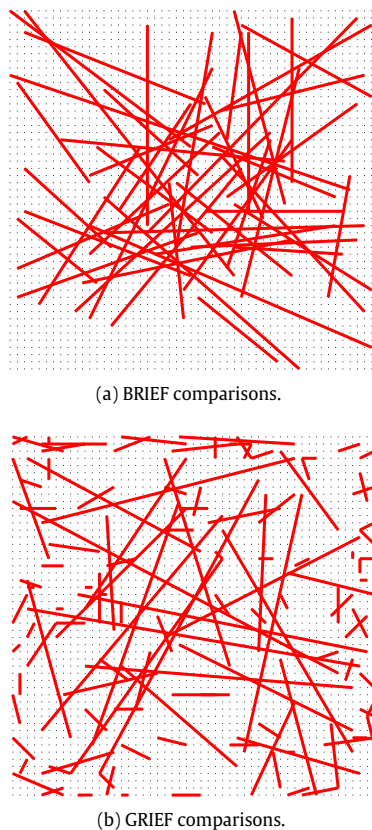


Fig. 11. A sample of the initial (BRIEF) and trained (GRIEF) comparison pairs. The GRIEF comparisons favor shorter distances.

incorrect heading estimations to the total number of image pairs compared (see Fig. 11).

The Michigan and Planetarium datasets contain 5 different locations with 12 images per location, which means that there are $12 \times 11 \times 5/2 = 330$ image pairs. The evaluation of the Stromovka dataset is based on 1000 (winter/summer) images arranged in two image sequences that cover a path of approximately 1.3 km, which means that the dataset contains 500 image pairs. The number of images in the North Campus and Nordland datasets is only slightly higher than in the Stromovka one, but their structure is the same, i.e. two long image sequences from winter and summer.

6.5. Number of features

The error rate for estimating the correct heading is dependent on the number of extracted features, which depends on the setting of the 'peak threshold' of a particular feature detector. Our benchmarking software allows to select the detector peak thresholds in such a way that the detection method extracts a given number of features per image. To show the dependence of the heading estimation error on the number of features extracted, we evaluated the performance of the most popular image features set to extract $\{100, 200, \dots, 1600\}$ features per dataset image. The results shown in Fig. 12 demonstrate how the number of extracted features influences the ability of the method to correctly estimate the robot heading. Fig. 12 also indicates that in some cases, it is not possible to reach a desired number of detected features (see the dashed lines). This is because the STAR detector does not extract enough features even if its peak threshold is set to the minimal value and the SpG detector was evaluated in three settings with 100, 220 and 740 features. The figure indicates that the lowest heading estimation error rates were achieved using the STAR/GRIEF and SpG/CNN image features.

Fig. 12 also shows that the performance of the features varies more for the North Campus and Stromovka datasets. This is caused by the fact that these datasets do not match images gathered on a monthly basis, but only from two opposite seasons, where the appearance changes are more prominent and the images are more difficult to register. To confirm this hypothesis, we divided the images of the Planetarium and Michigan datasets into two groups: 'winter' images, where the trees lack foliage and 'summer' images, where tree foliage is present. Then, we calculated the inter- and intra-season registration error rates of the upright-root-SIFT and STAR/GRIEF features. When matching images from the same season, both upright-root-SIFT and STAR-GRIEF methods achieved error rates below 3%. However, matching images across seasons by upright-root-SIFT resulted in approximately 24% error, while the STAR-GRIEF error rate was around 2%. This indicates that the error rate improvement is caused by ability of the STAR-GRIEF to register images with large perceptual changes.

6.6. Combining different detectors and descriptors

The performance of the image features is influenced by both the detector and descriptor phases. Although in some cases, the detection and description algorithms share the same data structures, which allows to speed up the feature's calculation (such as the integral image in SURF), there is no reason why the detection and description phases of different algorithms could not be combined in order to obtain features with desired properties. For example,

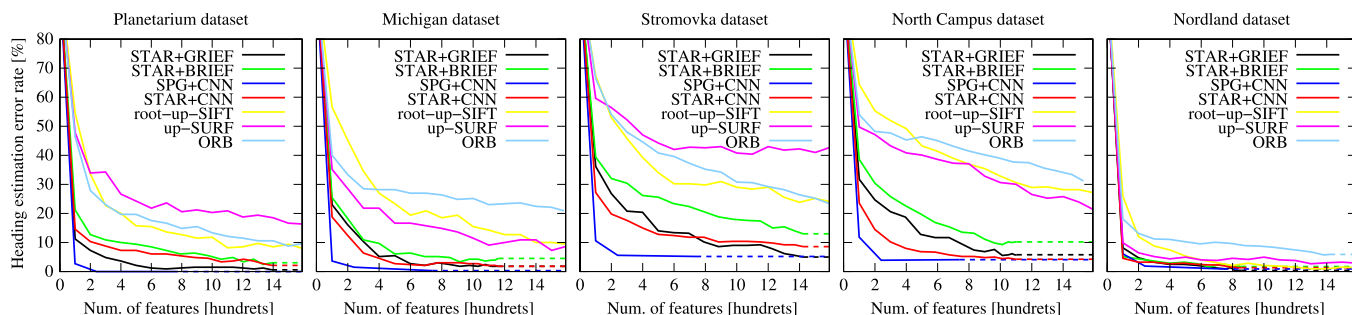


Fig. 12. The dependence of heading estimation error rate on the number of features extracted. Dashed lines indicate that the given detector was unable to extract the number of keypoints required.

Table 1
Error rates of various detector/descriptor combinations in the Planetarium dataset, assuming 1600 features per image.

	GRIEF	BRIEF	rSIFT	SIFT	SURF	BRISK	FREAK	ORB	CNN
SpG ¹	3.3	5.2	5.2	4.2	142	5.8	148	103	0.0
STAR	0.6	3.0	1.2	0.9	376	9.1	348	7.6	2.1
BRISK	0.3	2.4	0.9	0.9	345	8.5	264	7.3	-
uSIFT	2.4	9.1	7.9	9.7	47.0	182	367	100	-
SIFT	2.4	9.1	20.0	25.2	47.0	31.5	367	185	-
uSURF	0.3	2.4	3.0	2.1	164	0.9	173	5.2	-
SURF	0.3	2.4	3.0	2.4	164	145	173	203	-
ORB	2.1	9.1	6.1	6.1	133	6.1	294	9.7	-
FAST	2.4	4.2	2.1	2.1	436	112	312	7.6	-
MSER	3.9	11.5	10.0	6.4	306	103	345	182	-
GFTT	2.7	5.8	9.1	11.2	52.1	121	300	115	-

Table 2
Error rates of various detector/descriptor combinations in the Michigan dataset, assuming 1600 features per image.

	GRIEF	BRIEF	rSIFT	SIFT	SURF	BRISK	FREAK	ORB	CNN
SpG ^a	1.5	4.5	6.1	7.9	9.4	1.5	3.3	7.9	0.3
STAR	1.8	4.5	6.1	6.7	239	9.1	139	3.9	1.8
BRISK	1.5	3.0	3.9	5.5	142	3.0	121	4.2	-
uSIFT	8.5	9.7	9.4	10.6	35.2	130	279	11.5	-
SIFT	8.5	9.7	13.0	15.8	35.2	167	279	7.6	-
uSURF	1.5	2.4	7.3	8.5	8.8	2.1	3.3	5.8	-
SURF	1.5	2.4	3.9	5.5	8.8	5.8	3.3	8.2	-
ORB	15.5	20.0	24.8	28.5	25.8	18.8	24.2	20.9	-
FAST	7.3	8.8	8.5	9.1	20.9	6.1	15.2	9.4	-
MSER	16.1	21.8	13.6	15.5	27.6	14.5	32.4	21.8	-
GFTT	9.4	9.1	7.6	10.3	36.4	9.7	22.1	14.2	-

^a The Superpixel Grid detector (SpG) used 740 keypoints.

Table 3
Error rates of various detector/descriptor combinations in the Stromovka dataset, assuming 1600 features per image.

	GRIEF	BRIEF	rSIFT	SIFT	SURF	BRISK	FREAK	ORB	CNN
SpG ¹	164	260	9.8	10.2	360	184	288	340	5.2
STAR	5.0	130	11.2	9.4	642	346	602	300	8.6
BRISK	8.6	15.0	7.2	7.0	60.2	25.8	45.8	32.6	-
uSIFT	16.2	23.4	24.0	25.8	68.4	44.2	65.4	30.2	-
SIFT	16.2	23.4	40.8	45.4	68.4	58.8	65.4	45.4	-
uSURF	9.2	12.8	8.0	7.8	42.8	12.8	36.2	29.8	-
SURF	9.0	12.8	15.2	13.6	42.8	42.2	36.2	57.8	-
ORB	21.4	28.8	11.8	12.2	27.6	20.6	43.8	23.2	-
FAST	9.6	12.8	17.2	14.0	63.2	30.8	54.4	26.2	-
MSER	23.4	39.6	26.6	21.8	63.8	35.6	59.2	51.6	-
GFTT	10.4	19.6	25.0	24.6	65.8	36.6	59.4	27.2	-

Table 4

Error rates of various detector/descriptor combinations in North Campus dataset, assuming 1600 features per image.

	GRIEF	BRIEF	rSIFT	SIFT	SURF	BRISK	FREAK	ORB	CNN
SpG ^a	9.5	11.7	9.1	10.6	199	9.1	21.7	17.8	4.1
STAR	5.8	10.2	8.0	8.9	37.3	13.2	37.7	14.7	4.3
BRISK	6.1	8.0	6.1	6.7	28.4	10.8	32.8	13.2	–
uSIFT	16.7	25.2	27.1	30.4	54.4	29.1	48.4	27.8	–
SIFT	16.7	25.2	40.3	42.5	54.4	39.0	48.4	36.4	–
uSURF	6.3	7.6	8.2	9.5	21.2	6.1	17.1	8.3	–
SURF	6.1	7.6	11.9	13.2	21.2	20.6	17.1	17.4	–
ORB	25.8	34.9	31.2	31.9	45.6	27.6	53.4	31.2	–
FAST	15.6	18.7	20.2	21.9	51.6	26.0	47.5	25.0	–
MSER	27.3	36.2	23.0	24.3	46.2	26.0	48.4	37.3	–
GFTT	19.7	25.8	28.9	31.7	63.3	32.5	53.8	32.1	–

^a The Superpixel Grid detector (SpG) used 740 keypoints.**Table 5**

Error rates of various detector/descriptor combinations in Nordland dataset, assuming 1600 features per image.

	GRIEF	BRIEF	rSIFT	SIFT	SURF	BRISK	FREAK	ORB	CNN
SpG ^a	5.1	6.7	1.3	1.5	6.7	5.5	8.6	21.9	0.9
STAR	0.4	1.7	1.7	1.3	18.7	2.5	9.0	7.8	1.3
BRISK	0.6	0.6	0.8	1.0	13.9	1.0	1.9	4.6	–
uSIFT	1.3	0.6	1.3	1.0	28.8	2.9	4.4	2.9	–
SIFT	1.3	0.6	2.9	2.9	28.8	6.9	4.4	2.5	–
uSURF	0.4	1.0	1.1	0.8	2.9	0.4	1.9	5.0	–
SURF	0.4	1.0	2.1	1.7	2.9	2.3	1.9	8.0	–
ORB	5.7	4.0	4.4	4.2	3.4	4.0	11.6	5.9	–
FAST	0.6	0.6	1.1	1.1	11.6	1.1	2.5	2.1	–
MSER	45.9	50.3	25.5	24.2	51.8	47.2	60.2	58.1	–
GFTT	2.1	2.1	1.7	2.3	33.5	2.7	3.2	5.0	–

^a The Superpixel Grid detector (SpG) used 740 keypoints.

Matusiak and Skulimowski [57] report that the combination of the FAST detector and SIFT descriptor results in a computationally more efficient feature with similar robustness to the original SIFT. This lead us to test other detector/descriptor combinations of the features that we use. Tables 1–5 contain the error rates of the feature extractor algorithms obtained by combining different detectors and descriptors.

The results summarized in Tables 1–5 confirm the high robustness of the STAR/GRIEF and SpG/CNN combinations to seasonal changes. Moreover, the results also indicate that the default detector/descriptor combinations are often not the best ones and one should consider alternative combinations. For example, exchanging the detector phase of the root-SIFT algorithm with the BRISK method dramatically improves invariance to seasonal changes. Due to the high computational costs of the CNN descriptor, we evaluated it only with the STAR and SpG detectors, since the first showed the best results with the other descriptors and the latter is a region detector particularly developed for combination with rich descriptors like CNN-based ones.

6.7. Computational efficiency

An important property of an image feature is the amount of processing time required for its extraction from an image and the amount of time it takes to match it to the map. In our evaluations, we calculated the times it takes to detect, describe and match the given features and normalized this time per 1000 features extracted. The estimate is only coarse because the time required for feature detection is more dependent on the image size than on the number of features and the feature matching speed can be boosted by techniques like approximate nearest neighbor [58]. Moreover,

detectors and descriptors of the same features often share data structures, which means that if used together, the time for their extraction is lower than the sum of the detection and description times indicated in Table 6. However, the statistics shown in Table 6 are still useful to rank the algorithms according to their computational efficiency. The Table 6 shows the times to extract and match the conventional image features on an i7 processor and the CNN features on an NVidia Titan X GPU. We omitted the upright variants of SIFT and SURF as well as root SIFT, because their computational time is the same. The computational complexity of the GRIEF descriptor is the same as the BRIEF one, which is not surprising because these two algorithms differ only in the choice of pixel positions used for brightness comparisons. Table 6 shows that the combination of the STAR detector and (G)BRIEF descriptor is computationally inexpensive not only for the extraction itself, but also for matching. It also indicates that the CNN descriptor is computationally expensive—calculation of a single descriptor takes 3ms on a GPU, which is three orders of magnitude longer than BRIEF. Moreover, matching 1000 CNN descriptors takes more than a second, which is also significantly slower compared to the classic features. However, matching could be speeded up by techniques tailored for high-dimensional descriptors, e.g. binary locality-sensitive hashing [59].

6.8. Discussion

The results presented in Sections 6.5–6.7 indicate that the CNN-based descriptors in combination with the Superpixel Grid detector achieve low error rates even with a low number of detected features. When using a large number of keypoints, the performance of the SpG/CNN and STAR/GRIEF features evens out, and they both

Table 6

Time required to detect, describe and match 1000 features by the feature extractors used in our evaluation.

Method	Time [ms] required to		
	Detect	Describe	Match
SIFT	200	64	85
SURF	99	63	93
BRISK	63	5	64
ORB	8	6	58
BRIEF	–	3	63
GRIEF	–	3	60
FREAK	–	15	58
CNN-CPU	–	33000	1650
CNN-GPU	–	3100	1650
MSER	75	–	–
GFTT	16	–	–
STAR	16	–	–
FAST	9	–	–
SPGrid	49	–	–

achieve low heading estimation error rates. While the SpG/CNN performs better on the Michigan, North Campus and Planetarium datasets, which contain a higher number of man-made structures, the STAR/GRIEF achieves lower errors on the Stromovka and Nordland datasets, which contain a larger amount of foliage that exhibits significant appearance changes due to seasonal factors. Compared to the CNN features, the GRIEF is much faster to calculate even on an ordinary CPU.

Our analysis assumes a teach-and-repeat scenario, where a robot moves along a previously-taught path and thus, the visual navigation method does not have to be robust to large viewpoint changes. In a realistic scenario, a robot might have to deviate from the taught path, e.g. due to an obstacle. In order to deal with these situations, the image features used should still be able to handle small-scale viewpoint changes. Experiments with ground [56] and aerial [60] robots have shown that teach-and-repeat systems based on the STAR/BRIEF feature routinely deal with position deviations of up to 1 meter.

Unlike SpG/CNN, which is designed for general use, the STAR/GRIEF combination is not meant to handle large viewpoint changes and one should be cautious when applying it for general long-term navigation and localization. For example, [61] evaluated the performance of several image features in a scenario of lakeshore monitoring, where the on-board camera aims perpendicularly to the vehicle movement and thus, the viewpoint changes are significant. The authors of [61] concluded that in their scenario, the ORB feature, which is based on BRIEF, slightly outperformed the other features.

7. Conclusion

We report our results on the evaluation of image feature extractors to mid- and long-term environment changes caused by variable illumination and seasonal factors. Our evaluation was taken from the navigational point of view—it was based on the feature extractors' ability to correctly establish the robot's orientation, and hence, keep it on the intended trajectory. The datasets used for evaluation capture seasonal appearance changes of three outdoor environments from two different continents.

Motivated by previous works which indicated that certain combinations of feature detectors and descriptors outperform commonly used features, we based our evaluation on combinations of publicly-available detectors and descriptors. For example, substituting the detection phase of the root-SIFT algorithm with the BRISK method dramatically improves its invariance to seasonal changes, while making the algorithm computationally more efficient. We noted that the BRIEF descriptor based on bitwise

comparisons of the pixel intensities around a keypoint detected by the STAR method performed better than most other detector/descriptor combinations. To further elaborate on this result, we trained the comparison sequences that constitute the core of the BRIEF descriptor on a limited number of images, obtaining a new feature, which we call GRIEF.

The lowest registration error rates (2.4% and 3.0%) were achieved by the SpG/CNN and STAR/GRIEF detector/descriptor combinations, which makes these features a good choice for vision-based teach-and-repeat systems operating in outdoor environments for long periods of time. While the SpG/CNN performed better in semi-urban areas, the performance of STAR/GRIEF was slightly higher in environments with natural features such as foliage, where it was trained on. Moreover, the STAR/GRIEF feature was faster to calculate, which makes it suitable even for resource-constrained systems. We hope that this evaluation will be useful for other researchers concerned with long-term autonomy of mobile robots in challenging environments and will help them to choose the most appropriate image feature extractor for their navigation and localization systems. To allow further analysis of this problem, we provide the aforementioned benchmarking framework and the GRIEF training method as a documented, open-source software package [21].

Acknowledgments

The work has been supported by the EU ICT project 600623 'STRANDS' and UBACYT project 20020130300035BA. We would like to thank Nicholas Carlevaris-Bianco for sharing the Michigan dataset.

References

- [1] J. Li, N. Allinson, A comprehensive review of current local features for computer vision, *Neurocomputing* (2008).
- [2] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (10) (2005) 1615–1630. <http://dx.doi.org/10.1109/TPAMI.2005.188>.
- [3] D. Mukherjee, Q. JonathanWu, G. Wang, A comparative experimental study of image feature detectors and descriptors, *Mach. Vis. Appl.* (2015) 1–24. <http://dx.doi.org/10.1007/s00138-015-0679-9>.
- [4] D.G. Lowe, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vis.* 60 (2) (2004) 91–110.
- [5] S. Gauglitz, T. Höllerer, M. Turk, Evaluation of interest point detectors and feature descriptors for visual tracking, *Int. J. Comput. Vis.* 94 (3) (2011) 335–360.
- [6] A. Gil, O. Mozos, M. Ballesta, O. Reinoso, A comparative evaluation of interest point detectors and local descriptors for visual SLAM, *Mach. Vis. Appl.* (2010) 905–920. <http://dx.doi.org/10.1007/s00138-009-0195-x>.
- [7] S. Lowry, N. Sunderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, M. Milford, Visual place recognition: A survey, *IEEE Trans. Robot. PP* (99) (2015) 1–19. <http://dx.doi.org/10.1109/TRO.2015.2496823>.
- [8] N. Sunderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, M. Milford, Place recognition with ConvNet landmarks: Viewpoint-robust, condition-robust, training-free, *Robot. Sci. Syst. XII* (2015).
- [9] W. Churchill, P. Newman, Practice makes perfect? Managing and leveraging visual experiences for lifelong navigation, in: *ICRA*, 2012.
- [10] P. Furgale, T.D. Barfoot, Visual teach and repeat for long-range rover autonomy, *J. Field Robot.* (2010).
- [11] T. Krajník, J. Faigl, V. Vonásek et al., Simple, yet stable bearing-only navigation, *J. Field Robot.* (2010).
- [12] Z. Chen, S.T. Birchfield, Qualitative vision-based path following, *IEEE Trans. Robot. Autom.* (2009). <http://dx.doi.org/10.1109/TRO.2009.2017140>.
- [13] C. Valgren, A.J. Lilienthal, SIFT, SURF & seasons: Appearance-based long-term localization in outdoor environments, *Robot. Auton. Syst.* 58 (2) (2010) 157–165.
- [14] E. Royer, M. Lhuillier, M. Dhome, J.-M. Lavest, Monocular vision for mobile robot localization and autonomous navigation, *Int. J. Comput. Vis.* (2007).
- [15] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: binary robust independent elementary features, in: *ICCV*, 2010.
- [16] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: *International Conference on Computer Vision*, Barcelona, 2011.

- [17] S. Leutenegger, M. Chli, R.Y. Siegwart, BRISK: Binary robust invariant scalable keypoints, in: 2011 International conference on computer vision, IEEE, 2011, pp. 2548–2555.
- [18] P. Neubert, P. Protzel, Local region detector+ CNN based landmarks for practical place recognition in changing environments, in: ECMR, IEEE, 2015, pp. 1–6.
- [19] T. Krajník, P. Cristóforis, M. Nitsche, K. Kusumam, T. Duckett, Image features and seasons revisited, in: European Conference on Mobile Robots, ECMR, IEEE, 2015, pp. 1–7.
- [20] N. Carlevaris-Bianco, A.K. Ushani, R.M. Eustice, University of Michigan North Campus long-term vision and lidar dataset, *Int. J. Robot. Res.* (2015).
- [21] T. Krajník, GRIEF source codes and benchmarks, URL <http://purl.org/robotics/grief-code>.
- [22] G.D. Finlayson, S.D. Hordley, Color constancy at a pixel, *J. Opt. Soc. America: Optics, Image Sci. Vis.* 18 (2) (2001) 253–64.
- [23] W. Maddern, A.D. Stewart, C. McManus, B. Upcroft, W. Churchill, P. Newman, Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles, in: ICRA Workshop on Visual Place Recognition in Changing Environments, 2014.
- [24] C. McManus, W. Churchill, W. Maddern, A. Stewart, P. Newman, Shady dealings: Robust, long-term visual localisation using illumination invariance, in: International Conference on Robotics and Automation, ICRA, 2014, pp. 901–906.
- [25] K. MacTavish, M. Paton, T. Barfoot, Beyond a shadow of a doubt: Place recognition with colour-constant images, in: Field and Service Robotics, FSR, 2015.
- [26] M. Paton, K. MacTavish, C. Ostafew, T. Barfoot, It's not easy seeing green: Lighting-resistant stereo visual teach-and-repeat using color-constant images, in: International Conference on Robotics and Automation, ICRA, 2015.
- [27] J. Mount, M. Milford, 2d visual place recognition for domestic service robots at night, in: International Conference on Robotics and Automation, ICRA, 2016.
- [28] F. Dayoub, T. Duckett, An adaptive appearance-based map for long-term topological localization of mobile robots, in: IROS, 2008.
- [29] D.M. Rosen, J. Mason, J.J. Leonard, Towards lifelong feature-based mapping in semi-static environments, in: International Conference on Robotics and Automation, ICRA, IEEE, 2016.
- [30] N. Carlevaris-Bianco, R.M. Eustice, Learning visual feature descriptors for dynamic lighting conditions, in: IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IROS, 2014.
- [31] S. Lowry, G. Wyeth, M. Milford, Unsupervised online learning of condition-invariant images for place recognition, *Australas. Conf. Robot. Auto.* (2014).
- [32] T. Cieslewski, E. Stumm, A. Gawel, M. Bosse, S. Lynen, R. Siegwart, Point cloud descriptors for place recognition using sparse visual information.
- [33] S. Lowry, M. Milford, G. Wyeth, Transforming morning to afternoon using linear regression techniques, in: International Conference on Robotics and Automation, ICRA, IEEE, 2014.
- [34] P. Neubert, N. Sünderhauf, P. Protzel, Appearance change prediction for long-term navigation across seasons, in: ECMR, 2013.
- [35] N. Sünderhauf, P. Neubert, P. Protzel, Predicting the change—a step towards life-long operation in everyday environments, in: Robotics Challenges and Vision, RCV2013, 2014.
- [36] T. Krajník, J. Fentanes, G. Cielniak, C. Dondrup, T. Duckett, Spectral analysis for long-term robotic mapping, in: International Conference on Robotics and Automation, ICRA, 2014.
- [37] T. Krajník, J.P. Fentanes, O.M. Mozos, T. Duckett, J. Ekekrantz, M. Hanheide, Long-term topological localization for service robots in dynamic environments using spectral maps, in: Int. Conf. on Intelligent Robots and Systems, IROS, 2014.
- [38] C. McManus, B. Upcroft, P. Newmann, Scene signatures: localised and point-less features for localisation, in: RSS, 2014.
- [39] T. Krajník, S. Pedre, L. Přeučil, Monocular navigation system for long-term autonomy, in: International Conference on Advanced Robotics, ICAR, 2013.
- [40] N. Sünderhauf, F. Dayoub, S. Shirazi, B. Upcroft, M. Milford, On the performance of convnet features for place recognition, 2015. arXiv preprint [arXiv:1501.04158](https://arxiv.org/abs/1501.04158).
- [41] N. Sünderhauf, P. Corke, Visual Place Recognition in Changing Environments (VPRiCE), <https://roboticvision.atlassian.net/wiki/pages/viewpage.action?pageId=14188617>.
- [42] D. Mishkin, M. Perdoch, J. Matas, Place recognition with WxBS retrieval, in: CVPR 2015 Workshop on Visual Place Recognition in Changing Environments, 2015.
- [43] K. Mikolajczyk, C. Schmid, An affine invariant interest point detector, in: European Conference on Computer Vision, 2002.
- [44] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, *Image and vision computing* 22 (10) (2004) 761–767.
- [45] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: European Conf. on Computer Vision, 2006.
- [46] E. Mair, G.D. Hager, D. Burschka, M. Suppa, G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test, in: European Conference on Computer Vision, 2010.
- [47] M. Agrawal, K. Konolige, M.R. Blas, Censure: Center surround extremas for realtime feature detection and matching, in: European Conf. on Computer Vision, ECCV, Springer, 2008, pp. 102–115.
- [48] P. Neubert, P. Protzel, Beyond holistic descriptors, keypoints, and fixed patches: Multiscale superpixel grids for place recognition in changing environments, *IEEE Robot. Auto. Lett.* 1 (1) (2016) 484–491. <http://dx.doi.org/10.1109/LRA.2016.2517824>.
- [49] R. Arandjelović, A. Zisserman, Three things everyone should know to improve object retrieval, in: Computer Vision and Pattern Recognition, CVPR, 2012, pp. 2911–2918.
- [50] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features, *SURF, Comput. Vis. Image Underst.* (2008).
- [51] A. Alahi, R. Ortiz, P. Vandergheynst, FREAK: Fast retina keypoint, in: IEEE conference on Computer vision and pattern recognition, CVPR, IEEE, 2012.
- [52] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, in: British Machine Vision Conference, 2014.
- [53] M.A. Fischler, R.C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM* (1981) 381–395. <http://dx.doi.org/10.1145/358669.358692>.
- [54] Stromovka Dataset, [Cit: 2013-03-25]. URL http://purl.org/robotics/stromovka_dataset.
- [55] D.G.R. Bradski, A. Kaehler, Learning OpenCV, first ed., O'Reilly Media, Inc., 2008.
- [56] P. De Cristóforis, M. Nitsche, T. Krajník, T. Pire, M. Mejail, Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments, *Pattern Recognit. Lett.* (2015).
- [57] K. Matusiak, P. Skulimowski, Comparison of key point detectors in SIFT implementation for mobile devices, *Comput. Vis. Graph.* (2012) 509–516.
- [58] E. Kushilevitz, R. Ostrovsky, Y. Rabani, Efficient search for approximate nearest neighbor in high dimensional spaces, *SIAM J. Comput.* 30 (2) (2000) 457–474.
- [59] M.S. Charikar, Similarity estimation techniques from rounding algorithms, in: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 380–388.
- [60] M. Nitsche, T. Pire, T. Krajník, M. Kulich, M. Mejail, Monte Carlo localization for teach-and-repeat feature-based navigation, in: Towards Autonomous Robotics Systems, TAROS, 2014.
- [61] S. Griffith, C. Pradalier, Survey registration for long-term natural environment monitoring, *J. Field Robot.* (2016).



Tomáš Krajník is a research fellow at the Lincoln Center of Autonomous Systems, UK. He received the Ph.D. degree in Artificial Intelligence and Biocybernetics from the Czech Technical University, Prague, Czech Republic, in 2012. His research interests include life-long autonomous navigation, spatio-temporal modeling, and aerial robots.



Pablo De Cristóforis received the Ph.D degree in Computer Science from the University of Buenos Aires, Argentina in 2013. He is currently a research assistant at the National Council of Scientific and Technological Research (CON-ICET), Argentina. His research interests include autonomous vision-based navigation, visual SLAM and 3D vision reconstruction for mobile robotics.



Keerthy Kusumam obtained her masters by research in computer vision in 2015 from the university of Lincoln. She is currently working as a research assistant in computer vision and machine learning at the Lincoln Centre for Autonomous Systems Research. Her main interests are computer vision and machine learning.



Peer Neubert is a researcher at the chair for automation technology at the Technische Universität Chemnitz. He received his Ph.D degree in 2015 from the same university. His research interests include computer vision, machine learning and artificial intelligence in particular for application in the area of autonomous mobile robots. A key aspect is the transformation of biologically inspired models, concepts and ideas to application on computers.



Tom Duckett is a Professor of Computer Science at the University of Lincoln, UK, where he also leads the Lincoln Centre for Autonomous Systems. His research interests include autonomous robots, artificial intelligence and machine perception, with applications including service robotics and assistive technologies. Tom has co-authored over 100 scientific publications and held peer-reviewed grants worth over 1.5 million at the University of Lincoln.

D

KEY ARTICLE [14] - ICRA 2014

©[2014] IEEE. Reprinted, with permission, from Tomáš Krajník, Spectral Analysis for Long-Term Robotic Mapping, 2014.

Spectral Analysis for Long-Term Robotic Mapping

Tomáš Krajník Jaime Pulido Fentanes Grzegorz Cielniak Christian Dondrup Tom Duckett

Abstract—This paper presents a new approach to mobile robot mapping in long-term scenarios. So far, the environment models used in mobile robotics have been tailored to capture static scenes and dealt with the environment changes by means of ‘memory decay’. While these models keep up with slowly changing environments, their utilization in dynamic, real world environments is difficult.

The representation proposed in this paper models the environment’s spatio-temporal dynamics by its frequency spectrum. The spectral representation of the time domain allows to identify, analyse and remember regularly occurring environment processes in a computationally efficient way. Knowledge of the periodicity of the different environment processes constitutes the model predictive capabilities, which are especially useful for long-term mobile robotics scenarios.

In the experiments presented, the proposed approach is applied to data collected by a mobile robot patrolling an indoor environment over a period of one week. Three scenarios are investigated, including intruder detection and 4D mapping. The results indicate that the proposed method allows to represent arbitrary timescales with constant (and low) memory requirements, achieving compression rates up to 10^6 . Moreover, the representation allows for prediction of future environment states with $\sim 90\%$ precision.

Index Terms—long-term autonomy, mobile robotics, spatio-temporal mapping

I. INTRODUCTION

Long-term robotic autonomy has not yet been achieved due to many challenges, one of them being the fact that robotic mapping is vulnerable to changes in the environment. However, many future tasks performed by mobile robots will be carried out in places where humans perform their usual activities, causing the environment to change. Many of these activities consist of daily routines that are performed on a regular basis. This regularity can be exploited by the robots to build more robust representations of their surroundings. To address this issue, we propose to treat the observed environment states as signals over which a spectral analysis can be performed to identify regularities of environmental variations occurring on timescales from days to months. Knowledge of these regularities allows efficient representation of the world model’s temporal domain as well as prediction of the environment properties at a specific time.

This ability would allow to improve many tasks that the mobile robots have to perform. For example, a robot can predict the appearance of a particular location at a certain time in order to achieve more robust localization. Moreover, higher-level path planning can use the approach to predict which doors will be open or which areas will be difficult to navigate because they will be crowded at particular times.

Lincoln Centre for Autonomous Systems, University of Lincoln, UK
tkrajnik@lincoln.ac.uk

Mapping a static environment is a problem that has been widely studied for a long time [1], however mapping dynamic environments is still an open problem. So far, mapping dynamic environments has been done by removing moving objects from the representation of the environment [2], [3] or by tracking these objects and classifying them as moving landmarks [4], [5], [6]. In general, separation-based approaches can handle some problems of dynamic mapping, but they assume that most of the environment is static, which makes them unsuitable for scenarios where the environment appearance varies significantly. In [7] a new map type that represents local maps at different time scales is presented, where the best map for localisation is chosen by its consistency with current readings, which provided improved localisation over large time scales.

Adaptive approaches never assume the map to be complete and perform continuous mapping, adding new features to the map every time the robot observes its environment. In these approaches, the key problem is managing map size [8], [9], [10]. [11] presents a feature persistence system based on temporal stability in sparse vision-based maps. Some authors propose systems that learn a fixed set of possible states for the dynamic objects in the environment, e.g. corresponding to open and closed doors [12], [13], but this approach is limited in the real world, where the number of states is unpredictable and this approach does not offer state prediction capabilities.

[14] proposes a new representation that models occupancy grid maps in the wavelet space in order to optimize the amount of information that has to be processed for path planning. [15] presents a representation which models transitions of dynamic objects in the environment. Spectral analysis has also been used for 2D [16] and 3D [17] registration. Finally, [18] has shown state prediction can be useful on long-term robotic tasks by proposing an implementation of an appearance change prediction (ACP) method to improve the performance of place recognition on a large-scale dataset under extremely different conditions.

The representation proposed in this paper models the environment’s spatio-temporal dynamics by its frequency spectrum. We propose to model local states of an environment by means of a probability function which is a superposition of periodical functions. The model includes recurrent environment changes, which improves the robot’s knowledge of its surroundings and the events that take place in them. We hope that this proposal becomes a useful tool for modelling the environment and leads to developing new representations of the environment that consider the spatio-temporal dynamics that take place there.

II. SPECTRAL REPRESENTATION FOR SPATIO-TEMPORAL ENVIRONMENT MODELS

Most mapping approaches assume that the principal components of a particular environment model can be in two distinct states. For example, cells of an occupancy grid are occupied or free, edges of a topological map are traversable or not, doors are opened or closed, rooms are vacant or occupied, landmarks are visible or occluded, etc. In a typical situation, the state of each model component is uncertain, because it is measured indirectly by means of sensors which are affected by noise. A common way to represent the uncertainty in the state estimate of the j^{th} world model component is by its associated probability p_j . This allows to counter the effect of noisy measurements by employing statistical methods, such as Bayesian filtering. While Bayesian filtering methods allow to keep up with a changing environment, the mathematical foundations they are based on assume a static world, i.e. the p_j of the world components are assumed to be constant. As a result, a change in the environment causes the old state to be ‘forgotten’ over time.

Once we assume that p_j is a function of time, we need to outline a suitable representation for $p_j(t)$. Although one could simply store the whole history of the environment model, such an approach would quickly face memory limitations. Typical static 3D models of of complex environments contain millions of distinct components [19]. Storing the entire model history is infeasible. Moreover, in the context of robotic mapping, it is not clear how to utilize the past estimates of the environment models, i.e. what is the relation of the past models to the current state of the world.

In our approach we assume that the variations of the environment are caused by a number of unknown processes, which might be periodical. If we could identify the influence and periodicity of these processes, we could then calculate the probabilities $p_j(t)$ from the description of these processes. To identify these periodical processes, we propose to use a frequency transform, namely the Fourier transform [20].

A. An introduction to the Fourier Transform

The Fourier Transform (FT) is a well-established mathematical tool widely used in the field of statistical signal processing. It transforms a function of time $f(t)$, into a function of frequency $F(\omega)$, i.e. $F(\omega)=\mathcal{F}(f(t))$. The function $F(\omega)$ is commonly referred to as the frequency spectrum of $f(t)$. The Fourier transform is invertible, and therefore, one can recover the function $f(t)$ from its spectrum $F(\omega)$, i.e. $f(t) = \mathcal{F}'(F(\omega))$. If one wants to analyze or alter the periodic properties of a process characterized by a function $f(t)$, it is reasonable to calculate its spectrum $F(\omega)$, perform the analysis or alteration in the frequency domain, and then transform the altered spectrum $F'(\omega)$ back to the temporal domain. Such a process is referred to as spectral analysis.

Typically, $F(\omega)$ is a complex-valued function, whose absolute values and arguments correspond to the amplitudes and phase shifts of the frequency components ω . Considering

that $f(t)$ is a real periodical discrete function, the spectrum $F(\omega)$ can be represented by a finite set of complex numbers.

B. The proposed representation

Let us represent the environment as a set of independent components, which can be in two distinct states. Without loss of generality, we will explain our approach with an occupancy grid. Let us assume that each grid cell state $s_j = \{\textit{occupied}, \textit{free}\}$ is not constant, but is a function of time, i.e. $s_j(t)$. The uncertainty of the state $s_j(t)$ is represented by its probability $p_j(t)$. Now, let us assume that the occupancy of each grid cell is affected by a set of unknown periodical processes, which can be identified by the Fourier Transform. Since we assume the occupancy of the individual cells to be independent, we can explain the use of the Fourier transform on the state $s(t)$ of a single cell.

1) *The spectral model:* The main idea behind the proposed model is to measure the temporal sequence of states $s(t)$ and calculate their frequency spectrum by means of a Fourier Transform as $\mathcal{S}(\omega) = \mathcal{F}(s(t))$. Then, we select the l most prominent (i.e. of highest absolute value) coefficients S_i of the spectrum \mathcal{S} and store them along with their frequencies ω_i in a set \mathcal{P} . The coefficients stored in \mathcal{P} are then used to recover the smoothed signal $p(t)$ by means of the Inverse Fourier Transform $p(t) = \zeta(\mathcal{F}'(\mathcal{P}(\omega)))$, where ζ denotes a saturation function that ensures that $p(t) \in [0, 1]$. One can easily verify that $P(s(t) = \textit{occupied}) = p(t) \geq 0$ and $P(s(t) = \textit{occupied}) + P(s(t) = \textit{free}) = p(t) + 1 - p(t) = 1$, i.e. $p(t)$ satisfies Kolmogorov’s axioms and therefore is a probability. Thresholding the probability $p(t)$ allows us to calculate an estimate $s'(t)$ of the original state $s(t)$. In order not to lose any information of the original signal, the differences between $s'(t)$ and $s(t)$ are stored in an outlier set \mathcal{O} , which is Δ -encoded, see Figure 1.

Thus, our model of the state consists of two finite sets \mathcal{P} and \mathcal{O} . The set \mathcal{P} consists of l triples $abs(P_i)$, $arg(P_i)$ and ω_i , which describe the amplitudes, phase shifts and frequencies of the model spectra. Each such triple might be interpreted as the importance, time offset and periodicity of one particular periodical process influencing the state $s(t)$. We will refer to the number of modeled processes l (i.e. to the number of triples in \mathcal{P}) as the ‘order’ of the spectral model. The set \mathcal{O} represents a set of k time intervals, during which the state $s(t)$ did not match the state $s'(t)$ calculated from $p(t)$. Internally, the set \mathcal{O} is implemented as a sequence of values, indicating the starts and ends of time intervals when the predicted and observed state did not match, i.e. $s'(t) \neq s(t)$. Each interval is represented by its limits $< t_{2k}, t_{2k+1}$.

2) *Model adaptation:* To be able to build, maintain and use this representation, we define four operations: reconstruction of the original state $s(t)$, addition of a new measurement, model update and prediction of the future state with a given confidence level. The aforementioned representation allows us to retrieve the cell state $s(t)$ by means of the following equation:

$$s(t) = (\mathcal{F}'(\mathcal{P}) > 0.5) \oplus (t \notin \mathcal{O}), \quad (1)$$

where \oplus is a XOR operation. The idea behind this equation is to reconstruct the probability $p(t)$ from the spectrum \mathcal{P} , set $s(t)$ to 1 if $p(t)$ exceeds 0.5 and finally to negate $s(t)$ if t belongs to the set of outliers \mathcal{O} .

Whenever a real state $s^m(t)$ is measured, we calculate $s(t)$ by means of Equation (1) and if it differs from $s^m(t)$, the current time t is added to the set \mathcal{O} :

$$s^m(t) \neq ((\mathcal{F}'(\mathcal{P}) > 0.5) \oplus (t \notin \mathcal{O})) \rightarrow \mathcal{O} = \mathcal{O} \cup t. \quad (2)$$

Since $p(t)$ does not predict $s(t)$ with perfect accuracy, the set \mathcal{O} is likely to grow larger as measurements are added.

To update the model, we reconstruct $s(t)$ in the desired time interval $\langle t_{start}, t_{end} \rangle$ and calculate its spectrum \mathcal{P} . Again, we select the l coefficients with highest absolute values $|P_i|$ and reconstruct the outlier set \mathcal{O} by means of Equation 2. In a typical situation, the updated spectrum \mathcal{P} would reflect $s(t)$ more accurately, causing reduction of the set \mathcal{O} . Note, that the spectral model order l can be changed prior to the update step without causing any loss of information.

3) *Prediction*: Note, that the Equation (1) allows for calculating $s(t)$ for any t and that the threshold value of 0.5 can be set arbitrarily. In fact, a threshold c such that $P(s(t) = occupied) > c$ represents a confidence level of the grid cell being full at time t . Therefore, we can use Equation (1) for future prediction of $s(t)$ with a certain confidence level c . In the case of prediction, the outlier set \mathcal{O} is not included in the calculation and the predicted state might not match the real state, so we denote it as $s'(t, c)$. To simplify notation, we also define $s'(t)$ as $s'(t, 0.5)$. Therefore, $s'(t, c)$ and $s'(t)$ can be calculated as follows:

$$s'(t, c) = \mathcal{F}'(\mathcal{P}) > c. \quad (3)$$

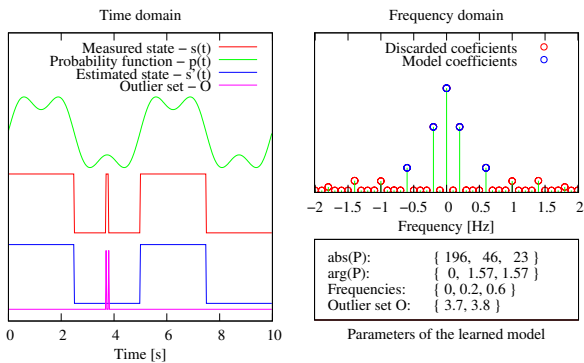


Fig. 1. An example of the measured state and its spectral model. The left part shows the time series of the measured state $s(t)$, probability estimate $p(t)$, predicted state $s'(t)$ and outlier set \mathcal{O} . The upper right part shows the absolute values of the frequency spectrum of $s(t)$ and indicates the spectral coefficients, which are included in the model.

An example of the third-order spectral model which represents a quasi-periodic function is provided in Figure 1. Since the observed processes are not identified perfectly, one can expect that the prediction becomes less and less accurate

over time. However, modelling the uncertainty of the model prediction is outside the scope of this paper.

C. Notes on notation

Throughout the rest of the article, we will keep the notation introduced in Section II-B. Therefore, $s_j(t)$ represents the measured state of the j^{th} world component, $P_{i,j}(\omega_i)$ is a complex number representing amplitude and phase shift of the state's frequency component ω_i , (i.e. i^{th} coefficient of \mathcal{P}), the value of $p_j(t)$ is the state's probability estimate given by the inverse Fourier transform of \mathcal{P} , $s'_j(t)$ is the most probable state at a given time t and $s'_j(t, c)$ is the estimate of $s_j(t) = 1$ with a confidence level c . For the sake of simplicity, the index j will be omitted in the case where our description concerns a single world model component only.

D. Spectrum and periodical processes

In the rest of the article, we will try to examine the tractability of using the Fourier transform as a core component of spatio-temporal models for mobile robotics. In particular, we will investigate the following questions:

- How many parameters of the spectrum typically have to be stored to represent and predict the environment state?
- What is the accuracy and efficiency of the spectral representation?
- Are the predictions of this approach good enough to detect anomalous situations?
- How does the approach deal with sensor noise and localization uncertainty?

To answer these questions, we analysed several types of environment models gathered by a mobile robot, which was continuously operating for more than one week in an indoor environment shared with humans.

III. DATA COLLECTION

Our experimental platform consists of a SCITOS-G5 mobile robot (see Figure 2) equipped with RGB-D and laser sensors. The robot gathered two datasets in two different environments: a staff office and the robotics lab of the Lincoln Centre for Autonomous Systems.

A. The office dataset

The first dataset, considered for basic evaluation, was gathered from a stationary position with sensors aimed at the door of one of the building's offices. The range measurement was used to establish the occupancy of a single cubic cell located in the middle of the room entrance. Since the office has an open door policy, the door remained open whenever the office was occupied by a person. Therefore, the measured state $s(t)$ not only indicated if the particular area of the environment was occupied or free, but also corresponded to the presence of people inside the office. Every time someone went through the door, the monitored area was briefly occupied and the room was considered empty, which introduced noise on the measured state $s(t)$. The measurements were taken continuously for one week (July 23-29 2013) at a rate

of 30Hz, so $s(t)$ consisted of 18 million values. After this week, two additional full-day datasets (July 31 and August 2 2013) were gathered.

B. The laboratory datasets

To gather the second dataset the robot was programmed to visit three designated areas of the robotics lab every five minutes. Each time an area was visited, the robot created a 2D and 3D point cloud and analysed the onboard color camera image to check for the presence of people. Thus, the robot created three datasets of different dimensionality, where each dataset contains observations from three different places, see Figure 2. We will refer to these datasets as Lab-2D, Lab-3D and Lab-1D respectively.



Fig. 2. The robot and its view of two locations of the ‘Lab’ dataset.

The data gathering process started on August 2013 and is still in progress. For this study, we use the data collected during the first week of September consisting of approximately 12,000 point clouds and 6,000 results of people detection.

The autonomous patrolling has been based on combination of the ROS nav stack and the visual localization method proposed in [21]. The robot reports its status regularly [22], so the occasional failures can be dealt with immediately.

IV. ALGORITHM PERFORMANCE

To answer the questions set in Section II-D, we analyse the performance of the proposed representation on the datasets described in Section III.

A. Model accuracy and efficiency

Knowing the coefficients $P_i(\omega_i)$ of the spectrum \mathcal{P} allows us to calculate an estimate $s'(t)$ of the original state $s(t)$. A natural concern is the accuracy of reconstruction of $s'(t)$, which affects the prediction capabilities of the model and the size of the outlier set \mathcal{O} . One can expect that increasing the number of spectral parameters will increase the reconstruction accuracy. However, as the number of parameters grows, the model becomes more adjusted to the specific time series of $s(t)$ and loses its generality. This loss of generality would hamper the ability of the model to predict the environment state in the future.

We define the accuracy of the spectral model $q(t_a, t_b)$ as the ratio of the correctly estimated signal $s'(t)$ on a given time interval $t \in \langle t_0, t_1 \rangle$ to the length of the interval:

$$q(t_a, t_b) = \frac{1}{t_b - t_a} \int_{t_a}^{t_b} |s'(t) - s(t)| dt. \quad (4)$$

In our case, $q = q(0, T)$ can be directly calculated from the values t_i stored in the outlier set \mathcal{O} by

$$q = \frac{1}{T} \sum_{k=0}^{|\mathcal{O}|/2-1} (t_{2k+1} - t_{2k}). \quad (5)$$

Suppose that the spectrum \mathcal{P} is estimated for $s(t)$ within interval $\langle t_c, t_d \rangle$ and q is calculated for an interval $\langle t_a, t_b \rangle$. If $t_c \leq t_a$, then $q(t_a, t_b)$ relates to the accuracy of model prediction and if $\langle t_a, t_b \rangle \in \langle t_c, t_d \rangle$, then q relates to the accuracy of reconstruction.

To estimate the dependence of the accuracy of reconstruction q_r and prediction q_p on the number of model parameters, we built a spectral model of the one-week-long ‘Office’ dataset. The accuracy of reconstruction q_r was calculated as the difference between the original and reconstructed signal. Moreover, we calculated the accuracies of prediction q_{p1} and q_{p2} for two days of the following week, see Figure 3. The

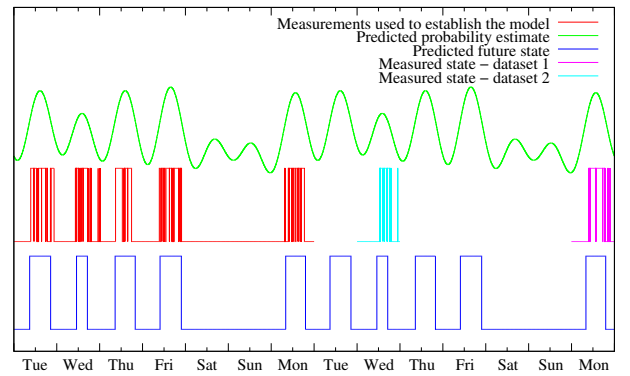


Fig. 3. Comparison of predicted and real values - office dataset.

dependence of the reconstruction and prediction accuracy on the number of parameters of the spectral model is shown in Figure 4. The Figure shows that the spectral model order

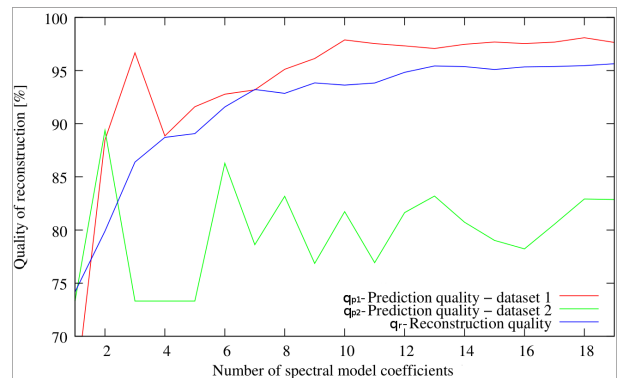


Fig. 4. Model accuracy vs. model complexity - office dataset.

of 15 parameters achieves 95% reconstruction accuracy. As expected, the reconstruction accuracy q_r increases monotonically with the number of spectral model coefficients j , but the prediction quality does not. The local maxima of q_{p1}

and q_{p2} at $l = 2$ and $l = 3$ suggest that for the purpose of prediction, one should use a spectral model of order 3.

The test indicates that the spectral model allows to represent millions of measurements with only a few complex numbers. Thus, the spectral representation \mathcal{P} without the outlier set \mathcal{O} achieves compression ratios in the order of millions while losing less than 5% of information. The full model is composed of 15 triples of spectrum \mathcal{P} and 160 values in the set \mathcal{O} that represent 80 time intervals where the spectral model did not match the measured sequence. Thus, the proposed model achieves lossless compression of the temporal data with a compression ratio reaching $\sim 10^5$.

Since the model that achieved the most accurate signal reconstruction included 15 spectral components, we used this model order for the Lab- n D datasets as well. The reconstruction quality of the ‘Office’ dataset was 0.95 and the corresponding values for the ‘Lab’ datasets are shown in Table I.

TABLE I
RECONSTRUCTION QUALITY FOR DIFFERENT DATASETS

Dataset	Lab-1D	Lab-2D	Lab-3D
Location 1	0.95	0.99	0.99
Location 2	0.99	0.97	1.00
Location 3	0.96	0.98	0.99

The data indicate that the model size remains more or less constant regardless of the time span it covers. Therefore, higher compression rates are achieved simply by representing larger datasets. On average, the proposed model represented the environment state with 98% accuracy while stationary models achieved 95% accuracy.

B. Anomaly detection

An anomalous situation can be defined as a local state of the world which deviates from the internal world model of the robot. Since our model can predict the state $s(t)$ with a given confidence value by Equation 3, we can assume that a measurement $s^m(t)$ is anomalous with confidence level c if

$$s^m(t) \neq (\mathcal{F}'(P) > c). \quad (6)$$

While setting c too high would lower the algorithm’s sensitivity to anomalies, a low value of c would result in an increased number of false positives. An optimal confidence level c can be calculated from the statistical properties of $p(t)$ and the requirements for the number of false positives and failed detections.

Figure 5 shows the results of anomaly detection for the Lab-1D dataset. The anomaly detection was performed “post-hoc”, i.e. we first build the model from the entire dataset and then applied Equation 6. The confidence level c was set to 90% and the anomalous situations correspond to the room being accessed at night or a sudden absence of all people just before and after a meeting, see Figures 6 and 2.

These examples demonstrate how the model adapts its inner dynamics to represent the observed environment. The

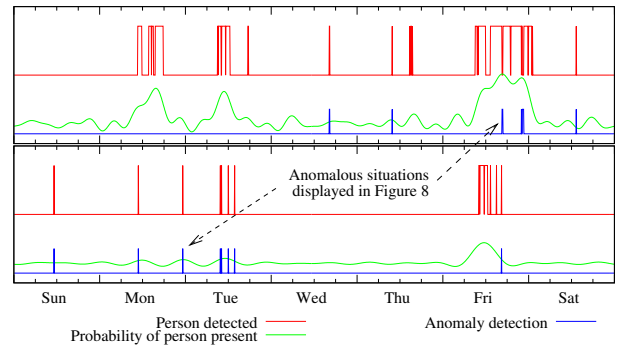


Fig. 5. Anomaly detection for two locations of the Lab-1D dataset.



Fig. 6. Anomalous situations in the ‘Lab’ environment. Left: Workplace empty on Friday early afternoon. Right: Person entering the room at night.

differences between the inner dynamics and real observations allow for natural detection of anomalous situations with arbitrary confidence levels. Figure 5 shows how the spectral models for each laboratory location adapt to the particular location’s dynamics. This results in a different anomaly detector for each location. For example, the present of a person on Monday morning is considered normal for location 1, but triggers the anomaly detector on locations 2 and 3.

C. Building spatiotemporal maps for mobile robotics

The quality of the aforementioned datasets was not significantly affected by the sensor position, because either the sensor was static or the measurement did not require perfect sensor localization. However, metric-map-building algorithms require precise localization of the sensor. While localization can be solved by means of known SLAM methods, the position estimate is never absolutely error-proof or perfectly accurate. Therefore, localization glitches and inherent sensor noise cause degradation of metric maps over time. This degradation can be countered by means of Bayesian filtering methods. Since our approach is not explicitly designed to suppress sensor noise, one might assume that it would not deal with the aforementioned issues.

However, from the theoretical point of view, taking into account only the most prominent spectral components in the update step (see section II-B) is equivalent to sensor data filtering. Thus, map errors caused by noisy measurements should fade out similarly as with the classical mapping approaches. To emphasize new measurements, the update step introduced in Section II-B might be performed for a limited slice of the data gathered causing the model to “forget”

observations that were made too long time ago. However, such a forgetting scheme is beyond the scope of this paper. To

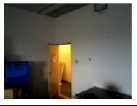



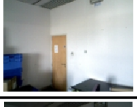
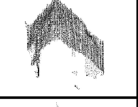


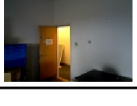



	RGB Image	Point Cloud	Measured Grid	Estimated Grid
Day 1				
Day 2				
Day 6				

Fig. 7. Reconstructed and real environment state at different times.

demonstrate that our representation can deal with the noisy measurements of the environment states $s_j(t)$, we used the 3D and 2D measurements to build 3D and 2D occupancy grids, see Section III. To suppress the effects of imprecise localization, the point clouds gathered by the range sensors were aligned by means of the ICP algorithm prior to the occupancy grid calculation. The individual grid maps were aligned over each other. Therefore we were able to track how the occupancy of the individual grid cells changes over time. Since the cells are considered independent, each cell of the aforementioned grids maintains its own spectral model built from approximately 2000 measurements gathered regularly during one week. Figure 7 provides a detail of the 3D occupancy grid at location 2 of the ‘Lab’ dataset, showing the estimated probabilities and measured states of individual cells at different times.

The accuracy of reconstruction of each cell’s spectral model was calculated, see Table I. The results indicate that the accuracy and compression ratio of the spectral model was not significantly affected by the imperfections of the robot localization.

V. CONCLUSION

A novel approach for spatiotemporal mapping in the context of mobile robotics has been presented. The approach is based on an assumption that in a mid-term perspective, the environment is influenced by several processes which might be periodical and that the evolution of the environment can be described by the periodicity, amplitude and time shift of these underlying processes. To identify the parameters of these processes and to predict the environment’s local state we use the direct, respectively inverse Fourier transform. The core of the proposed temporal representation is composed of the most prominent frequency components of the Fourier spectrum – these relate to the most important periodical processes influencing the environment.

To evaluate the performance of the proposed method in a real scenario, we have applied it to data gathered by a mobile robot continuously patrolling in an indoor environment for a

period of one week while detecting people and building 2- and 3-D occupancy grids of designated locations.

The results indicate that the proposed method allows to represent arbitrary timescales with constant (and low) memory requirements, achieving compression rates up to 10^6 . Moreover, we demonstrate that the representation allows prediction of future states of the environment with accuracies ranging from 88% to 99%, which enables easy detection of unexpected (anomalous) environment states.

In the future, we will study the utility of the spectral model for mobile robot localization, path planning, information-driven spatio-temporal exploration and semantic segmentation. We will examine the method’s accuracy in long-term scenarios and extend the spectral representation by modelling the uncertainty of its long-term predictions.

ACKNOWLEDGMENTS

The work has been supported by the EU ICT project 600623 ‘STRANDS’. We thank Miroslav Kulich and Pablo Urcola for valuable remarks regarding probability theory.

REFERENCES

- [1] S. Thrun, “Robotic mapping: A survey,” *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2002.
- [2] D. Hähnel, D. Schulz, and W. Burgard, “Mobile robot mapping in populated environments,” *Advanced Robotics*, 2003.
- [3] D. Wolf and G. Sukhatme, “Mobile robot simultaneous localization and mapping in dynamic environments,” *Autonomous Robots*, 2005.
- [4] L. Montesano *et al.*, “Modeling dynamic scenarios for local sensor-based motion planning,” *Autonomous Robots*, 2008.
- [5] C. Wang *et al.*, “Simultaneous localization, mapping and moving object tracking,” *International Journal of Robotics Research*, 2007.
- [6] D. Migliore *et al.*, “Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments,” in *ICRA Workshop on Safe navigation in dynamic environments*, 2009.
- [7] P. Biber and T. Duckett, “Experimental analysis of sample-based maps for long-term slam,” *International Journal of Robotics Research*, 2009.
- [8] M. Milford and G. Wyeth, “Persistent navigation and mapping using a biologically inspired SLAM system,” *The International Journal of Robotics Research*, vol. 29, no. 9, pp. 1131–1153, 2010.
- [9] K. Konolige and J. Bowman, “Towards lifelong visual maps,” in *International Conference on Intelligent Robots and Systems*, 2009.
- [10] S. Hochdorfer and C. Schlegel, “Towards a robust visual SLAM approach,” in *International Conference on Advanced Robotics*, 2009.
- [11] F. Dayoub, G. Cielniak, and T. Duckett, “Long-term experiments with an adaptive spherical view representation for navigation in changing environments,” *Robotics and Autonomous Systems*, 2011.
- [12] C. Stachniss and W. Burgard, “Mobile robot mapping and localization in non-static environments,” in *Conf. on Artificial Intelligence*, 2005.
- [13] N. Mitsou and C. Tzafestas, “Temporal occupancy grid for mobile robot dynamic environment mapping,” in *Mediterranean Conference on Control Automation*, 2007.
- [14] M. Yguel *et al.*, “Wavelet occupancy grids: a method for compact map building,” in *Field and Service Robotics*, 2006.
- [15] T. Kucner *et al.*, “Conditional transition maps: Learning motion patterns in dynamic environments,” in *Int. Robots and Systems*, 2013.
- [16] M. Pfingsthorn *et al.*, “Maximum likelihood mapping with spectral image registration,” in *Int. Conf. on Robotics and Automation*, 2010.
- [17] A. Makadia *et al.*, “Fully automatic registration of 3d point clouds,” in *Computer Vision and Pattern Recognition*, 2006.
- [18] P. Neubert *et al.*, “Appearance change prediction for long-term navigation across seasons,” in *European Conf. on Mobile Robotics*, 2013.
- [19] U. Frese and L. Schroder, “Closing a million-landmarks loop,” in *Intelligent Robots and Systems*, 2006.
- [20] R. N. Bracewell and R. Bracewell, *The Fourier transform and its applications*. McGraw-Hill New York, 1986, vol. 31999.
- [21] T. Krajník and M. Nitsche, “Visual Localization System for Mobile Robotics,” in *International Conference on Advanced Robotics*, 2013.
- [22] “Linda’s status.” [Online]. Available: <http://twitter.com/LindaStrands>

E

KEY ARTICLE [22] - ICRA WORKSHOP ON LONG-TERM
AUTONOMY 2016

©[2016] Reprinted, with permission, from Tomáš Krajník, Frequency Map Enhancement:
Introducing Dynamics into Static Environment Models, 2016.

Frequency Map Enhancement: Introducing Dynamics into Static Environment Models

Tomáš Krajník

Jaime Pulido Fentanes

João Santos

Tom Duckett

Abstract—We present applications of the Frequency Map Enhancement (FreME_n), which improves the performance of mobile robots in long-term scenarios by introducing the notion of dynamics into their (originally static) environment models. Rather than using a fixed probability value, the method models the uncertainty of the elementary environment states by their frequency spectra. This allows to integrate sparse and irregular observations obtained during long-term deployments of mobile robots into memory-efficient spatio-temporal models that reflect mid- and long-term pseudo-periodic environment variations. The frequency-enhanced spatio-temporal models allow to predict the future environment states, which improves the efficiency of mobile robot operation in changing environments. In a series of experiments performed over periods of weeks to years, we demonstrate that the proposed approach improves mobile robot localization, path and task planning, activity recognition and allows for life-long spatio-temporal exploration.

I. INTRODUCTION

As robots gradually enter human-populated environments, they have to deal with the fact that the environments are uncertain because they change over time. While the probabilistic mapping methods used in robotics can handle uncertain and incomplete environment knowledge, their theoretical foundations assume that the uncertainty is caused by sensor noise rather than by natural processes that govern the environment changes. The assumption of a static world negatively impacts the ability of these models to reflect the environment changes and effectively support long-term autonomous operation of mobile robots. However, several studies [1], [2], [3], [4], [5] indicated that explicit modeling of the environment changes improves localization robustness.

Biber and Duckett [5] proposed to represent the world dynamics by multiple maps with different timescales, which are switched on the fly based on their consistency with the current observations. Dayoub et al. [6] present a system that evaluates the persistence of visual features over time in order to identify features that are more likely to be stable. Churchill and Newman [1] demonstrated that clustering spatially-close observations into ‘experiences’ improves long-term localization. The article [3] associates each cell of an occupancy grid with a hidden Markov model, which improves the localization robustness as well. Kucner’s method [7] assumes that occupancies of grid cells are influenced by a moving objects, which allows to infer typical motion patterns in a given environment. Sünderhauf’s method [4] proposes to learn typical appearance changes caused by seasonal factors

and use this knowledge for long-term predictions of environment appearance. Finally, Rosen et al. [8] use Bayesian-based survivability analysis to predict which landmarks will be visible after some time and which are going to disappear.

Our approach to environment change modeling is based on the assumption that some of the mid- to long-term processes that cause the environment changes are (pseudo-)periodic, e.g. seasonal foliage variations, daily illumination cycle or routine human activities. To reflect this assumption, we represent the probability of each local environment state not by a single value, but by a probabilistic function of time composed of several harmonic functions whose periodicities and amplitudes relate to the frequencies and influences of these hidden processes.

II. METHOD DESCRIPTION

The proposed method, coined the Frequency Map Enhancement (FreME_n), represents the probability of each environment state by a function of time

$$p(t) = p_0 + \sum_{j=1}^n p_j \cos(\omega_j t + \varphi_j), \quad (1)$$

where n is the number of environment processes taken into account, ω_j , φ_j and p_j relate to the frequencies, time offsets and influences of these processes, and p_0 is the mean probability of the state. To obtain the parameters ω_j , φ_j and p_j , we first obtain the frequency spectra $S(\omega)$ of long-term observations of each environment state $s(t)$ by means of a (non-uniform) Fourier transform [9], i.e. $S(\omega) = \mathcal{FT}(s(t))$. The parameters ω_j , φ_j and p_j in Equation (1) are equal to the amplitudes, phases and frequencies of the n most prominent spectral components of the spectrum $S(\omega)$. To deal with the fact that robots might observe the environment on an irregular basis, we employ a non-uniform Fourier Transform scheme [9] similar to the one used in [10].

The approach, which was originally presented in [11], can be applied to all environment models that represent the world as a set of independent component with binary states. In this short overview, we will show its use does not improve only long-term localisation in changing environments [2], [12], but that it also improves robotic search [13], path planning [14] and activity recognition. We will also show that the time-dependent probability of the environment states expressed by Equation (1) allows the calculation of the spatio-temporal environment entropy, which, combined with information-theoretic planning, results in life-long spatio-temporal exploration of dynamic environments [9], [15].

III. VISUAL LOCALIZATION

The problem of visual place recognition in changing environments has received considerable attention during recent years [16]. We propose to represent the variations in appearance of different locations by modeling the visibility of individual image features in the frequency domain. Thus, we can predict which visual features are going to be visible at which time and use these time-specific features to localise the robot [2]. To evaluate our approach, we performed both indoors and outdoors experiments. The indoor experiment was performed at the Lincoln Centre for Autonomous Systems, where a SCITOS-G5 robot captured images of 8 areas every 10 minutes for one week and used the models created to localize itself after one week, three months and one year [2]. The outdoor experiment was performed in the Stromovka park in Prague, where a P3AT robot captured images of designated places on a monthly basis for one year and used the FreMEn feature map to determine its location during three testing runs during the following year. While the indoor dataset was mainly affected by the daily illumination cycle and human activities, the outdoor dataset captured seasonal variations of foliage.

The dependence of the localization error on the number of features used is shown on Figure 2, which indicates that the frequency-enhanced models that generate a set of likely-to-be-visible features for a particular time outperform the ‘static’ approach that relies on the most stable features, i.e. modelling the appearance variations by our approach improves the robustness of localization.

IV. CONTINUOUS LASER-BASED LOCALIZATION

While the advantages of using dynamic representations for visual localisation are clear, because the environment appearance variations are significant due to the passive nature of the cameras, the usefulness of dynamic maps for laser-based, 2d localisation was demonstrated only in highly dynamic environments, such as parking lots [3]. However, maps are not useful only for localization, but also for planning. Thus, a mobile robot that operates in the long-term should be able to reason about the nature of the environment changes it encountered during its deployment: knowing that obstacles which were blocking a corridor a hours ago are likely to be gone by now or that during noon, the cafeteria is too crowded, is beneficial when planning the robot’s path.

However, the major part of the changes observed in 2D occupancy grids build by lasers are not periodic. Typically, they are caused by temporary stationary objects that tend to disappear after some time. Since modeling periodic changes alone is not sufficient, we combined FreMEn with the approach presented in [3] and created a 2D occupancy grid that models both the periodicity and stationarity of the changes [12]. This FreMEn 2D occupancy grid was integrated into the ROS navigation stack of our SCITOS-G5 robot, which operated in a populated open plan office for 2 weeks. During these two weeks, we observed that the efficiency of the robot navigation gradually improved

(e.g. path re-planning was triggered less often), while the reduction in the localization error was only marginal [12].

V. TOPOLOGICAL PATH PLANNING

Imagine a robot operating in an office like environment 24/7, performing different user-defined tasks at different locations. The robot needs to schedule not only these tasks, but also has to determine when to visit its charging station. To create the schedule, the robot needs to predict which areas of the environment are going to be accessible at which times.

To address this problem, we represented the environment as a topological map, where the traversability of individual edges was modelled by FreMEn [14], see Figure 4. Using this topological map, the robot can not only plan its path, but it can also determine what is the chance of the path’s successful traversal, i.e. the chance of reaching a given destination at a particular time. To evaluate our approach, we let our SCITOS-G5 robot operate in an populated office-like environment for more than 10 weeks, during which the robot learned that two edges of the topological map exhibit periodic changes to their traversability. The first edge corresponded to a laboratory safety door that was kept closed at night and the second edge led through a narrow area behind lecturer’s desks, which was occasionally blocked by chairs. Using its knowledge about the dynamics of these edges, the robot could infer that the best time to perform its activities in the office areas was afternoon during the weekdays, because at other times, it would risk that after completing these tasks, it would not be able to return to the laboratory and reach its charging station, see the map in Figure 4.

VI. ROBOT SEARCH

Another combination of topological representations with FreMEn was used to model object and people presence in a robotic search scenario. Here, a mobile robot has to find a certain object or person as quickly as possible. For the sake of simplicity, we assume that the object or person is detected as soon as the robot arrives at its location.

To plan an efficient search path through the individual locations, the robot needs to take into account the probability of object occurrence. Improved knowledge about possible object location leads to more efficient plans and hence, shorter times to locate the desired object. We formulated the search as a path planning problem in a graph where the probability of object occurrences at particular nodes is a function of time represented by FreMEn [13]. To evaluate our approach, we used three datasets collected over several months containing person and object occurrences in residential and office environments. Several types of spatio-temporal models were created for each of these datasets and the efficiency of the search method was assessed by measuring the time it took to locate a particular object. The experimental results indicated that representing the dynamics of object occurrences by FreMEn reduced the search time by 25% to 65% compared to maps that consider the probability of object occurrences as independent of time.

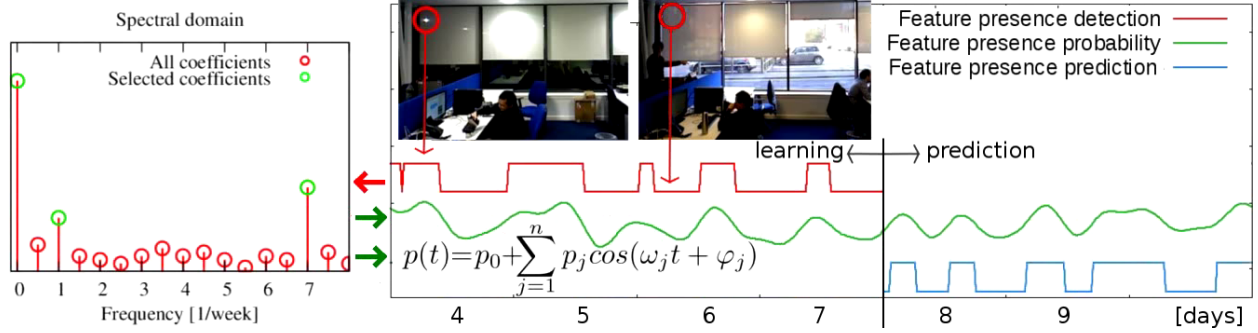


Fig. 1: Frequency-enhanced feature map [2] for visual localization: The observations of image feature visibility (centre, red) are transferred to the spectral domain (left). The most prominent components of the model (left, green) constitute an analytic expression (centre, bottom) that represents the probability of the feature being visible at a given time (green). This allows to predict the feature visibility at a time when the robot performs self-localization (blue).

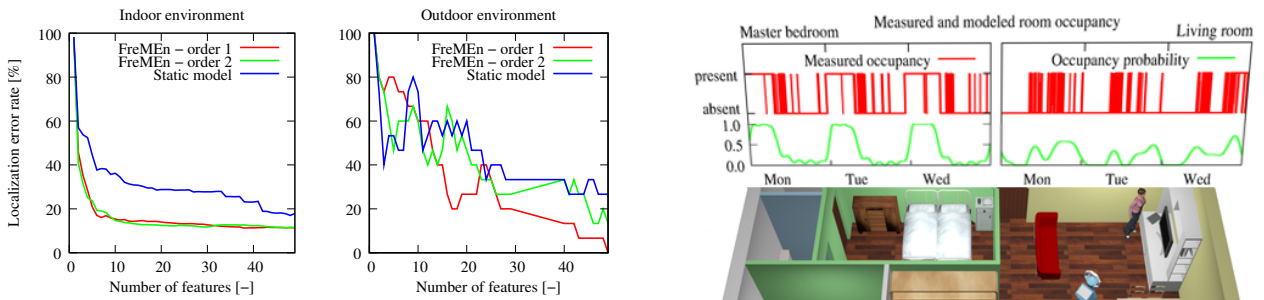


Fig. 2: The dependency of localization error on the number of features used.



Environment change example (cubboard doors open/closed)

Fig. 3: Example of the regular variations and corresponding map sections predicted for morning (left) and evening (right).

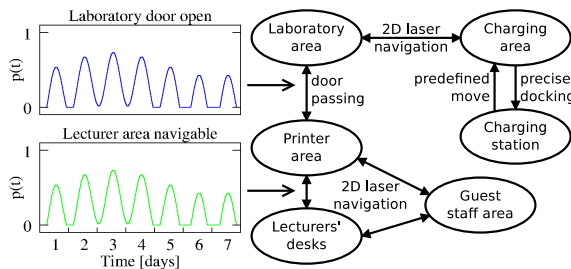


Fig. 4: Partial map of the operational environment with temporal edge traversability models. Nodes represent locations and edges movement actions which can fail, e.g. *door passing* requires an open door. The (illustrative) graphs show the predicted probability of successful edge traversal.

VII. LIFE-LONG EXPLORATION

In the previous scenarios, the map was created from sensory data which were gathered when the robot was

Fig. 5: Aruba ‘CASAS’ [17] apartment with probabilities of person presence in two rooms.

performing user-motivated tasks. This passive approach to mapping is not only slow, but it typically results in an incomplete knowledge that might lead to misinterpretations of the processes that govern the environment changes. To deal with this, the robot has to explore its environment in an active way, which does not only mean that it should visit all the relevant locations at least once, but it should also revisit them to understand how they change over time.

The crucial issue is to determine where and when to perform observations in order to refine and complete the spatio-temporal model. Since the FreMEn model predicts the probability of the environment states, we use it to calculate the states’ entropy, which directly corresponds to the amount of information obtained by observing these states at a particular time. Application of information-based exploration methods to the spatio-temporal entropy predicted by FreMEn resulted in intelligent and continuously improving exploratory behaviour, which evolves as the environment knowledge becomes more refined over time [9], [15].

Figure 6 illustrates the exploratory behaviour on the Aruba [17] dataset, where the robot created a spatio-temporal model of person presence used for the robot search [13]. During the first day, the robot has no knowledge of the

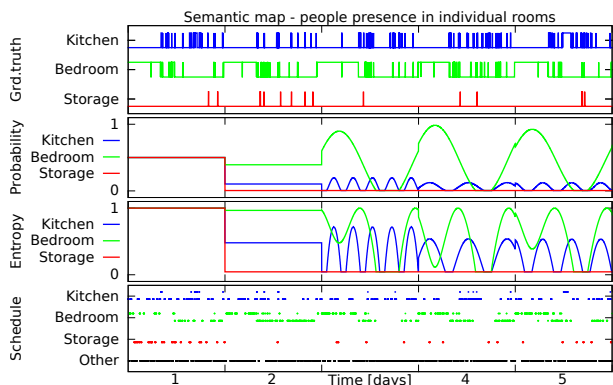


Fig. 6: Spatio-temporal exploration behaviour: The robot uses its probabilistic world model (second row) and spatio-temporal entropy estimates (third row) to schedule its observations (bottom graph) and learn the environment dynamics (top). As the environment knowledge improves over time, the scheduled observations provide more information which allows for further refinement of the environment model.

environment and it has no room or time preference when scheduling its observations. After the first day, it schedules more visits of the rooms where the person presence changes more often. The second day observations provide information about the rooms’ dynamics: the robot assumes that the bedroom has a daily periodicity and that the kitchen is visited five times per day. This causes the expected information gain to be time-dependent – e.g. evening and morning observations of the bedroom provide more information than in the afternoon, which is reflected by the exploration schedule, see the last row of Figure 6.

VIII. ACTIVITY RECOGNITION

Using the office and household datasets from Sections III and VI respectively, we also tested the use of the FrEMEn as a model providing temporal-based priors for human activity recognition. The FrEMEn models were created incrementally by a Bayesian update scheme, which was performed every time an activity was recognized. The method gradually learned about the typical rhythms of the people’s activities, effectively reducing the error in activity classification, see Figure 7 and article [18].

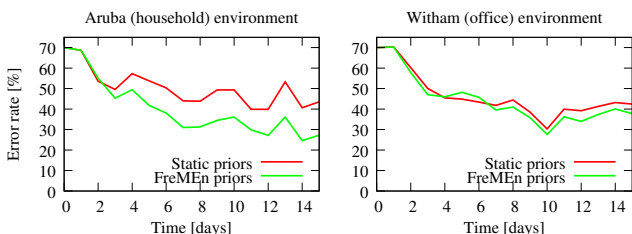


Fig. 7: Activity recognition error over time

IX. CONCLUSION

We presented applications of a method that improves the efficiency of long-term mobile robot operation in chang-

ing environments. The method assumes that in a mid-term perspective, the environment is influenced by processes which might be periodical and that the evolution of the some environment states can be described by the periodicity, amplitude and time shift of these underlying processes. To identify the parameters of these processes and to predict the environment’s local state we use techniques based on the Fourier transform. We gave an overview of the methods’ applications so far, showing that in long-term scenarios, it reduces localisation error [2], [12], speeds-up robotic search [13], improves path planning [14], activity recognition [18] and allows for active, life-long environment exploration [9], [15]. To facilitate the use of the method by other researchers, we published its ROS-compatible source code at <http://fremen.uk>.

REFERENCES

- [1] W. S. Churchill and P. Newman, “Experience-based navigation for long-term localisation,” *IJRR*, 2013.
- [2] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, “Long-term topological localization for service robots in dynamic environments using spectral maps,” in *IROS*, 2014.
- [3] G. D. Tipaldi *et al.*, “Lifelong localization in changing environments,” *The International Journal of Robotics Research*, 2013.
- [4] P. Neubert, N. Sünderhauf, and P. Protzel, “Superpixel-based appearance change prediction for long-term navigation across seasons,” *Robotics and Autonomous Systems*, 2014.
- [5] P. Biber and T. Duckett, “Dynamic maps for long-term operation of mobile service robots,” in *Robotics: Science and Systems*, 2005.
- [6] F. Dayoub, G. Cielniak, and T. Duckett, “Long-term experiments with an adaptive spherical view representation for navigation in changing environments,” *Robotics and Autonomous Systems*, 2011.
- [7] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, “Conditional transition maps: Learning motion patterns in dynamic environments,” in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [8] D. M. Rosen, J. Mason, and J. J. Leonard, “Towards lifelong feature-based mapping in semi-static environments,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, to appear.
- [9] T. Krajník, J. Santos, and T. Duckett, “Life-long spatio-temporal exploration of dynamic environments,” in *ECMR*, 2015.
- [10] S. Ferraz-Mello, “Estimation of periods from unequally spaced observations,” *The Astronomical Journal*, vol. 86, p. 619, 1981.
- [11] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, “Spectral analysis for long-term robotic mapping,” in *ICRA*, 2014.
- [12] T. Krajník, J. P. Fentanes, M. Hanheide, and T. Duckett, “Persistent Localization and Life-long Mapping in Changing Environments using the Frequency Map Enhancement,” in *International Conference on Intelligent Robots and Systems (IROS)*, 2016, in review.
- [13] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, and T. Duckett, “Where’s waldo at time t? using spatio-temporal models for mobile robot search,” in *ICRA*, 2015.
- [14] J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide, “Now or later? predicting and maximising success of navigation actions from long-term experience,” in *ICRA*, 2015.
- [15] J. M. Santos, T. Krajník, J. Pulido Fentanes, and T. Duckett, “Lifelong information-driven exploration to complete and refine 4d spatio-temporal maps,” *Robotics and Automation Letters*, 2016.
- [16] S. Lowry, N. Sünderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford, “Visual place recognition: A survey,” *IEEE Transactions on Robotics*, vol. PP, no. 99, pp. 1–19, 2015.
- [17] D. J. Cook, “Learning setting-generalized activity models for smart spaces,” *IEEE Intelligent Systems*, no. 99, p. 1, 2010.
- [18] C. Coppola, T. Krajník, N. Bellotto, and T. Duckett, “Learning temporal context for activity recognition,” in *European Conference on Artificial Intelligence (ECAI)*, 2016, in review.

F

KEY ARTICLE [24] - IEEE TRANSACTIONS ON ROBOTICS 2017

©[2017] IEEE. Reprinted, with permission, from Tomáš Krajník, FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments, 2017.

FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments

Tomáš Krajník, Jaime P. Fentanes, João M. Santos, and Tom Duckett

Abstract—We present a new approach to long-term mobile robot mapping in dynamic indoor environments. Unlike traditional world models that are tailored to represent static scenes, our approach explicitly models environmental dynamics. We assume that some of the hidden processes that influence the dynamic environment states are periodic and model the uncertainty of the estimated state variables by their frequency spectra. The spectral model can represent arbitrary timescales of environment dynamics with low memory requirements. Transformation of the spectral model to the time domain allows for the prediction of the future environment states, which improves the robot’s long-term performance in changing environments. Experiments performed over time periods of months to years demonstrate that the approach can efficiently represent large numbers of observations and reliably predict future environment states. The experiments indicate that the model’s predictive capabilities improve mobile robot localization and navigation in changing environments.

Index Terms—Localization, long-term autonomy, mapping.

I. INTRODUCTION

ADVANCES in the field of mobile robotics are gradually enabling long-term deployment of autonomous robots in human environments. As these environments change over time, the robots have to deal with the fact that their world knowledge is incomplete and uncertain. Although probabilistic mapping methods [1] have demonstrated the ability to represent incomplete knowledge about the environment, they generally assume that the corresponding uncertainty is caused by inherent sensor noise rather than by natural processes that cause the environment to change over time. Thus, traditional mapping methods treat measurements of dynamic environment states as outliers [2]. This undermines the ability of the mapping methods to reflect

Manuscript received June 5, 2016; revised December 10, 2016; accepted January 9, 2017. Date of publication March 14, 2017; date of current version August 7, 2017. This paper was recommended for publication by Associate Editor M. Kaess and Editor C. Torras upon evaluation of the reviewers’ comments. This work was supported by the EU ICT Project 600623 “STRANDS” and the Czech Science Foundation under Project 17-27006Y.

T. Krajník is with the Lincoln Centre for Autonomous Systems, University of Lincoln, Lincoln LN6 7TS, U.K., and also with Faculty of Electrical Engineering, Czech Technical University, Prague 16636, Czechia (e-mail: tkrajnik@lincoln.ac.uk).

J. P. Fentanes, J. M. Santos, and T. Duckett are with the Lincoln Centre for Autonomous Systems, University of Lincoln, Lincoln LN6 7TS, U.K. (e-mail: jpulidofentanes@lincoln.ac.uk; jsantos@lincoln.ac.uk; tduckett@lincoln.ac.uk).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TRO.2017.2665664

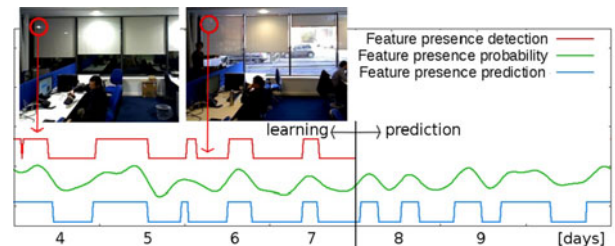


Fig. 1. Frequency-enhanced model of a single image feature visibility. The observations of image feature visibility (red) are processed by the FreMEn method that extracts the time-dependent probability of the feature being visible (green). This allows us to reconstruct and predict the feature’s visibility for a given time (blue).

the environment dynamics and provide support for long-term mobile robot autonomy. Recent works have demonstrated that exploiting the outlying measurements allows us to characterize some environment changes, which improves robot localization in changing environments [3]–[6].

In our approach, we assume that some of the mid- to long-term processes that exhibit themselves through environment changes are periodic. These processes can be both natural, e.g., seasonal foliage changes, or artificial, e.g., human activities characterized by regular routines. Regardless of the primary cause of these processes, we hypothesize that the regularity of the environment changes can be exploited by robots to build more robust representations of their surroundings. We propose to represent the probability of the elementary environment states by combination of harmonic functions whose amplitudes and periodicities relate to the influences and frequencies of the hidden processes that cause the environment variations. This allows for efficient representation of the spatiotemporal dynamics as well as prediction of future environment states. To obtain the parameters of the harmonic functions, we propose to treat the long-term observations of the environment states as signals, which can be analyzed in the frequency domain.

An advantage of our approach is its universal applicability—it can introduce dynamics to any stationary environment model that represents the world as a set of independent components. Introduction of the dynamics is achieved simply by representing the uncertainty of the elementary states as probabilistic functions of time instead of constants that are updated only when the given state is observed by a robot. The approach, which was originally introduced in [7], was successfully applied to

landmark maps to improve localization [4] and to topological maps to improve the robotic search [8] and navigation [46]. The application of the method to occupancy grids not only reduces memory requirements [9], but also enables lifelong spatiotemporal exploration [10], [47] of changing environments. In this paper, we summarize and extend the previous results by a thorough examination of the method's ability to efficiently represent environment changes over long time periods, predict the future environment states, and use these predictions to improve the robustness of robot localization and navigation. While the main aim of our method is to deal with periodic changes, we also show that its combination with a persistence model allows us to learn and deal with nonperiodic dynamics as well.

II. RELATED WORK

While the mapping of stationary environments has been widely studied [11] and generating large-scale stationary environment models has been in the spotlight of robotics research for a long time, mapping methods that explicitly model the environment dynamics gained importance only after robots attained the ability of autonomously operating for longer time periods [44].

The first approaches to address the problem of dynamic environments were object-centric. These methods identify moving objects and remove them from the environment representations [12], [13] or use them as moving landmarks [14], [15] for self-localization. But not all dynamic objects actually move at the moment of mapping, which means that their identification requires long-term observations. To tackle this issue, Ambrus *et al.* [16] proposed to process several 3-D point clouds of the same environment obtained over a period of several weeks to identify and separate the movable objects and refine the static environment structure at the same time. While object-centric representations can handle some problems of dynamic mapping, they still assume that most of the environment is static, which makes them unsuitable for scenarios where the environment varies significantly.

Considering this aspect, other authors propose approaches that assume the map to never be complete and perform mapping in a continuous manner, adding new features to the map every time the robot observes its environment. In these approaches, managing map size is crucial [17]–[20].

Alternatively, some authors propose systems that learn a fixed set of possible states for the dynamic objects, e.g., corresponding to open and closed doors [21], [22], which can limit the map size, but this approach is limited in the real scenarios, where the number of states is unpredictable.

Other approaches do not attempt to explicitly identify movable objects, but rely on less abstract environment representations. For example, Biber and Duckett [17] and Arbutle *et al.* [23] represent the environment dynamics by multiple temporal models with different timescales where the best map for localization is chosen by its consistency with current readings. Dayoub *et al.* [24] and Rosen *et al.* [25] each present a feature persistence system based on temporal stability in sparse visual maps that can identify environmental features which are more likely to be stable. Yguel *et al.* [26] propose to model occupancy

grid maps in the wavelet space in order to optimize the amount of information that has to be processed for path planning.

Churchill and Newman [3] propose to integrate similar observations at the same spatial locations into “experiences” which are then associated with a given place. They show that associating each location with multiple “experiences” improves autonomous vehicle localization. Tipaldi *et al.* [5] represent the states of the environment components (cells of an occupancy grid) with a hidden Markov model and show that their representation improves localization robustness. Kucner *et al.* [27] learn conditional probabilities of neighboring cells in an occupancy grid to model typical motion patterns in dynamic environments. Another method can learn appearance changes based on a cross-seasonal dataset and use the learned model to predict the environment appearance [6] showing that state prediction can be useful for long-term place recognition in changing environments. Finally, Krajník *et al.* [7] represent the environment dynamics in the spectral domain and apply this approach to image features to improve the localization [4] and to occupancy grids to reduce memory requirements [9].

While most of the aforementioned methods are aimed specifically at the problem of lifelong localization, our approach was shown to be applicable in other scenarios as well [28]. In this paper, we extend the results and experimental analysis presented in [4], [7], and [9]. The efficiency of spatiotemporal representation, which was only briefly mentioned in [9], is now thoroughly investigated on a FreMEN 4D (3D+time) occupancy grid, which represents almost 2 million observations of a small office over 112 days. Compared to the work presented in [4], which provides only a coarse evaluation compared to a naïve localization method, the experiments in this paper demonstrate how the localization robustness depends on the number of predicted visual landmarks, compare its performance to the experience-based approach [3], and present additional evaluation on outdoor datasets. This paper also demonstrates that integration of the method in the ROS navigation stack improves both the accuracy of robot localization and the efficiency of navigation.

III. SPECTRAL REPRESENTATION FOR SPATIOTEMPORAL ENVIRONMENT MODELS

Many environment models used in mobile robotics consist of independent components that can be in two distinct states. For example, the cells of an occupancy grid are occupied or free, edges of a topological map are traversable or not, doors are open or closed, rooms are vacant or occupied, landmarks are visible or occluded, etc. The states of the real world cannot be observed directly, but through sensors that are affected by noise. Thus, the state of each world model component is uncertain, which is typically represented by the probability of a particular component being in a given state, e.g., the uncertainty of occupancy of the j th cell is typically represented by $p_j = P(s_j = \text{occupied})$. This allows us to counter the effect of noisy measurements by employing statistical methods, such as Bayesian filtering [1]. However, the mathematical foundations described in [1] assume a static world, i.e., the probabilities of

the world components are assumed to be constant. While this still allows us to update the environment model if a change has been observed for long enough, the old states are simply “forgotten” over time and the system does not learn from the change observed.

We propose to represent the uncertainty of the environment states not as probabilities p_j , but as probabilistic functions of time $p_j(t)$. Assuming that the variations of the environment are caused by a number of unknown processes, some of which exhibit periodic patterns, the $p_j(t)$ can be represented by a combination of harmonic functions that relate to these periodic processes. To identify the parameters of these harmonic functions, we propose to use spectral analysis methods, namely the Fourier transform [29].

A. Fourier Transform

The Fourier Transform is a well-established mathematical tool widely used in the field of statistical signal processing. In a typical case, it transforms a function of time $f(t)$ into a function of frequency $F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-j\omega t} dt$. The function $F(\omega)$ is commonly referred to as the frequency spectrum of $f(t)$. The Fourier transform is invertible, and therefore, one can recover the function $f(t)$ from its spectrum $F(\omega)$, i.e., $f(t) = \mathcal{F}^{-1}(F(\omega))$. If one wants to analyze or alter the periodic properties of a process characterized by a function $f(t)$, it is reasonable to calculate its spectrum $F(\omega)$, perform the analysis or alteration in the frequency domain, and then transform the altered spectrum $F'(\omega)$ back to the temporal domain. Such a process is referred to as spectral analysis.

Typically, $F(\omega)$ is a complex-valued function, whose absolute values and arguments correspond to the amplitudes and phase shifts of the frequency components ω . Given that $f(t)$ is a real periodic discrete function, the spectrum $F(\omega)$ can be represented by a finite set of complex numbers.

B. Proposed Representation

Although the approach can be applied to most state-of-the-art representations, we will explain it with an occupancy grid. To keep this explanation simple, we assume that the occupancy of the individual cells is independent of each other and explain the approach on a single cell. So let us assume that at a given time t , a single cell of an occupancy grid is either *occupied* or *free*. Let us represent the state as a binary function of time $s(t) \in \{0, 1\}$, where $s(t) = 0$ corresponds to the cell being *free* at time t and vice versa.

The main idea behind the proposed model is to treat the values of the function $s(t)$ as real numbers and calculate the Frequency spectrum of the sequence $s(t)$ by means of a (Discrete) Fourier transform as

$$S(\omega) = \mathcal{F}(s(t)). \quad (1)$$

The resulting frequency spectrum $S(\omega)$ is a discrete complex function whose absolute values $|S(\omega)|$ correspond to the influences of periodic processes on $s(t)$. In other words, each local maximum of $|S(\omega)|$ indicates that the function $s(t)$ might be influenced by a hidden process whose period is $T = 2\pi/\omega$. Since

we do not want to represent the state $s(t)$ directly, but as a combination of l periodic processes, we select the l most prominent (i.e., of highest absolute value) coefficients of the spectrum $\mathcal{S}(\omega)$ and store them along with their frequencies ω_i in a set $\mathcal{P}(\omega)$. The coefficients stored in the set \mathcal{P} are then used to recover a function $p(t)$ by means of the inverse Fourier transform

$$p(t) = \varsigma(\mathcal{F}^{-1}(\mathcal{P}(\omega))) \quad (2)$$

where ς denotes a function that ensures that $p(t) \in [0, 1]$. For our purposes, we choose a simple saturation function $\varsigma(x) = \min(\max(x, 0), 1)$, which achieved better results than other normalization schemes in our experiments.

Now, let us assume that

$$\begin{aligned} P(s(t) = 1) &= p(t) \\ P(s(t) = 0) &= 1 - p(t). \end{aligned} \quad (3)$$

The ς function ensures that both $1 - p(t)$ and $p(t)$ are always positive, i.e.,

$$P(s(t)) \geq 0 \quad (4)$$

for all possible states $s(t)$. The cell is always either *free* or *occupied*, i.e., the state $s(t)$ is always either 0 or 1, meaning that

$$P(\{s(t) = 0\} \cup \{s(t) = 1\}) = 1. \quad (5)$$

Finally, the sum of all $P(s(t))$ for all $s(t) \in \{0, 1\}$ is

$$P(\{s(t) = 1\}) + P(\{s(t) = 0\}) = 1 - p(t) + p(t) = 1. \quad (6)$$

Since $P(s(t))$ satisfies (4)–(6), which are Kolmogorov’s axioms, we can assume that $P(s(t))$ is a probability. Thus, the function $p(t)$ recovered from the frequency spectrum of $s(t)$ by (2) represents the probability that the cell is *occupied* at time t .

By thresholding the probability $p(t)$, we can calculate an estimate $s'(t)$ of the original state $s(t)$. However, the original observation of $s(t)$ can differ from the probabilistic estimate $s'(t)$. In the case that the given application has to preserve all past observations correctly, the differences between $s'(t)$ and $s(t)$ are stored in an outlier set \mathcal{O} .

Thus, our model of the state consists of two finite sets \mathcal{P} and \mathcal{O} . The set \mathcal{P} consists of l triples $abs(P_i)$, $arg(P_i)$, and ω_i , which describe the amplitudes, phase shifts, and frequencies of the model spectrum. Each such triple is related to the importance, time offset, and periodicity of one particular periodic process influencing the state $s(t)$. We will refer to the number of modeled processes l (i.e., to the number of triples in \mathcal{P}) as the “order” of the spectral model. The set \mathcal{O} represents a set of k time intervals, during which the state $s(t)$ did not match the state $s'(t)$ calculated from $p(t)$. To achieve low memory requirements, the set \mathcal{O} is Δ -encoded, i.e., it is implemented as a sequence of values, indicating the starts and ends of time intervals when the predicted and observed state did not match, i.e., $s'(t) \neq s(t)$. Thus, each such interval is represented by its limits $[t_{2k}, t_{2k+1})$.

Fig. 2 provides a graphic representation of the model building process and a commented video is available at [30]. The process starts with the measured state $s(t)$ (red line, left box), which is transformed into the frequency domain $S(\omega)$ (right top, red).

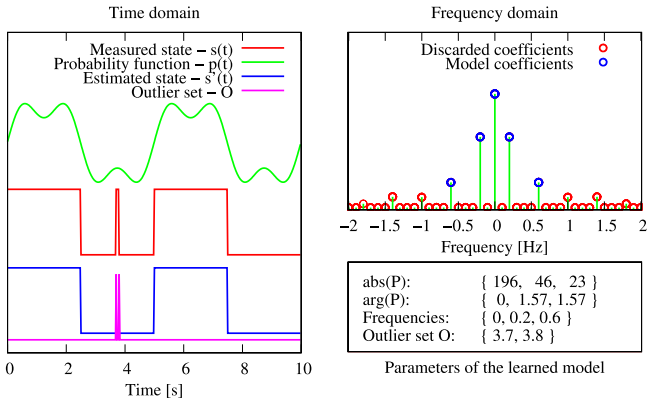


Fig. 2. Example of the measured state and its spectral model. The left part shows the time series of the measured state $s(t)$, probability estimate $p(t)$, predicted state $s'(t)$, and outlier set \mathcal{O} . The upper right part shows the absolute values of the frequency spectrum of $s(t)$ and indicates the spectral coefficients, which are included in the model, i.e., in the set \mathcal{P} . The spectrum is symmetric and the spectral coefficient with frequency 0 corresponds to mean probability of $s(t) = 1$. Thus, the model encodes two periodic nodes—its order l is 2.

The most relevant spectral components $\mathcal{P}(\omega)$ (right top, green) are then selected and transformed back to the time domain as $p(t)$ (green line, left box). The probability $p(t)$ is then thresholded to obtain $s'(t)$ (left, blue line) and the difference is stored in the outlier set \mathcal{O} (left box, violet line).

To be able to build, maintain, and use this representation, we define four operations: reconstruction of the measured state $s(t)$, addition of a new measurement, model update, and prediction of the future state with a given confidence level.

1) *Reconstruction of the Measured State:* The aforementioned representation allows us to retrieve the past cell state $s(t)$ as

$$s(t) = (\mathcal{F}^{-1}(\mathcal{P}(\omega)) \geq 0.5) \oplus (t \in \mathcal{O}) \quad (7)$$

where \oplus is an XOR operation. The idea behind this equation is to reconstruct the probability $p(t)$ from the spectrum \mathcal{P} , set $s'(t)$ to 1 if $p(t)$ exceeds 0.5 and finally apply the XOR operator to negate $s'(t)$ if t belongs to the set of outliers \mathcal{O} .

2) *Addition of a New Measurement:* Whenever a real state $s^m(t)$ is measured, we calculate $s(t)$ by means of (7) and if it differs from $s^m(t)$, the current time t is added to the representation of the set \mathcal{O} :

$$s^m(t) \neq ((\mathcal{F}^{-1}(\mathcal{P}(\omega)) \geq 0.5) \oplus (t \in \mathcal{O})) \rightarrow \mathcal{O} = \mathcal{O} \cup t. \quad (8)$$

Since (8) takes into account the current contents of the outlier set \mathcal{O} , the time t is added to \mathcal{O} only when $s'(t)$ starts and stops matching $s(t)$, which results in Δ -encoding of the set \mathcal{O} . Nevertheless, $p(t)$ does not predict $s(t)$ with perfect accuracy and the set \mathcal{O} is likely to grow as measurements are added. After some time, the outlier set \mathcal{O} itself might contain information about dynamics that were previously unobserved and is thus not included in the set \mathcal{P} . To take into account the new information, our method offers an efficient way to update the entire spectral model.

3) *Model Update:* To update the spectral model, we reconstruct $s(t)$ including the newly added measurements by (7) and calculate its spectrum $\mathcal{S}(\omega)$. Again, we select the l coefficients with highest absolute values of the spectrum $\mathcal{S}(\omega)$, store them in $\mathcal{P}(\omega)$ and reconstruct the outlier set \mathcal{O} using (8). In a typical situation, the updated spectrum \mathcal{P} would reflect $s(t)$ more accurately, causing reduction of the set \mathcal{O} . The spectral model order l can be changed prior to the update step without any loss of information. Thus, we can change the model order and recalculate it whenever required. In our experiments, model update was typically performed on a daily basis as discussed in Section IV-B.

4) *Estimation and Prediction of Future States:* Note, that (7) can calculate $s(t)$ for any time t and that the threshold value of 0.5 can be set arbitrarily. In fact, a threshold c such that $p(t) \geq c$ represents a confidence level of the grid cell being *occupied* at time t . Therefore, we can use (7) for future prediction of $s(t)$ with a given confidence level c . In the case of prediction, the outlier set \mathcal{O} is not included in the calculation and the predicted state might not match the real future state, so we denote the prediction as $s'(t, c)$. To simplify notation, we also define $s'(t)$ as $s'(t, 0.5)$. Therefore, $s'(t, c)$ and $s'(t)$ can be calculated as

$$s'(t, c) = \mathcal{F}^{-1}(\mathcal{P}(\omega)) \geq c. \quad (9)$$

An example of the second-order spectral model which represents a quasi-periodic function is provided in Fig. 2.

5) *Estimation and Prediction for a Single Time Instant:* In many cases (such as in the scenarios described in Sections VII and VIII), one does not need to recover or predict environment states over a long time interval, but for a single time instant. Here, it is impractical to use (9) or (2), because these use the inverse Fast Fourier transform, which generates an entire sequence of probabilities. Instead, one can exploit the sparsity of the spectral model $\mathcal{P}(\omega)$ and calculate $p(t)$ simply as

$$p(t) = \alpha_0 + \sum_{j=1}^n \alpha_j \cos(\omega_j t + \varphi_j) \quad (10)$$

where ω_j , φ_j , and α_i represent the frequencies, time shifts, and amplitudes of the spectral components stored in the set $\mathcal{P}(\omega)$. The parameter α_0 , which corresponds to $\omega_0 = 0$ is the mean of $s(t)$.

C. Nonuniform Sampling Scheme

Typically, the Fourier transform is applied not to continuous functions, but to discrete sequences of data measured on a regular basis. The assumption of equally spaced samples $s(t)$ allows us to employ the fast Fourier transform (FFT) algorithm, which calculates the frequency spectrum $\mathcal{S}(\omega)$ in a very efficient manner.

However, the FFT-based model update requires recovery of the entire sequence of the observed states, which becomes computationally expensive over time. Additionally, the FFT relies on the assumption that the observations of the environment states can be performed frequently and on a regular basis, which is hard to satisfy even in laboratory settings. The requirement of regular observations also means that the robot's activity has to

be separated into a learning phase, when it frequently visits individual locations to build its dynamic environment model, and a deployment phase when it uses its model to perform the tasks requested. This division means that while the robot can create a dynamic model which is more suitable for long-term operation, it cannot be updated and thus cannot adapt to variations that were not present during the learning phase. Thus, the predictive capability of the method will become less and less reliable over time, which will negatively affect the efficiency of robot operation in long-term scenarios. To allow the robot to cope with the changing dynamics, we introduce a generalized method that can build and update the spatiotemporal model from sporadic irregular observations in an incremental manner.

This version of the method maintains a sparse frequency spectrum, which is a set \mathcal{C} of complex numbers γ_k for each modeled state. These correspond to the set Ω of modeled periodicities ω_k that might be present in the environment. Each time a state $s(t)$ is observed at time t , the aforementioned representation is updated as

$$\begin{aligned}\gamma_0 &\leftarrow \frac{1}{n+1} (n\gamma_0 + s(t)) \\ \gamma_k &\leftarrow \frac{1}{n+1} (n\gamma_k + (s(t) - \gamma_0)e^{-jt\omega_k}) \quad \forall \omega_k \in \Omega \\ n &\leftarrow n+1\end{aligned}\quad (11)$$

where n represents the number of observations. The proposed update step is analogous to incremental averaging—the absolute values of $|\gamma_k|$ correspond to the average influence of a periodic process (with a frequency of ω_k) on the values of $s(t)$. To perform predictions, we select the l components with the highest absolute value of γ_k from the set \mathcal{C} , store them in the set $\mathcal{P}(\omega)$, calculate $\alpha_j = |\gamma_j|$, $\varphi_j = \arg(\gamma_j)$ and predict $p(t)$ using (10).

The choice of set Ω , which determines the periods of the potential cyclic processes, depends on the memory size that can be allocated for the model and the longest period that is going to be modeled. In the indoor navigation experiment described in Section VIII, Ω consisted of 168 components covering periodicities from one week to 1 h. In the outdoor case Section VII-B, Ω consisted of 1000 components covering periods from one year to 8 h. The discussion about the optimal choice of set Ω along with other details of the nonuniform sampling scheme is provided in [10]. In the case of uniform sampling, the spectrums generated by (11) and FFT are identical. However, while the set of modeled periodicities of the FFT-based method scales naturally with the duration of the data collection, the set of periods Ω captured by the nonuniform scheme is fixed.

D. Modeling Persistence

The aforementioned representation is primarily aimed at modeling the environment changes from a long-term perspective. Thus, the predictions of future states are based on the observed periods of the changes in the past. While this is useful for long-term forecasts, prediction of near future states should take into account not only the states' periodicity, but also their persistence. For example, if a given visual feature was observed 10 s ago, it is quite likely that it will be still observable even

though it is not usual to observe it at this time of day or week. To enable the deployment of the proposed method on continuously operating mobile robots, the ability to perform short-term predictions is also important. Thus, we extended the FreMEn representation with a persistence model, which acts as short-term memory that represents the expectation that the given state did not change since the last observation if the observation was performed recently. This is achieved by extending the update scheme of (11) by

$$\begin{aligned}\tau^{-1} &\leftarrow \frac{1}{n+1} \left(n\tau^{-1} + \frac{|s(t) - s(t_l)|}{t - t_l} \right) \\ s(t_l) &\leftarrow s(t) \\ t_l &\leftarrow t\end{aligned}\quad (12)$$

where $s(t_l)$ represents the last observation at time t_l and τ represents the modeled state persistence, i.e., the mean time between the state's changes. To predict the value of state $s(t)$ for a future time t , we calculate

$$p'(t) = s(t_l)e^{\frac{t_l-t}{\tau}} + p(t) \left(1 - e^{\frac{t_l-t}{\tau}} \right) \quad (13)$$

where $p(t)$ is calculated by means of (10). Note that for the predictions which closely follow the last observation, i.e., $|t - t_l| \ll \tau$, the expression $e^{\frac{t_l-t}{\tau}}$ is close to 1, which means that the expected occupancy would be the same as the one recently observed. Using (13) to predict the more distant future, i.e., $|t - t_l| \gg \tau$, causes the expression $e^{\frac{t_l-t}{\tau}}$ to be close to 0, which suppresses the effect of the latest observation on $p'(t)$ and emphasizes $p(t)$, which represents the behavior of the predicted state from a long-term perspective. The experiments presented in Section VIII show that the addition of the persistence model to the FreMEn representation allows us to deal with nonperiodic changes as well.

IV. PERFORMANCE EVALUATION

In the rest of this paper, we examine the tractability of using our approach, the Frequency Map Enhancement (FreMEn), as a core component of spatiotemporal models for mobile robotics. In particular, we investigate the following questions:

- 1) How many parameters of the spectrum typically have to be stored to represent and predict the environment state?
- 2) How efficiently can it represent long-term observations?
- 3) What is the accuracy of its predictions?
- 4) How can the approach benefit long-term autonomy of mobile robots?

To answer these questions, we analyzed several types of environment models gathered by a mobile robot which was continuously operating for several months in a human-populated indoor environment. To quantitatively evaluate the performance of the FreMEn, we use four different criteria relevant to mobile robot mapping. The prediction and estimation errors ϵ_p and ϵ_e relate to the faithfulness of the FreMEn, i.e., its ability to correctly estimate and predict the environment states for a given time period. The compression ratio relates to the memory efficiency of the FreMEn representation, i.e., the memory needed to represent

the long-term observations of the environment. The update time relates to the computational complexity of the FreMEn.

A. Prediction and Estimation Error

Knowing the coefficients $P_i(\omega_i)$ of the spectrum \mathcal{P} allows us to calculate an estimate $s'(t)$ of the original state $s(t)$ by (9). A natural concern is the accuracy of reconstruction of $s'(t)$, because it will affect the prediction capabilities of the spectral model and the size of the outlier set \mathcal{O} . One can expect that increasing the spectral model order (i.e., including more coefficients in \mathcal{P}) would enable more precise reconstruction of $s'(t)$ from the spectral model \mathcal{P} alone. However, as the number of parameters grows, the model becomes more adjusted to the specific time series of the observations $s(t)$, which decreases its ability to predict the environment state in the future.

To evaluate the quality of the spectral model, we define the estimation error $\epsilon(t_a, t_b)$ as the ratio of the correctly estimated signal $s'(t)$ on a given time interval $t \in [t_a, t_b]$ to the length of the entire interval

$$\epsilon(t_a, t_b) = \frac{1}{t_b - t_a} \int_{t_a}^{t_b} |s'(t) - s(t)| dt. \quad (14)$$

The estimation error can be also calculated from the intersection of the intervals in the outlier set \mathcal{O} and (t_a, t_b) as

$$\epsilon(t_a, t_b) = \frac{|(t_b, t_a) \cap \mathcal{O}|}{|(t_b, t_a)|}. \quad (15)$$

Since the outlier set \mathcal{O} is Δ -encoded, the calculation of (15) can be performed very efficiently.

Typically, the error would be calculated for the entire series of observations, i.e., from time 0 to the time of the latest observation τ . We call this error the estimation error of the spectral model and denote it as $\epsilon_e = \epsilon(0, \tau)$.

Suppose that the sequence $s(t)$ includes observations made from 0 until τ and that the spectral model $\mathcal{P}(\omega)$ had been calculated using only observations made between 0 and τ' , where $\tau' < \tau$. Then, calculation of $s'(t)$ for $t \in (\tau', \tau]$ by (9) is actually a prediction. Thus, the estimation error $\epsilon(\tau', \tau)$ relates to the ability of the spectral model to predict future states from past observations. We denote the error $\epsilon(\tau', \tau)$ as the prediction error ϵ_p . Note that the aforementioned situation happens every time the model is updated: the value of τ' corresponds to the time of the last update, while the outlier set already contains observations that have been obtained after τ' . Since calculation of ϵ_e and ϵ_p by (15) is computationally efficient, the proposed algorithm can use it to decide whether a model update is needed as well as the optimal order of the spectral model. This can be employed to adapt the model order based on the observed dynamics rather than using a fixed model order.

Although the calculation of both errors is similar, they represent different properties of the FreMEn model. The estimation error ϵ_e relates to the ability of the spectral model to recover past observations and ϵ_p represents the ability to predict future states. While ϵ_e decreases with the model order, the dependence of ϵ_p on the model order is more complex.

Note also that (14) relates only to the reconstruction of the states $s(t)$ from the spectral model \mathcal{P} before the outlier set is taken into account. The application of the outlier set \mathcal{O} allows us to recover the sequence $s(t)$ in an exact way.

B. Choosing the Model Order

As mentioned before, the dependence of the prediction error ϵ_p on the model order l is not straightforward. Choosing too low a value l causes overgeneralization, while choosing too high a value of l causes overfitting of the FreMEn model. To select the proper value of the model order l , we evaluate the model's predictive capability for different values of l , choose the order l' with the lowest prediction error ϵ_p and then perform the model update with the value l' . In a typical scenario of robot deployment in our project [31], updates of the FreMEn models are performed at midnight every day when the robot replenishes its batteries at its charging station. Before updating, the performance of the FreMEn models with different orders l is evaluated by comparing their predictions to the observations gathered since the last update (i.e., since midnight the previous day). Then, the models are updated using the order which achieved the lowest prediction error. A typical value for an optimal model order l' is 2 or 3 and the typical time to establish the optimal order and update the spatiotemporal models used in our robot deployments is less than a minute.

C. Compression Ratio

The compression ratio indicates the efficiency of the model in representing the spatiotemporal dynamics of the environment. Rather than evaluating the compression ratio from a theoretical point of view, we adopt a more practical approach and base our calculations on the actual size of the file that contains the spectral model. Assuming that a file of size z [bits] contains a FreMEn model of an environment with n states and m observations, and that a traditional model would use one bit per observation, the compression ratio is simply

$$r = \frac{mn}{z}. \quad (16)$$

In some scenarios, maintaining an entire outlier set \mathcal{O} might be infeasible due to memory constraints, and the past observations $s(t)$ are represented solely by the set \mathcal{P} . While this results in lossy compression with quality corresponding to the estimation error ϵ_e , the memory size of this reduced representation is independent of the number of measurements and is determined by the number of modeled states n and the model order l , which can be selected *a priori*.

D. Update Time

The computational complexity of the proposed method is given by the complexity of the fast Fourier transform algorithm, which is $O(m \log m)$, where m is the length of the processed sequence. This indicates that the time t needed to build, update, or reconstruct a spectral model with n states and m observations by (1) and (7) is $t \sim m n \log(m)$. Thus, the update time of the

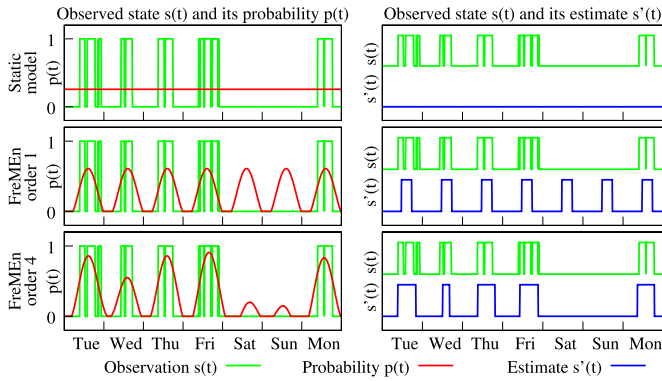


Fig. 3. Week-long state model of a single cell of an occupancy grid. While the traditional static model simply assumes that the probability of the cell occupancy is $\sim 25\%$, the frequency-enhanced model captures the cell's dynamics. Note the model improvement as more spectral components are included.

FFT-based model increases with the number of past observations.

However, the computational complexity of the incremental calculation scheme performed by (11) depends only on the number of new observations m' , the number of independent states n , and the number of maintained spectral components k , and therefore, does not depend on the number of past observations. On the other hand, it requires to maintain a larger number of spectral components and is less memory efficient than the FFT-based model.

Since we are concerned with the practical applicability of our approach rather than with theoretical bounds of computational complexity, we measured the real time required to calculate and update the spectral models in our evaluations.

V. SINGLE-STATE DYNAMIC MODEL

To experimentally verify the feasibility of the proposed approach, we first gathered a week-long dataset containing a single state.

This dataset was gathered by a RGB-D camera monitoring a small university office from a fixed location. Its range measurements were used to establish the occupancy of a single $20 \times 20 \times 20$ cm cell located in the middle of the room entrance. This cell was occupied when the door was closed and when people passed through the door, otherwise it was free. The office had an “open door” policy, i.e., the door remained open whenever the office was occupied. Therefore, the measured state $s(t)$ corresponded strongly to the presence of people inside the office. Every time someone went through the door, the monitored cell was briefly occupied and the room was considered empty, which introduced noise on the measured state $s(t)$. The measurements were taken continuously for one week (July 23–29, 2013) at a rate of 30 Hz, so the state observation consists of 18 million values. For the purpose of this evaluation, we subsampled the values by 15, which means that the state $s(t)$ is measured twice per second, so that $s(t)$ consists of more than a million values. After this week, two additional single-day datasets (July 31 and August 5 2013) were gathered.

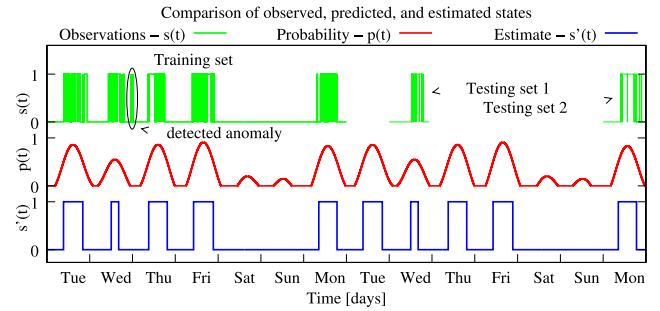


Fig. 4. Comparison of state observations, established probabilistic model, and predicted values.

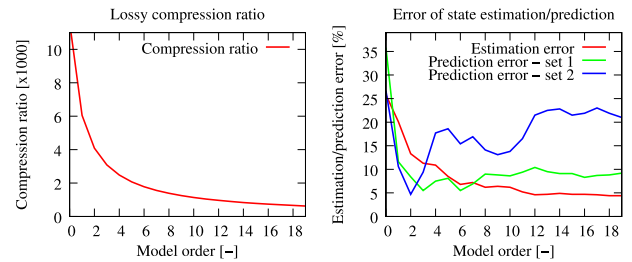


Fig. 5. Influence of the number of spectral components (model order l) on the model's compression ratio and errors of estimation and prediction.

To evaluate the proposed method's capability to represent the temporal dynamics of the observed state, we built several spectral models of the training dataset. Fig. 3 shows that the spectral model captures the state dynamics, which results in a more faithful representation of the given state. The first row of Fig. 3 shows that the traditional probabilistic model would simply assume that the door is open with 25% probability. Modeling the state with FreMEn of order 1, i.e., considering only one periodic process results in a model that suggests that the door is likely to be open during the afternoon rather than at night—see the second row of Fig. 3. Adding three other spectral components results in a model that captures the weekly periodicities as well—the probability of the door being open (see the last row of Fig. 3) during weekends is lower than during the working days. This result suggests that the method's ability to model the dynamics of the measured state increases with the number of model parameters included.

The dependence of the estimation and prediction errors on the number of components of the spectral model is shown in Fig. 5. To estimate the dependence of the model estimation and prediction errors on the number of model parameters, we built a spectral model of the one-week-long training dataset. The accuracy of estimation ϵ_e was calculated as the difference between the original and reconstructed signal by (14). Moreover, we calculated the accuracies of prediction ϵ_{p1} and ϵ_{p2} for two days of the following week, see Fig. 4.

The results in Fig. 4 indicate that the static model (i.e., FreMEn order 0) achieves an estimation error of 25%–35%. The results also indicate that while the estimation error ϵ_e decreases monotonically with the model order, the prediction error is not

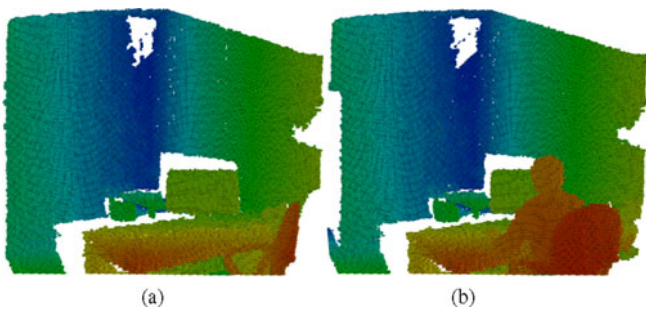


Fig. 6. Fine-grained 3-D occupancy grids of the “Office” dataset. (a) Empty office 3-D grid. (b) Occupied office 3-D grid.

necessarily monotonic. Rather, the local minima of the prediction errors suggest that for the purpose of predictions, one should use spectral models of orders around 2 or 3 to prevent overfitting. The overfitting effect is more prominent with the second testing set, which might be caused by the longer prediction horizon.

The test indicates that the spectral model can represent millions of measurements with only a few complex numbers. Fig. 5 shows that the spectral model without the outlier set \mathcal{O} achieves compression ratios in the order of 1 : 1000 while losing less than 5% of information. The size of the Δ -encoded outlier set was about 360 values representing 180 time intervals where the spectral model did not match the measured sequence, which corresponds to a lossless compression ratio of $\sim 1:100$. The time needed to build the spectral representation on an i7 processor was 3.7 ms, which illustrates the efficiency of the chosen Fast Fourier transform implementation [32].

Our method can be also used to detect anomalies, i.e., situations where a local state of the world deviates significantly from the spectral world model of the robot. Since our model can predict the state $s(t)$ with a given confidence value by (9), we can assume that a measurement $s^m(t)$ is anomalous with confidence level c if $s^m(t) < s(t, c)$ or if $s^m(t) > s(t, 1 - c)$. Fig. 4 shows that FreMEn-based anomaly detection with confidence level 99% correctly detected a situation when the room was accessed by an unexpected visitor shortly after midnight.

VI. LARGE SPATIOTEMPORAL REPRESENTATION

To evaluate the ability of the proposed method to represent the long-term dynamics of three-dimensional environments, we collected two million occupancy grids of a University office over the course of 112 days. Similarly to the previous experiment, the dataset was collected by a stationary RGB-D camera that captured and stored a depth image every 5 s. These range measurements were integrated into a FreMEn occupancy grid [9], where the occupancy of each cell was modeled by the proposed method. Fine-grained occupancy grids captured by the RGB-D camera are shown in Fig. 6 (for the purpose of visualization, the resolution of the grids shown is higher than those in the dataset). Each day, the spectral model of the entire grid was updated and the resulting representations were saved in separate files. To evaluate the efficiency of the resulting 4-D representations, we measured the compression ratios, estimation

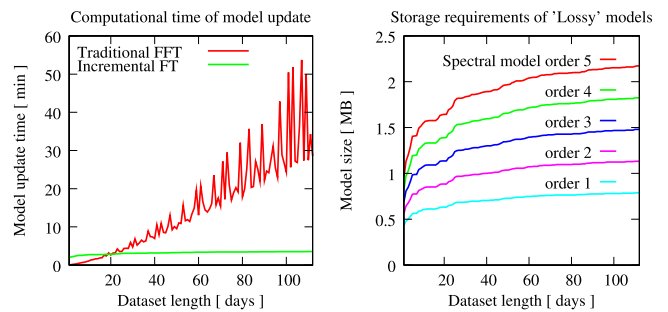


Fig. 7. Computational and memory requirements of the FreMEn spatiotemporal occupancy grids.

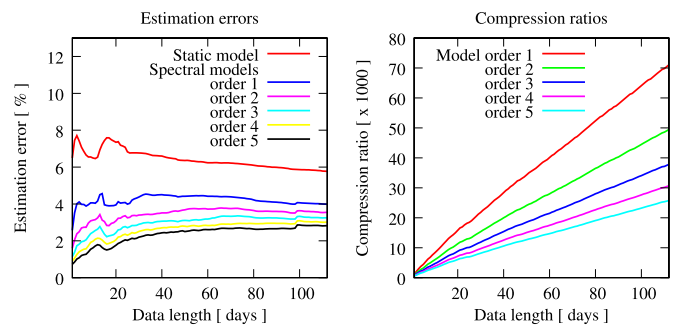


Fig. 8. Estimation errors and compression ratios of the FreMEn spatiotemporal occupancy grids.

precisions, and times needed to calculate the update. The compression ratios were calculated simply by comparing the size of the saved files to the theoretical size of a traditional model by (16), where the number of modeled states n , i.e., the number of cells in the grid was $\sim 213\,000$ and 17 200 observations per day were considered. This means that storing all the observed states would require ~ 500 MB per day and a naïve representation of the entire dataset would require around 50 GB of storage space. The estimation error of the entire model was calculated as an average of estimation errors of the individual cells that changed at least once—calculating the average estimation error for all cells would result in small numbers, because most of the cells represent space that is always empty. Finally, the update time was obtained by the direct measurement of the time needed to update the spectral models of all the grid cells. These experiments were performed on an i7-4500U processor with 16 GB of RAM.

Five types of spectral models were calculated. The first, “lossless” model maintains not only the spectral representation, but also an outlier set \mathcal{O} of each cell, and can recover all the measurements accurately. The other, “lossy—order 1–5” models did not use the outlier set and maintained 1 to 5 spectral components of the dynamic cells. The dependencies of the sizes of the “lossy” models on the length of the dataset represented are shown in Fig. 7. One can see that after some initial growth, the storage requirements of the models stabilize at the order of megabytes. The growth of the “lossy” models is caused by the fact that longer data collection means that more cells change

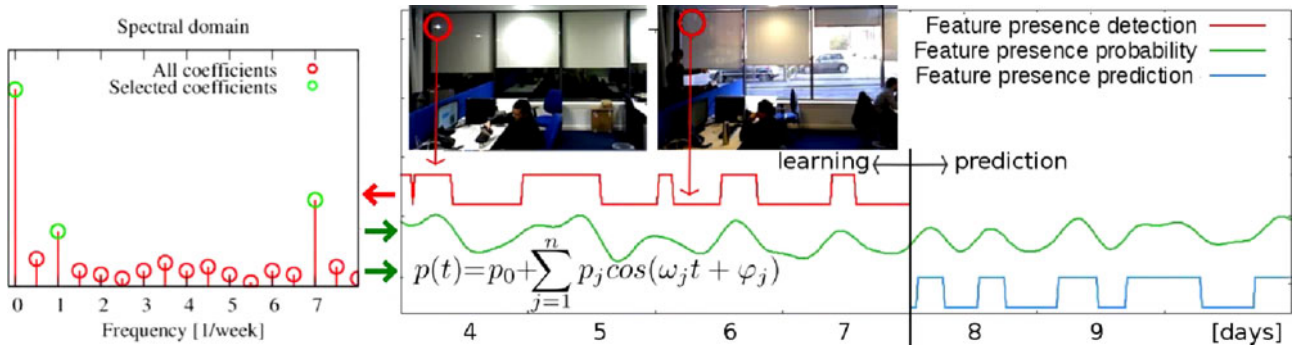


Fig. 9. Frequency-enhanced feature map [4] for visual localization: The observations of image feature visibility (center, red) are transferred to the spectral domain (left). The most prominent components of the model (left, green) constitute an analytic expression (center, bottom) that represents the probability of the feature being visible at a given time (green). This is used to predict the feature visibility at a time when the robot performs self-localization (blue).

their states at least once, which causes the method to extend their temporal models.

Given that the naïve representation of the dataset grows by 500 MB per day, the compression rates of the “lossy” models actually grow in time (see Fig. 8) and are in orders of 10 000. The “lossless” representation grows linearly with time at a rate of 2 MB per day achieving compression rates of 1:250. Fig. 7 also shows that the time needed to update the model, which represents 4×10^{11} cell observations is reasonably short—creation of a 16-week-long spatiotemporal model takes less than 1 h. Using the nonuniform incremental Fourier Transform results in an update time that exhibits a similar trend to the “lossy” model sizes. This is caused by the fact that the number of cells for which the transform has to be calculated increases over time, i.e., the same effect that causes the growth of the “lossy” models. Finally, the estimation errors of the spatiotemporal models with different orders are presented in Fig. 8, which shows that as the model includes more spectral components, its estimation error and compression rates drop. Compared to the “Static” model, which fails to correctly estimate approximately 6% of the states, the “lossy” FreMEn estimates fail in 3% to 4% cases. This means that using the FreMEn method reduces the amount of incorrectly estimated states by 30%–50%. Using the lossless method results in faithful (0% error) state reconstruction at the expense of a lower (1:250) compression rate.

VII. FREMEN FOR MOBILE ROBOT LOCALIZATION

The results of the previous experiments demonstrate that through explicit modeling of the environment dynamics, our method can efficiently represent the evolution of indoor environments over time. Moreover, we have shown that the method can predict future environment states. In this experiment, we evaluate the usefulness of these predictions for mobile robot localization in indoor and outdoor environments. The considered scenario is vision-based localization. Given a topological map, where each node is associated with a set of image features visible at that particular location, the robot has to decide on its current location based on its camera image. The difficulty is that the appearance of the locations (i.e., visibility of the image features) varies over time. This problem has been tackled by

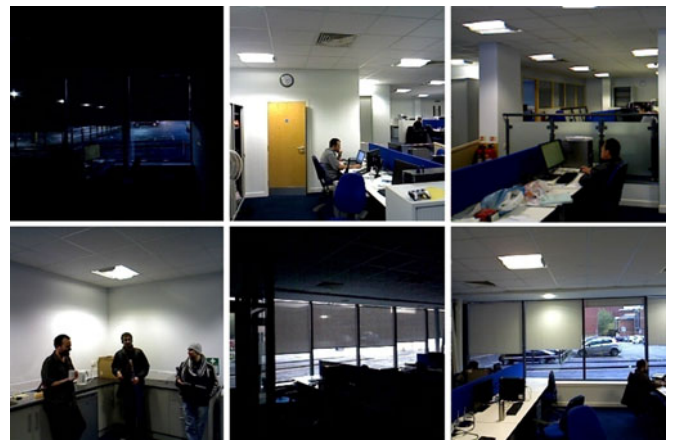


Fig. 10. Example images of the indoor training dataset. Shows the appearance of six monitored locations on November 2013.

attempting to identify the most stable [24] or most useful [33] features, or by remembering several appearance models for the same location [3]. Other approaches [6], [25] attempted to infer the environment appearance for the particular time(s) by modeling the persistence [25] or systematic appearance change of visual features [6]. In this experiment, we predict the visibility of the individual image features at a given time by FreMEn, see Fig. 9 and video at <http://fremen.uk>.

A. Indoor Localization

The environment considered is a large open-plan office of the Lincoln Centre for Autonomous Systems, where an autonomous robot captured RGB-D images of eight designated areas every 10 min. During a week-long data collection session in November 2013, the robot visited each of the eight locations 144 times per day, collecting a training dataset that contains more than 8000 images. To document the appearance change over one year, we provide images from the three testing datasets in Fig. 11. The three testing datasets were collected one week (November 2013), three months (February 2014), and one year (December 2014) after the training dataset collection. Each of these datasets was gathered for 24 h and contains over 1000 images. Representative examples of the images of the training



Fig. 11. Example images of the indoor testing datasets. Shows the evolution of one of the monitored places over the course of one year. (a) November 2013. (b) February 2014. (c) December 2014.

dataset are shown in Fig. 10. The gathered images were processed by the BRIEF algorithm [34], which was evaluated as one of the best performing image feature extractors in outdoor scenarios of long-term localization [35], [36], and our tests confirmed its good performance in indoor scenarios as well. The features of the training dataset belonging to the same locations were matched and, thus, we obtained their visibility over time, which was then processed by our method. To choose the order of the FreMEN models, we adopted the scheme described in Section IV-B, i.e., to select the correct order l , the FreMEN models were trained initially on the first six days of the training data and their predictive capability was evaluated on the last training day. Next, the models were trained using the entire 7-day-long dataset. Thus, we obtained a dynamic appearance-based model of each topological location that can predict which features are likely to be visible at a particular time, see Figs. 1 and 9.

To test if these predictions actually improve robot localization, the following procedure was performed for each of the ~ 3000 images in the testing datasets. First, the method established the time t_c when the testing image was captured. Then, the dynamic map created during training was used to calculate the probability of each feature’s visibility at time t_c . Next, the n most likely visible features at each location were selected, which resulted in eight sets denoted as \mathcal{F}_i , each containing n image features. Finally, the features of the testing image were extracted and matched to the sets \mathcal{F}_i . If the set with the highest number of matches corresponded to the real location of the robot, localization was considered successful, otherwise it was considered a failure.

To compare the proposed algorithm with other localization methods, we implemented a simple version of the experience-based approach developed by the Churchill and Newman [3]. During training, this method attempts to determine the robot location based on the camera input, and if it fails, the current appearance (aka experience) is added to the set of “experiences” that are associated with the given location. Thus, each location is associated with several experiences which are matched to the currently perceived sensory data. While the method introduces a certain computational overhead caused by the fact that there are more experiences than actual locations, this overhead is compensated by the method’s robustness to significant appearance changes. This computational overhead was reduced in [37] by inferring the most probable appearances that the robot will experience around a given location. Since we use a slightly different

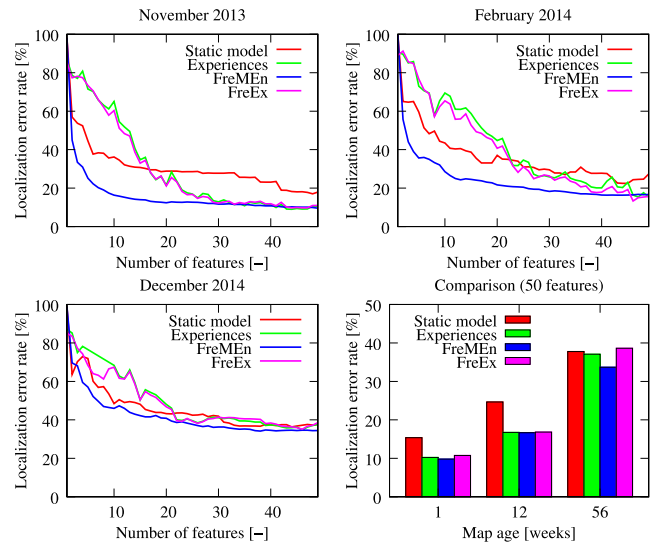


Fig. 12. Localization error rates for different indoor testing datasets, methods, and feature numbers. The first three graphs show the dependence of the error on the number of features used for localization. The fourth graph compares the localization errors of the different methods and datasets assuming that the number of features used is 50.

setup and scenario than the one considered in [3], we had to introduce a slightly different version of the experience-based localization. In our case, an experience consists of the robot position and image coordinates and descriptors of the detected visual features, and we did not use the optimizations introduced in [37].

We also attempted to reduce the aforementioned computational overhead by combining the experience-based approach with FreMEN—the FreMEN was used to calculate the probability of a given experience for a given time, so we could use only the relevant experiences for localization. In the following evaluation, this frequency-enhanced experience method will be coined as “FreEx.” Processing of our training dataset by the experience-based method generated over 170 different experiences tied to eight different locations.

The dependence of the average localization error for each indoor testing dataset on the number of features n used for localization is shown in Fig. 12. The results indicate that the localization robustness of the FreMEN is only marginally better compared to the experience-based method and they both outperform the “static” approach that relies on the most stable image features. However, while the FreMEN approach improves the robustness by predicting the appearance of the eight locations, the experience-based method requires that the current camera image is matched to all of the 170 experiences, which is computationally more expensive. This is partially mitigated by the FreEx approach, which typically localizes the robot based on 100 experiences, which are selected from the 170 learned ones based on the current time.

While the results show that explicit representation of environment change improves the localization robustness, the improvement diminishes with map age. Since we can observe the same effect for the FreMEN and experience-based methods, the effect is probably not caused by change in the environment dynamics.



Fig. 13. Seasonal variations at location I of the Michigan dataset. (a) Winter 2012. (b) Summer 2012.



Fig. 14. Seasonal variations at location I of the Stromovka dataset. (a) Winter 2010. (b) Summer 2010.

Rather, the environment is subject to unexpected and cumulative changes, which affect its appearance in a way that is not possible to predict by the approaches evaluated. This issue severely affected the FreEx approach, which failed to correctly predict the relevant experiences to be used for visual localization. The effect of map decay could possibly be mitigated by active re-observation of locations that were not visited for a long time, e.g., by means of lifelong exploration [10]. This problem also leads to fascinating questions that regard forgetting of obsolete observations and adaptation of the forgetting speed to the rate of environmental change, although these questions are beyond the scope of the work presented here.

B. Outdoor Localization

To evaluate the performance of the FreMEN for visual localization in outdoor environments, we performed the same comparison on two datasets, which capture the seasonal changes of ten different locations in two semiurban environments.

The images of the first five locations were obtained from the North Campus long-term vision and lidar dataset (NCLT) which was collected at University of Michigan to support research on image features for dynamic lighting conditions [38]. The original NCLT dataset [39] was gathered during 27 data-collection sessions performed over 15 months and includes LIDAR, GPS, and odometry data. For our evaluation, we selected five different locations from the NCLT dataset and created the training dataset from 12 images captured at each location at a different time. To create the testing dataset, we randomly selected three images per location from the set of images not used for training. Unlike the two aforementioned datasets, the Michigan set was not gathered on a regular basis and thus, we used the nonuniform version of FreMEN introduced in Section III-C.

The second set of outdoor images was obtained from the Stromovka dataset [40] that was collected in one of Prague's arboretums to support research on long-term teach-and-repeat navigation [41]. The Stromovka dataset contains images that were captured by a mobile robot every month from September 2009 until the end of 2010, and three additional image sets that were collected during 2011 and 2012. Compared to the Michigan dataset, the Stromovka one spans a longer time period and contains more foliage and fewer buildings. Moreover, seasonal weather variations in Prague are more extreme than in Ann Arbor, see Figs. 13 and 14. Thus, the appearance variations of the Stromovka dataset images are greater than the Michigan ones.

To perform the evaluation, we trained both methods using the datasets gathered during the first 12 months. Then, we

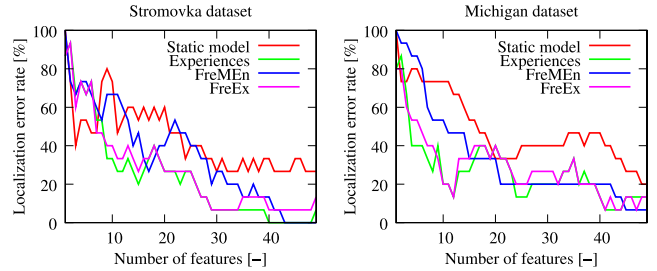


Fig. 15. Localization error rates for the Stromovka and Michigan outdoor datasets. Shows the dependence of the error rates on the methods and feature numbers used.

calculated the localization error rates on the testing sets, which were collected during the following months and years. The dependence of the localization error for both outdoor datasets on the number of features n is shown in Fig. 15. Similarly to the indoor case, the localization error rates of the FreMEN and experience-based methods were much lower compared to the “static” method, which neglects the appearance change and takes into account only the most stable features. However, the FreMEN localization was computationally more efficient, because it had to match the current camera image to five predicted maps, while the experience-based approach used 15 and 21 different experiences in the Stromovka and Michigan cases, respectively. For the case of outdoor datasets, we did not have enough data to properly estimate the best-performing model order l , so we set l to a conservative value of 1.

The aforementioned localization experiments were performed with a relatively low number of image features per image, because the number of locations to distinguish is low. In such cases, extracting a large number of image features will cause the evaluated methods to exhibit a similar performance. To demonstrate the advantages of our approach while utilizing the full power of the feature extractors available would require long-term data collection in much larger environments.

C. Predictive Capability

To evaluate the predictive capability of the FreMEN approach, we calculated the average probability that a predicted feature will actually be visible in the testing images and compared this with a static approach. First, we calculated the ten most stable features across the training sets and calculated how often these are matched to the features extracted from the testing images of the same location. This corresponds to the Static

TABLE I
PROBABILITY OF FEATURE REOBSERVATION [%]

Method	Indoor			Outdoor	
	Nov'13	Feb'13	Dec'14	Strom	Mich
Static	39.5	25.7	24.3	38.1	30.8
FreMEn	55.2	31.2	26.8	47.5	40.8

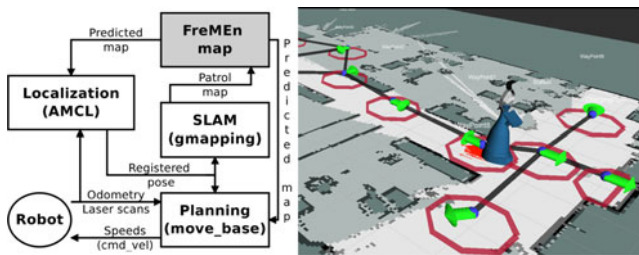


Fig. 16. Navigation system overview. Proposed navigation stack on the left and predicted and observed 2-D grids on the right.

method described in the previous sections. Then, we repeated the procedure with the ten features, which were predicted by FreMEn to be most likely visible at the given time. The results, summarized in Table I indicate that the image features predicted by the FreMEn method for a particular time are more likely to be visible compared to the features that were most frequently reobserved in the training sets.

VIII. FREMEN FOR MOBILE ROBOT NAVIGATION

The experiments presented previously were conducted in an offline manner on prerecorded data. To use FreMEn online as an integral component of a long-running autonomous system, we developed a FreMEn occupancy grid which was integrated in the ROS navigation stack [42]. This spatiotemporal grid uses the nonuniform version of FreMEn with the recency model proposed in Section III-D. During autonomous navigation, our robots build temporally local maps and integrate them into the global spatiotemporal grid. Through reobservation of the same spatial locations, the spatiotemporal grid obtains information about long-term environment dynamics and gains the ability to predict the future environment states. This predictive ability enables the generation of time-specific 2-D maps which can be used by the robot's localization and planning modules. The integration of this predictive spatiotemporal model in the system and a visualization of the map building process are shown in Fig. 16. In this scenario, we evaluated the impact of the proposed spatiotemporal representation on localization accuracy and efficiency of path planning. To do this, we deployed a mobile robot for several days at the Lincoln Centre for Autonomous Systems, having it regularly patrolling the office in a predetermined path several times per hour, using the proposed modification of the ROS navigation stack. The patrolled area contained a 1.5-m-wide corridor. On its sides, there are storage cupboards that are used by research staff and closed at the end of their working day. When a cupboard door is left open, the corridor

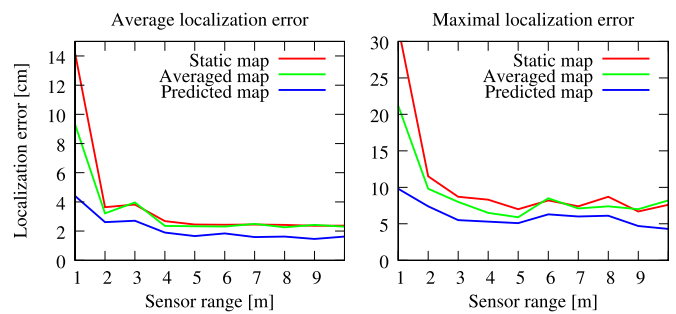


Fig. 17. Localization error for different ranges of the laser scanner and different types of the maps. Predicting a map for a particular time improves localization accuracy, although the improvement is only marginal for long-range sensors.

TABLE II
NAVIGATION STATISTICS

Environment	Static	Changing		
		Average	Predicted	
Map	Static			
Average speed	$[\frac{m}{s}]$	0.21	0.15	0.18
Recovery events	$[-]$	1	21	12

appears to be wider and its center may be perceived as displaced to one side.

To evaluate the accuracy of robot self-localization, we installed an independent localization infrastructure over the monitored corridor [45]. To estimate the impact of the environment change and sensor range on the localization precision, we processed laser, odometry, and ground truth data from 20 different passes of the robot and trimmed the laser data at different lengths. We then performed standard ROS-based AMCL localization on the “static,” “averaged,” and “predicted” 2-D maps and compared the robot positions to the ground truth obtained by the independent localization infrastructure.

The results shown in Fig. 17 indicate that use of the time-specific, predicted maps improves the localization precision in a significant way if the range of the laser rangefinder is lower than the overall map size. However, a small difference in localization precision can have a significant impact on the efficiency of the robot navigation and quality of the constructed maps.

To evaluate the navigation efficiency, we processed navigation statistics of 180 different patrol runs. The data from each patrol run contains the robot's average speed and the number of events where normal navigation behavior failed and the robot had to perform custom recovery behaviors in order to proceed with its patrol. The gathered navigation statistics were divided into three groups of 60 patrols each. The first group contained patrols where the system was using a static map when no environment changes were happening. The second group contained patrols where the robot was using an “averaged” map, which slowly adapts to the observed change. The third group contained patrols where the robot was using a “predicted,” time-specific map that took into account not only the periodicity, but also the persistence of the observed changes.

Table II indicates that in a static environment, the robot could navigate efficiently even when using a static map, but as soon as the environment began to change, the navigation efficiency was affected in a negative way. However, the negative effect of the changes was slightly lowered through the use of the proposed dynamic map, which represents the environment changes in an explicit way.

IX. CONCLUSION

We have presented a novel approach for spatiotemporal environment modeling in the context of mobile robotics. The approach is based on an assumption that from mid- to long-term perspectives, the environment is influenced by various processes, some of these being periodical. We hypothesize that certain regularities in the environment dynamics can be represented by the periodicity, amplitude, and time shift of these underlying processes, and propose to identify these parameters through spectral analysis based on the Fourier transform.

Knowledge of these processes allows us to represent the elementary states of the environment models by probabilistic functions of time, which enables efficient representation of arbitrary timescales, anomaly detection, and prediction of future states. To evaluate the performance of the proposed method in real long-term scenarios, we applied it to data gathered by mobile robots over extended time periods of months and years.

The results indicate that the proposed method can represent arbitrary timescales with constant (and low) memory requirements, achieving compression rates between 10^3 and 10^5 while predicting the future states with error rates of less than 10%. We have also demonstrated that our method's prediction of the environment appearance improved vision-based localization in changing environments. Moreover, we demonstrated that integrating the method in the ROS navigation stack improves the efficiency of robot navigation.

In the future, we would like to extend the approach so that it can take into account sensor noise and represent not only binary, but also higher dimensional states, such as object positions. While the method itself does not exceed the performance of other approaches for persistent localization in changing environments, such as [3], [5], [6], [43], its simplicity enables its application to other scenarios related to long-term autonomy and life-long learning. To provide an overview of the method's applications and to allow its use by other researchers, we have released the method's source code, examples of use, and datasets at <http://fremen.uk>.

ACKNOWLEDGMENT

The authors would like to thank M. Kulich and P. Urcola for remarks regarding probability theory.

REFERENCES

- [1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambridge, MA, USA: MIT Press, 2005.
- [2] D. Austin, L. Fletcher, and A. Zelinsky, "Mobile robotics in the long term-exploring the fourth dimension," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, 2001, pp. 613–618.
- [3] W. S. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *Int. J. Robot. Res.*, vol. 32, pp. 1645–1661, 2013.
- [4] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localization for service robots in dynamic environments using spectral maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 4537–4542.
- [5] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *Int. J. Robot. Res.*, vol. 32, pp. 1662–1678, 2013.
- [6] P. Neubert, N. Sünderhauf, and P. Protzel, "Superpixel-based appearance change prediction for long-term navigation across seasons," *Robot. Auton. Syst.*, vol. 69, pp. 15–27, 2014.
- [7] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3706–3711.
- [8] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, and T. Duckett, "Where's waldo at time t? using spatio-temporal models for mobile robot search," in *Proc. Int. Conf. Robot. Autom.*, 2015, pp. 2140–2146.
- [9] T. Krajník, J. Santos, B. Seemann, and T. Duckett, "Fractomap: An efficient spatio-temporal environment representation," in *Advances in Autonomous Robotics Systems*. New York, NY, USA: Springer, 2014, pp. 281–282.
- [10] J. M. Santos, T. Krajník, J. Pulido Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4D spatio-temporal maps," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 684–691, Jul. 2016.
- [11] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millennium*, San Mateo, CA, USA: Morgan Kaufmann, pp. 1–35, 2002.
- [12] D. Hähnel, D. Schulz, and W. Burgard, "Mobile robot mapping in populated environments," *Adv. Robot.*, vol. 17, pp. 579–597, 2003.
- [13] D. Wolf and G. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Auton. Robots*, vol. 19, pp. 53–65, 2005.
- [14] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *Int. J. Robot. Res.*, vol. 26, no. 9, pp. 889–916, 2007.
- [15] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti, "Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments," in *Proc. ICRA Workshop Safe Navigation Dynamic Environ.*, 2009.
- [16] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Proc. Int. Conf. Intell. Robots Syst.*, 2014, pp. 1854–1861.
- [17] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proc. Robot., Sci. Syst.*, 2005, pp. 17–24.
- [18] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *Int. J. Robot. Res.*, vol. 29, no. 9, pp. 1131–1153, 2010.
- [19] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. Int. Conf. Intell. Robots Syst.*, 2009, pp. 1156–1163.
- [20] S. Hochdorfer and C. Schlegel, "Towards a robust visual SLAM approach," in *Proc. Int. Conf. Adv. Robot.*, 2009, pp. 1–6.
- [21] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proc. 20th Nat. Conf. Artif. Intell.*, 2005, pp. 1324–1329.
- [22] N. Mitsou and C. Tzafestas, "Temporal occupancy grid for mobile robot dynamic environment mapping," in *Proc. Mediterranean Conf. Control Autom.*, 2007, pp. 1–8.
- [23] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: A method for classifying the spatio-temporal properties of the environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, 2002, pp. 409–414.
- [24] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *J. Robot. Auton. Syst.*, vol. 59, pp. 285–295, 2011.
- [25] D. M. Rosen, J. Mason, and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *Proc. Int. Conf. Robot. Autom.*, 2016, pp. 1063–1070.
- [26] M. Yguel, O. Aycard, and C. Laugier, "Wavelet occupancy grids: A method for compact map building," in *Field and Service Robotics*. New York, NY, USA: Springer, 2006, pp. 219–230.
- [27] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 3–8, 2013, pp. 1–6.
- [28] T. Krajník, J. Pulido Fentanes, J. Machado Santos, and T. Duckett, "Frequency map enhancement: Introducing dynamics into static environment models," in *Proc. ICRA 2016 Workshop AI Long-Term Autonomy*, 2016.

- [29] R. N. Bracewell and R. Bracewell, *The Fourier Transform and Its Applications*. New York, NY, USA: McGraw-Hill, 1986, vol. 31999.
- [30] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, “Long-term mobile robot localization in dynamic environments using spectral maps,” in *Proc. AAAI Conf. Artif. Intell. (Video Session)*, 2015. [Online]. Available: http://www.aaaivideos.org/2015/03_spectral_map_localization/
- [31] N. Hawes *et al.*, “The STRANDS project: Long-term autonomy in everyday environments,” *IEEE Robot. Autom. Mag.*, 2017, to be published.
- [32] “The FFTW C library,” 2016. [Online]. Available: <http://www.fftw.org/>
- [33] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, “Summary maps for lifelong visual localization,” *J. Field Robot.*, vol. 33, pp. 561–590, 2015. [Online]. Available: <http://dx.doi.org/10.1002/rob.21595>
- [34] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 778–792.
- [35] T. Krajník, P. Cristóforis, M. Nitsche, K. Kusumam, and T. Duckett, “Image features and seasons revisited,” in *Proc. IEEE Eur. Conf. Mobile Robots*, 2015, pp. 1–7.
- [36] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, “Image features for visual teach-and-repeat navigation in changing environments,” *Robot. Autom. Syst.*, vol. 88, pp. 127–141, 2016.
- [37] C. Linegar, W. Churchill, and P. Newman, “Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 90–97.
- [38] N. Carlevaris-Bianco and R. M. Eustice, “Learning visual feature descriptors for dynamic lighting conditions,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2769–2776.
- [39] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, “University of Michigan North Campus long-term vision and lidar dataset,” *Int. J. Robot. Res.*, vol. 35, pp. 1023–1035, 2015.
- [40] “Stromovka dataset,” 2017. [Online]. Available: <http://mobilerobotics.eu/datasets/stromovka>
- [41] T. Krajník *et al.*, “Simple, yet stable bearing-only navigation,” *J. Field Robot.*, vol. 27, pp. 511–533, Sep./Oct. 2010.
- [42] T. Krajník, J. P. Fentanes, M. Hanheide, and T. Duckett, “Persistent localization and life-long mapping in changing environments using the frequency map enhancement,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4558–4563.
- [43] D. Mishkin, M. Perdoch, and J. Matas, “Place recognition with WxBS retrieval,” in *Proc. CVPR Workshop Visual Place Recog. Changing Environ.*, 2015.
- [44] C. Cadena *et al.*, “Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [45] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil, “External localization system for mobile robotics,” in *Proc. Int. Conf. Advanced Robot.*, 2013. doi: 10.1109/ICAR.2013.6766520.
- [46] J. P. Fentanes, B. Lacerda, T. Krajník, N. Hawes and M. Hanheide, “Now or later? Predicting and maximising success of navigation actions from long-term experience,” *Int. Conf. Robot. Autom.*, May 2015, pp. 1112–1117. doi: 10.1109/ICRA.2015.7139315.
- [47] M. J. Santos, T. Krajník, and T. Duckett, “Spatio-temporal exploration strategies for long-term autonomy of mobile robots,” *Robot. Autom. Syst.*, Elsevier, 2016.



Tomáš Krajník received the Ph.D. degree in artificial intelligence and biocybernetics from Czech Technical University, Prague, Czech Republic, in 2012.

He is currently a Research Fellow at the Lincoln Centre of Autonomous Systems, Lincoln, School of Computer Science, University of Lincoln, U.K. His research interests include life-long autonomous navigation, spatio-temporal modelling, and aerial robots.



Jaime Pulido Fentanes received the Ph.D. degree in industrial engineering and automation from the University of Valladolid, Valladolid, Spain.

He is currently a Research Fellow at the Lincoln Centre for Autonomous Systems, School of Computer Science, University of Lincoln, Lincoln, U.K. where he is involved with multiple projects including the EU FP7 project STRANDS, which aims to enable a robot to achieve robust and intelligent behavior over long periods of time. His research interests include mobile robotics, mapping and navigation, and robot exploration.



João Machado Santos received the M.Sc. degree in electrical engineering and computers, with specialization in automation, from the Faculty of Sciences and Technology, University of Coimbra, Coimbra, Portugal, in 2013. He is currently working toward the Doctoral degree at the Lincoln Centre for Autonomous Systems, School of Computer Science, University of Lincoln, Lincoln, U.K.

He is currently involved with the project STRANDS within the FP7 framework, which aims to enable a robot to achieve robust and intelligent behavior over long periods of time.

His research interests include mobile robotics, mapping, localization, and exploration.



Tom Duckett received the Ph.D. degree in the AI Group at the University of Manchester, U.K.

He is currently a Professor of computer science at the University of Lincoln, Lincoln, U.K., where he also leads the Lincoln Centre for Autonomous Systems. He worked previously at the Centre for Applied Autonomous Sensor Systems, Örebro University, Örebro, Sweden, where he led the Learning Systems Laboratory. Prior to becoming an academic, he worked for several years as a programmer, developing and supporting software solutions for the fresh

food industry. His research interests include autonomous robots, artificial intelligence, and machine perception, with applications including agri-food and assistive technologies.

G

KEY ARTICLE [27] - ROBOTICS AND AUTOMATION MAGAZINE
2017

©[2016] IEEE. Reprinted, with permission, from Nick Hawes, The STRANDS Project:
Long-Term Autonomy in Everyday Environments, 2016.

The STRANDS Project

Long-Term Autonomy in Everyday Environments

Thanks to the efforts of the robotics and autonomous systems community, the myriad applications and capacities of robots are ever increasing. There is increasing demand from end users for autonomous service robots that can

operate in real environments for extended periods. In the Spatiotemporal Representations and Activities for Cognitive Control in Long-Term Scenarios (STRANDS) project (<http://strands-project.eu>), we are tackling this demand head-on by integrating state-of-the-art artificial intelligence and robotics research into mobile service robots and deploying these systems for long-term installations in security and care environments. Our robots have been operational for a combined duration of 104 days over four deployments, autonomously performing end-user-defined tasks and traversing 116 km in the process. In this article, we describe the approach we used to enable long-term autonomous operation in everyday environments and how our robots are able to use their long run times to improve their own performance.

Long-Term Autonomy in STRANDS

Autonomous robots come in myriad forms and can be used in a range of applications. With these differences, long-term autonomy (LTA) has a variety of meanings. For example, NASA's *Opportunity* rover has been autonomous for more than ten years on the surface of Mars; wave gliders can automatically monitor stretches of ocean for months at a time; and autonomous cars have completed journeys of thousands of kilometers. In this article, we restrict our contributions to mobile robots operating in everyday, indoor environments, such as offices and hospitals, and capable of performing a variety of



By Nick Hawes, Chris Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrová, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Körtner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Fülhammer, Michael Zillich, Markus Vincze, Eris Chinellato, Muhannad Al-Omari, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomáš Krajník, João M. Santos, Tom Duckett, and Marc Hanheide

Digital Object Identifier 10.1109/MRA.2016.2636359

Date of publication: 8 June 2017

1070-9932/17©2017IEEE TRANSLATIONS AND CONTENT MINING ARE PERMITTED FOR ACADEMIC RESEARCH ONLY.
PERSONAL USE IS ALSO PERMITTED, BUT REPUBLICATION/REDISTRIBUTION REQUIRES IEEE PERMISSION.
SEE [HTTP://WWW.IEEE.ORG/PUBLICATIONS_STANDARDS/PUBLICATIONS_RIGHTS/INDEX.HTML](http://www.ieee.org/publications_standards/publications/rights/index.html) FOR MORE INFORMATION.

service tasks. Across the various robots described previously, there are commonalities in low-level, short-term control algorithms (e.g., closed-loop motor control). Beyond this, the algorithms used to provide long-term, task-specific autonomous capabilities—and the hardware these algorithms control—vary greatly, according to application and environmental requirements. The challenges that distinguish indoor service robots from other LTA robots relate to both their environment and their task capabilities. Indoor task environments are less physically risky than outdoor environments, but they have a comparatively higher degree of short- to medium-term physical variability, e.g., moving objects such as people, doors, and furniture. You might argue that traffic is highly variable, but roads are generally similar to each other and the movement of vehicles is generally more predictable than the movement of people. In terms of application requirements, multipurpose service robots must be capable of predictable scheduled behavior while also being retaskable on demand with high availability and must be able to navigate in relatively confined, dynamic environments. This is in contrast to the largely restricted-purpose systems mentioned previously, such as rovers and wave gliders. Taken together, the set of requirements for indoor service robots presents unique challenges, and, thus, LTA in this context warrants dedicated research.

Given the state of the art, we consider *long-term* for a mobile service robot to mean at least multiple weeks of continuous operation. In very general terms, such LTA operation requires a robot's hardware and software to be robust enough to overcome failure to enable such operation. Such robustness can be provided by both design-time and run-time approaches. It is essential that LTA systems actively manage consumable resources (e.g., a battery) and that any autonomy-supporting capabilities (e.g., localization) are not adversely affected by long run times. While this latter point is common sense and may be common practice in many other technologies (from operating systems to cars), it has only recently been considered in autonomous robotics.

One reason it is challenging to design a service robot to meet the requirements of LTA is the impossibility of anticipating all situations in which it may find itself. If we can enable robots to run for long periods of time, however, then they will have opportunities to learn about the structure and dynamics of such situations. By exploiting the results of such learning, the robots should be able to increase their robustness further, leading to a virtuous cycle of improved performance and greater autonomy. It is this latter point that motivates STRANDS: To go beyond robots that simply survive to those that can improve their performance in the long term. It is within this context that this article makes its main contribution: the STRANDS Core System, a robotic software architecture that was designed for LTA service robot applications and has been evaluated across four end-user deployments. The STRANDS Core System contains a mix of common sense and novel elements that have enabled it to

support more than 100 days of autonomous operation. This is the first time all of these elements have been presented together, and this is the first presentation of metrics describing performance across deployments. Our approach is inspired by the work of Willow Garage [1] and the CoBot project [2], plus the pioneering work on the Rhino and Minerva systems (e.g., [3]). Our work is distinguished from previous work by the combination of multiple service capabilities in a single system capable of weeks-long continuous autonomous operation in dynamic indoor environments while using various forms of learning to improve system performance. Many other projects address one or two of these elements but not all four simultaneously.

Application Scenarios

To ensure our research meets the demands of end users, our work is evaluated in two application scenarios—security and care. Space does not permit a detailed explanation of the tasks in each scenario; instead, we cite other works that include further information on the tasks and technology from each scenario.

Our security plan was developed with G4S Technology. The aim of this system was to have a robot monitor an indoor office environment and generate alerts when it observed prohibited or unusual events. We completed two security deployments in which a mobile robot routinely created models of the environment's three-dimensional (3-D) structure [4], objects [5], and people [6]; modeled their changes over time; and used these models to detect anomalous situations and patterns. For example, we developed robot behaviors to detect when a human moves through the environment in an unusual manner [6], to build models of the arrangement of objects on desks [7], and to check whether fire exits have been left open. Long-term deployments are essential for these services to gather sufficient data to build appropriate models.

Our care scenario was developed with the Akademie für Alterforschung at the Haus der Barmherzigkeit. In this arrangement, the robot supported staff and patients in a large elder-care facility. We completed two care deployments in which a mobile robot guided visitors, provided information to residents, and assisted in walking-based therapies. In the care scenario, the robot serves users more directly, and, therefore, long-term system robustness is crucial, as is adapting to the routines of the facility. More information on this plan is available in [8] and [9].

Our robots have been operational for a combined duration of 104 days over four deployments, autonomously performing end-user-defined tasks and traversing 116 km in the process.

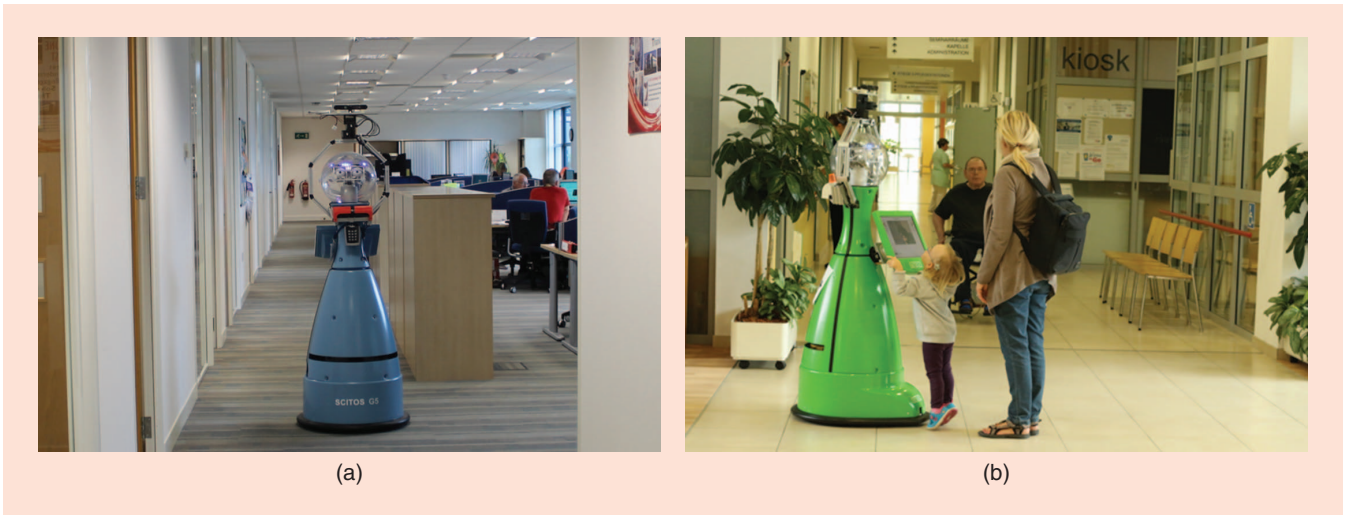


Figure 1. Two of the STRANDS MetraLabs SCITOS A5 robots in their application environments. (a) The robot *Bob* at G4S's Challenge House in Tewkesbury, United Kingdom. (b) The robot *Henry* in the reception area of Haus der Barmherzigkeit, Vienna.

Robot Technology

The systems reported in this article are developed in Robot Operating System (ROS), available under open-source licenses and binary packaged for Ubuntu (<http://strands-project.eu/software.html>) [2]. While

We also take advantage of the robot's need to regularly dock with a charging station by resetting the robot's position to this known location while it is docked.

the majority of our work is platform-neutral, all of our deployed systems are based on the MetraLabs SCITOS A5 robot (Figure 1). This is an industry-standard mobile robot capable of long run times (12 h on one charge) and autonomous charging. Our robots each have SICK S300 lasers in their bases (for localization, leg detection, and so on) and two Asus Xtion PRO RGB-D cameras, one at chest height

pointing downward (for obstacle avoidance) and the other on a pan-tilt unit (PTU) above the robot's head. The SCITOS has an embedded computer with an Intel Core i7 processor with 8 GB of random-access memory (RAM), to which we networked two additional computers, each with an Intel Core i7 processor and 16 GB of RAM.

The STRANDS Core System

The STRANDS Core System (Figure 2) is an application-neutral architecture for LTA in mobile robots. It is a combination of widely used components and components designed specifically for LTA. As mentioned previously, hardware and software robustness is essential for LTA. Hardware robustness is beyond the scope of our research; thus, we assume our software is running on an appropriate robot and computational platform. We address software component robustness through a mix of

strategies. During development, we encourage components to be designed in a way that makes the minimal assumptions about the existence of other components and services (e.g., by checking service existence before running). We also pay particular attention to error handling to ensure component-local errors and exceptions do not propagate unnecessarily. This allows components and whole subsystems to be brought up and down automatically. At run-time, we use built-in ROS functionality to automatically relaunch crashed components, and most subsystems run only when required, thus saving the energy and processing power and reducing opportunities for errors. We also use run-time topic monitoring to detect problems (e.g., low publish rates) and trigger component restarts. Finally, we run a continuous integration server that tests components and the whole system in isolation, on recorded data, and in simulation.

The overall performance of a mobile robot is constrained by its localization and navigation systems, so we use widely adopted ROS packages to provide state-of-the-art performance. At the start of a deployment, we build a fixed map from laser scans, localize in it with adaptive Monte Carlo localization, and navigate using the dynamic window approach (DWA) over 3-D obstacle information [3]. See <http://wiki.ros.org/navigation> for details on these techniques. While our use of a fixed map appears at odds with LTA in a dynamic environment, our environments are dominated by static features (e.g., walls), which prevent the robot's localization performance from degrading. We also take advantage of the robot's need to regularly dock with a charging station by resetting the robot's position to this known location while it is docked. This limits localization drift to that which can occur during time away from the dock.

We manually build a topological map on top of the fixed continuous map. We place topological nodes at key places in the environment for navigation (e.g., either side of a door) or for tasks (e.g., by a desk to observe). The topological map from our 2015 security deployment is shown in Figure 3. Edges in the

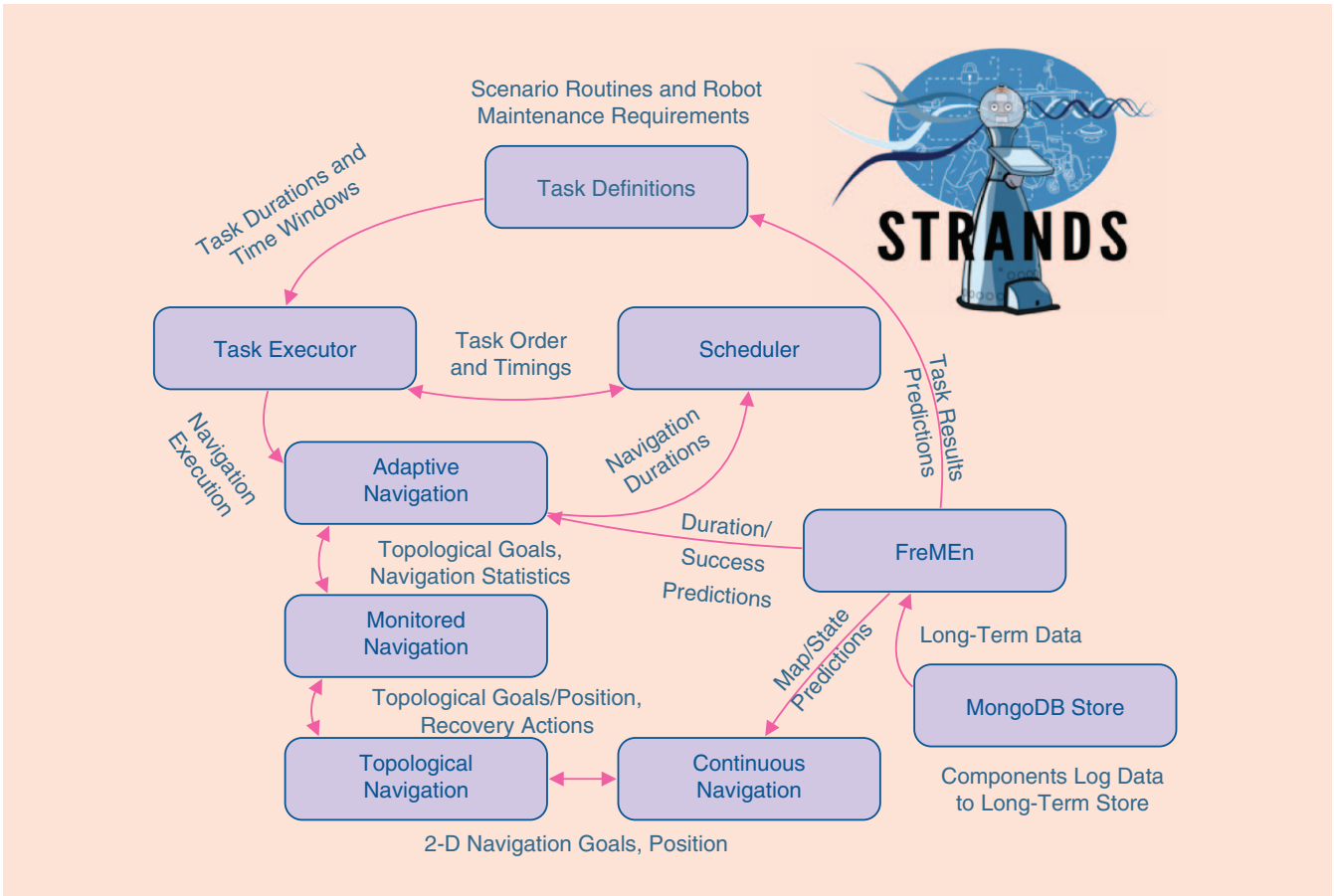


Figure 2. A schematic overview of the STRANDS Core System. 2-D: two-dimensional.

topological map are parametrized by the action required to move along them. In addition to DWA navigation, our system can perform door passing, docking with a charging station, and adaptive navigation near humans [10].

In our experience, navigation performance is a major determinant of the autonomous run time of a mobile robot. This is because navigation failures (e.g., getting stuck near obstacles) can result in the robot being unable to return to its charging station. The elements of the STRANDS Core System support LTA in the following ways: By constraining the robot’s movements to the topological map, we are able to restrict navigation to known good areas of the environment. We additionally restrict movement by marking areas of the static map as “no go” zones that cannot be planned through. Despite these restrictions, navigation failures still occur due to environmental dynamics (e.g., people walking in front of the robot). Therefore, edge traversals in the topological map are executed by a monitored navigation layer that can perform a range of recovery actions in the

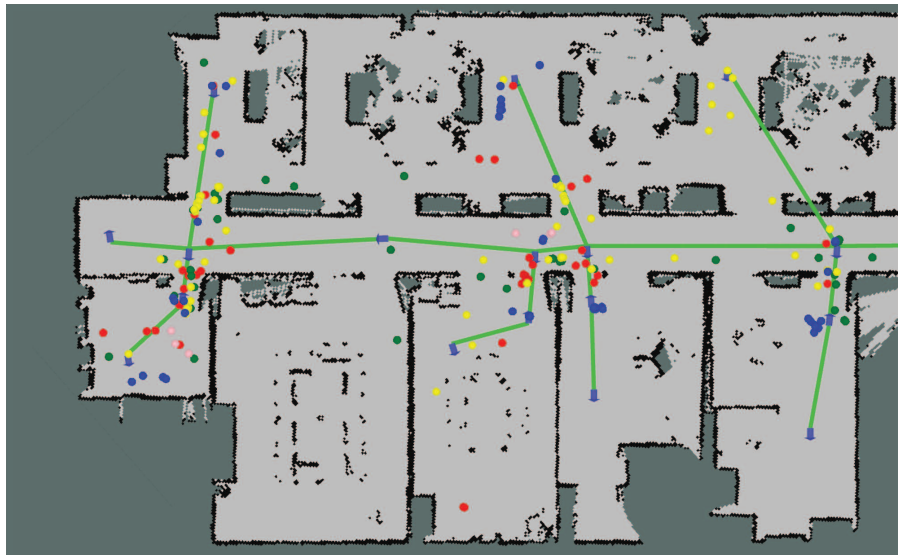


Figure 3. The map of the deployment area in Challenge House in Tewkesbury, United Kingdom, with the topological map superimposed. Also displayed are the locations where the robot successfully recovered from a navigation failure. Locations where the bumper was triggered are red, and green locations indicate nonbumper fails; the robot asked humans for help at these locations. Places where recoveries were performed by reversing along the previous path are marked in yellow, and recoveries performed by simply retrying are shown in blue.

event of failure (see the “Monitored Navigation” section). Topological route planning and execution is one area where our core system adapts to long-term experience, as described in the “Adaptive Topological Navigation” section.

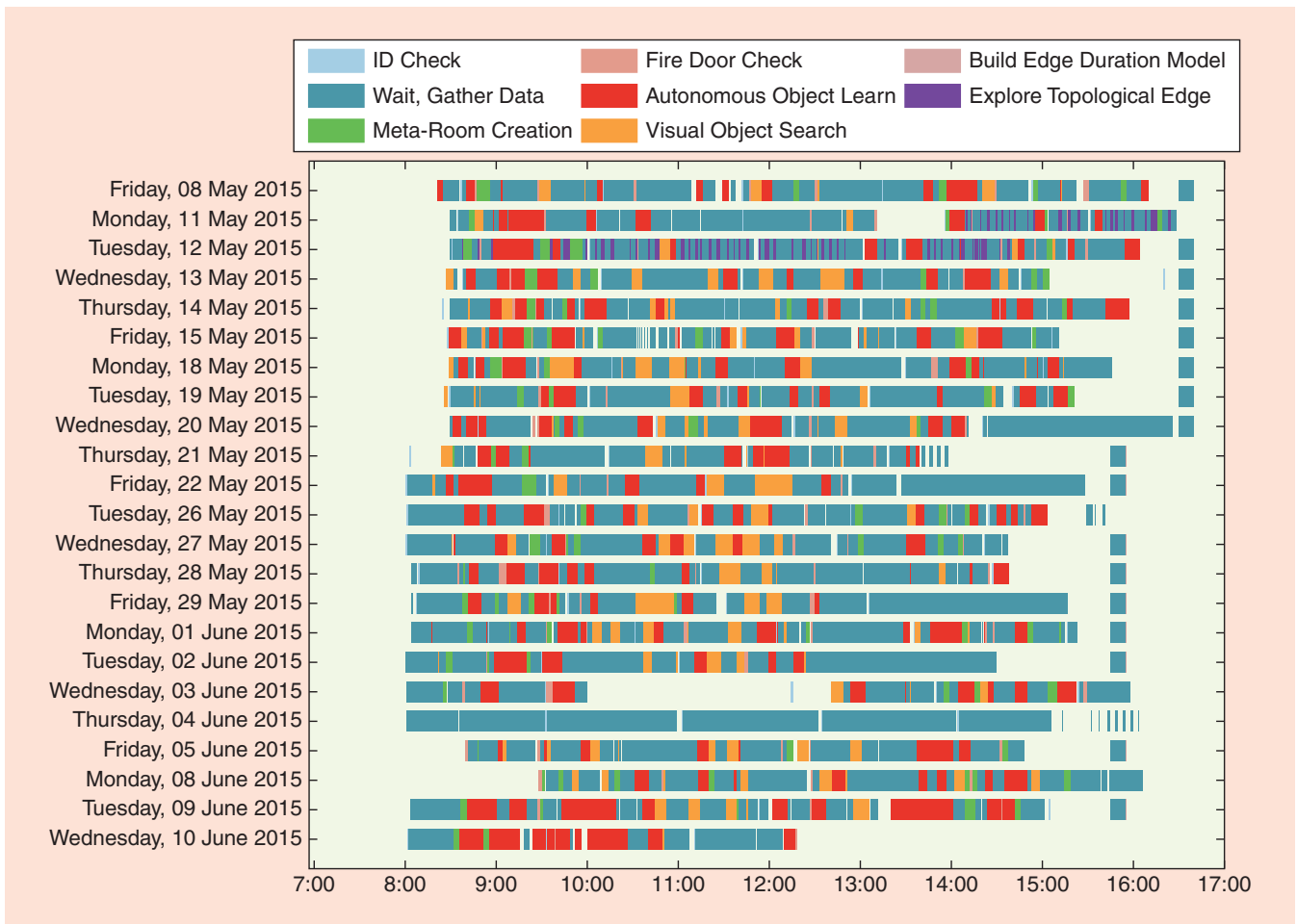


Figure 4. A plot of the tasks performed by the robot during the 2015 security deployment. White space represents times when the robot was not performing any tasks, which indicates that the robot was charging or a failure had occurred.

The main unit of behavior in our system is a task. Tasks represent something the robot can do (e.g., check whether a fire door is open, deliver information via a graphical user interface), and tasks have an associated topological location, a maximum duration, and a time window for execution. Our executive framework [11] schedules tasks to be executed within their time windows and manages task-directed navigation and execution. To prevent task failures from interfering with long-term operations, our framework detects task time-outs and failures, at which point it will stop or restart robot behaviors as necessary. Maintenance actions such as charging, batch learning, and database backups are all handled as tasks, allowing the executive framework to control most of the robot's behavior.

The data observed and generated (e.g., as intercomponent communication) by an LTA system is crucial for both learning and for monitoring and debugging the system.

This is essential for LTA as it enables the system to actively manage its limited resources. A plot of tasks from the 2015 security deployment can be seen in Figure 4.

Our system relies on separate pipelines for perceiving different elements of its environment: real-time multiperson RGB-D detection and tracking [12], visual object instance and category modeling and recognition [13], and 3-D spatiotemporal mapping [4]. This article does not cover our work on perceptually challenging tasks. Instead, we refer readers to other sources where we have exploited these perception pipelines, e.g., [5], [7], and [10].

The data observed and generated (e.g., as intercomponent communication) by an LTA system is crucial for both learning and for monitoring and debugging the system. We therefore use tools based on MongoDB (http://wiki.ros.org/mongodb_store) to save ROS messages to a document-oriented database. Database contents (e.g., observations of doors being opened or closed) can then be interpreted by the Frequency Map Enhancement (FreMEn) component [14], which integrates sparse and irregular observations into spatiotemporal models representing (pseudo) periodic environment variations. These can be used to predict future environment states (see the “Adaptive Topological Navigation” section).

Table 1. LTA metrics from the first four STRANDS system deployments.

	Care 2014	Security 2014	Care 2015	Security 2015	Total
Total Distance Traveled (km)	27.94	20.64	23.41	44.25	116.24
Total Tasks Completed	1,985	963	865	4,631	8,444
Maximum TSL	7 d, 3 h	6 d, 19 h	15 d, 6 h	28 d, 0 h	
Cumulative TSL	20 d, 19 h	21 d, 0 h	27 d, 8 h	35 d, 3 h	104 d, 7 h
Individual Continuous Runs	18	18	5	2	43
A%	38.80%	18.27%	53.51%	51.10%	

Metrics

So far, we have performed two evaluation deployments for each of the security and care scenarios. For each deployment, we monitored overall system performance against two metrics: the total system lifetime (TSL) and the autonomy percentage (A%). The TSL measures how long the system is available for autonomous operation and is reset if the system experiences an unrecoverable failure or needs an unrequested expert intervention (i.e., something that cannot easily be done by an end user on site). The A% measures how long the system was actively performing tasks as a proportion of the time it was allowed to operate autonomously; in our deployments, this is typically restricted to office hours. The motivation of the A% is that it is of little value to achieve a long TSL if the system does nothing. Neither the TSL nor the A% measures the quality of the services being provided. As this article focuses on LTA, we restrict our presentation to task-neutral but LTA-specific metrics. End-user evaluations of task-specific performance are ongoing and will be published in the future (see [8] and [9] for early evaluations from the care scenario).

Table 1 presents our systems’ LTA performance in 2014 and 2015. In 2014, we aimed for 15 days for the TSL; however, the longest run we achieved was seven days. Most of our system failures were caused by the lack of robustness of our initial software, leading to unrecoverable component behavior (crashes or deadlock states). This was fixed for our 2015 deployments by following the development approaches outlined in “The STRANDS Core System” section. In 2015, we targeted 30 days for the TSL, coming close with 28 days in the security deployment. This long run was terminated when the robot’s motors failed to respond to commands, an issue that has since been fixed with a firmware update. In the 2015 deployments, most failures were due to computer-related issues beyond the direct contributions of the project (e.g., USB drivers, power cables, network problems, and so on). Of the seven runs in 2015, one run was ended due to user intervention (a decorator powered off the robot), two due to bugs in our software, and the remaining four due to faults in software or hardware beyond our components.

The variations across deployments in terms of the number of tasks completed and distance traveled are largely attributable to the different types of tasks performed by the robots and the different environments in which they were

deployed. For example, information-serving tasks may take several minutes with very little travel, but door-checking tasks will be brief but will require the robot to travel both before and during the task.

Systems in the literature have delivered more autonomous time and distance cumulatively (i.e., accumulated across multiple robots and system runs), but we believe the 28-day run is the longest single continuous autonomous run of an indoor mobile service robot capable of multiple tasks. The most relevant comparison we can make is to the CoBots, which reported a total of 1,279.5 h of autonomy time, traversing 1,006.1 km [2]. This was achieved by four robots in 3,199 separate continuous autonomous runs over three years, at an average of 23 min and 0.31 km per run. They did not report the longest single continuous run (either in time or distance), but even an extremely long run for a CoBot would only be measured in hours, not days, because the CoBot does not have autonomous charging capabilities. In contrast, the STRANDS systems performed a total of 43 separate continuous runs, yielding a total of 2,545 h and 116 km over the four deployments, at an average of 2.7 km and 58 h 12 min per run. The varied durations of individual runs can be seen in Figure 5. Note that we use this data to provide a point of comparison.

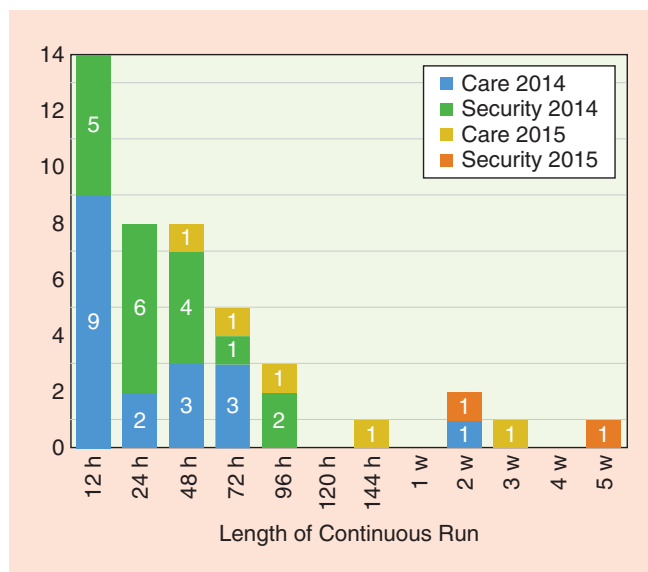


Figure 5. A histogram of individual continuous run lengths over the four STRANDS deployments.

Table 2. Classes of navigation failure, their associated recoveries, and the overall counts of successful and unsuccessful recoveries from these failures. Per-recovery counts are shown in Figure 6.

Failure	Recoveries	Successful	Unsuccessful	Total
Bumper pressed	Request help via screen and voice. Repeated request until recovered.	177	148	325
Navigation failure (no valid local or global path)	Sleep then retry; backtrack to last good position; request help via screen and voice. Repeated request until recovered.	707	993	1,700
Stuck on carpet	Increased velocities commanded to motors	16	247	263

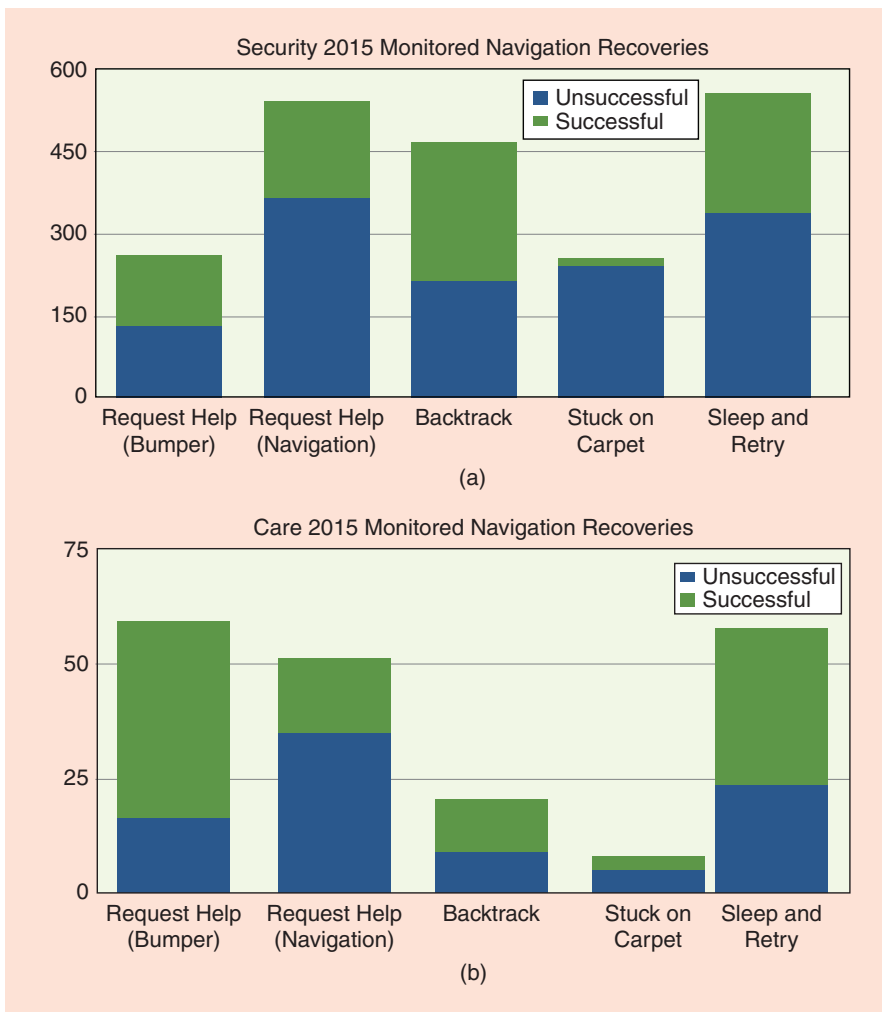


Figure 6. Per-recovery counts for the 2015 (a) security and (b) care deployments.

The two projects are targeting different metrics (i.e., total distance for CoBots and single-run duration for STRANDS); thus, the systems naturally have different performance characteristics.

Monitored Navigation

Given the huge variety of situations an LTA service robot will encounter, it is impossible to develop a navigation algorithm that will successfully account for all of them. We

therefore developed a framework that executes topological navigation actions and monitors them for failure. If a failure is detected, then the framework steps through a list of recovery behaviors until either the navigation action completes successfully or the list is exhausted, in which case, failure is reported back to the calling component. Failure types can be mapped to specific lists of recoveries. When the robot's bumper is pressed, a hardware cut-off prevents it from moving forward, which means that the robot must ask to be pushed away from obstructions by nearby humans. If the local DWA planner fails to find a path, then simply clearing the navigation costmap to remove transient obstacles may suffice. We also developed a backtrack behavior that uses the PTU-mounted depth camera to sense backward while the robot reverses along the path it took to the failure location. This is triggered when navigation fails and clearing the costmap does not overcome the failure.

Table 2 and Figure 6 present the recovery behaviors used in our 2015 deployments. Successful recoveries are those that were not followed by another failure within one minute or

1 m of travel; otherwise, they are unsuccessful. A successful recovery may be preceded by any number of unsuccessful recoveries. A sequence of unsuccessful recoveries can come from the monitored navigation system as it attempts recoveries that then fail or from the task execution framework unsuccessfully trying to navigate the robot to another task after a previous failure. Figure 3 shows where all the successful recoveries from our 2015 security deployment occurred. They are largely clustered around areas where it

was difficult to navigate, such as near doors and close to desks. This novel approach contributed significantly to the LTA performance of our systems, as each recovered failure could have potentially caused the end of a continuous run.

Adaptive Topological Navigation

While monitored navigation helps the robot recover from navigation failures, it does not help it to avoid them. To avoid navigation failures in the future, the robot's navigation experience is aggregated into a Markov decision process (MDP) automatically built from the topological map [15]. Using the MDP allows the system to model uncertainty over the success of the robot traversing an edge in the map and its expected duration. By learning models for these success probabilities and durations online, the robot is able to continually adapt its behavior to the environment in which it is deployed. Every time the robot navigates an edge, the duration and success of the traversal is logged to the robot's database. These logs are processed by FreMEN to produce a temporal predictive model that allows the actions of the MDP to be assigned probabilities and travel durations appropriate for the time of execution [11]. This MDP is then solved for a target location to produce a policy for topological navigation that prefers low-duration edges with high success probabilities (see [15] for details). This improves the system's robustness by making it avoid areas where it previously encountered navigation failures. This is only possible in an LTA setting where the robot runs repeatedly in the same environment.

Predicting Human–Robot Interaction

In the Haus der Barmherzigkeit care facility, our robot acted as an information terminal, using its touch screen to present the weather, daily menu, news, and so on, to staff and residents with potentially severe dementia. This behavior was scheduled as a task at different topological nodes in the care home. As we did not know in advance the locations and times people would prefer to interact with the robot, we allowed it to adapt its routine based on long-term experience. To achieve this, each node in the topological map was associated with a FreMEN model that represented the probability of someone interacting with the robot's screen at a given time. This was built from logs of screen interactions stored in MongoDB. These FreMEN models were used to predict the likelihood of interactions at given times and locations. These predictions were used by the robot to schedule where and when it should provide information during the day.

The schedule must satisfy two contradicting objectives common to many online active-learning tasks: exploration (to create and maintain the spatiotemporal models) and exploitation (using the model to maximize the chance of interacting with people). Exploration requires the robot to visit locations at times when the chance of obtaining an interaction is uncertain. Exploitation requires scheduling visits to maximize the chance of obtaining interactions. To tackle this trade-off, the schedule was generated using Monte Carlo sampling

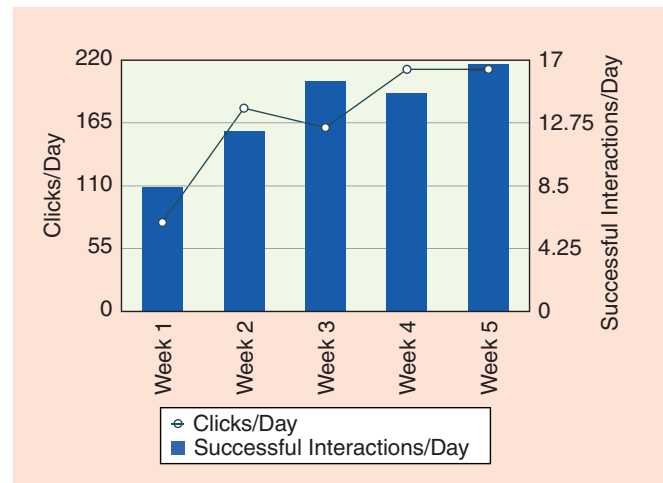


Figure 7. The results of the robot selecting interaction times and locations using FreMEN models learned during the 2015 care deployment.

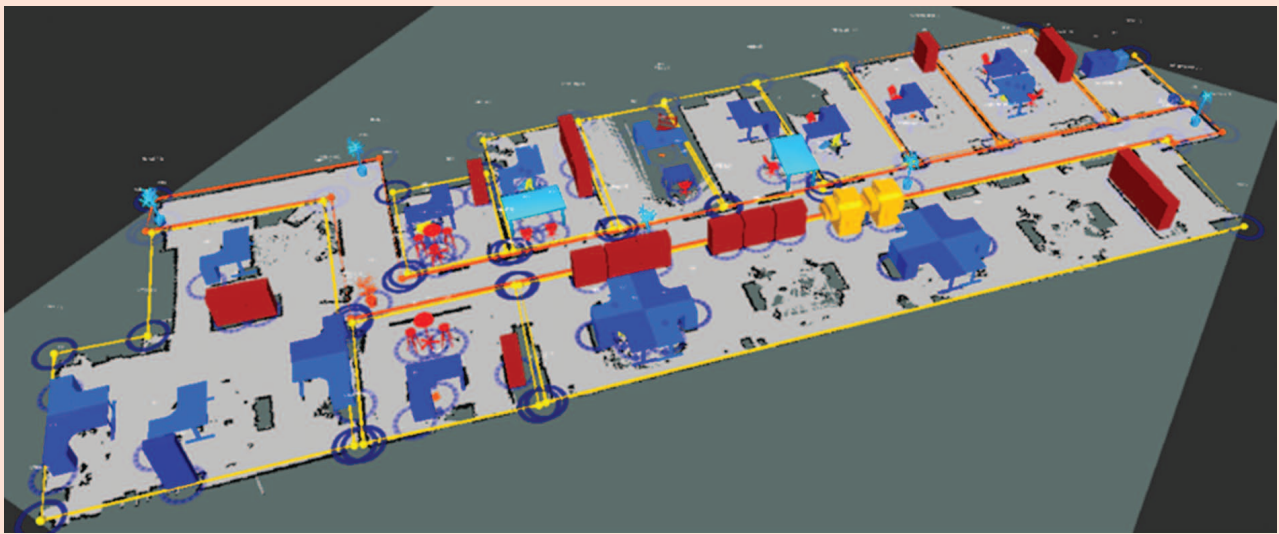
from the location/time pairs according to their FreMEN-predicted interaction probability (exploitation) and entropy (exploration). For more details see [16].

Figure 7 shows that the robot was able to increase the number of successful interactions (i.e., when information was offered and someone interacted with the screen) on average, per day, over the course of its deployment. Although we have no control group to compare against, our on-site observations indicate that the robot's choices had a positive effect. This demonstrates the ability of the system to improve its application-specific behavior from long-term experience.

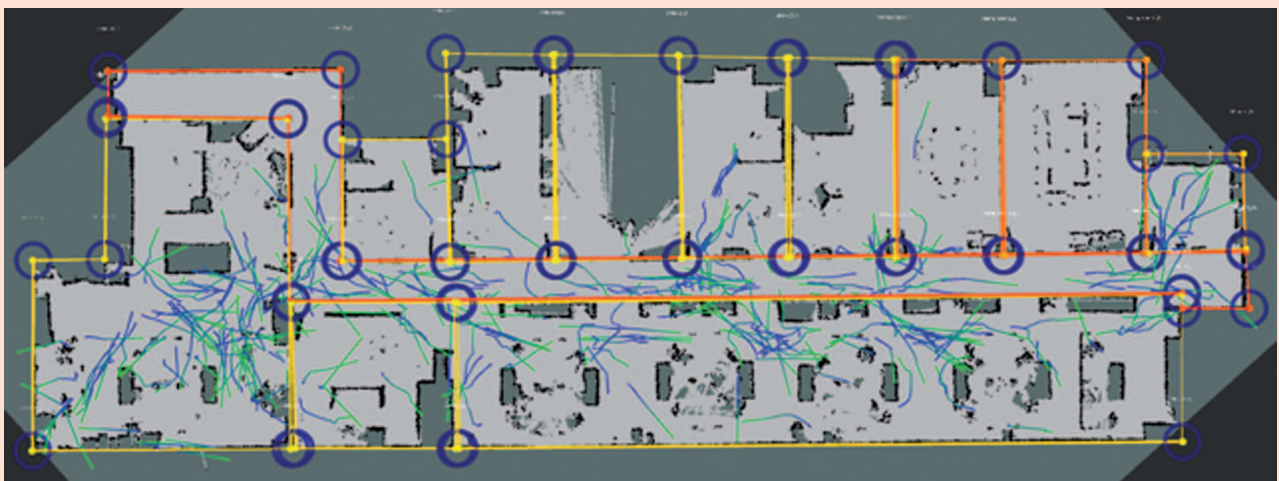
Activity Learning

In our security scenario, the robot had to learn models of normal human activity and then raise an alert if an observation deviated from this. We explored activity learning using walking trajectories [see Figure 8(b)]. Over the 2015 security deployment, the robot detected 42,850 individual trajectories. As described in [6], we used qualitative spatiotemporal activity graphs (QSTAGs) to generalize from individual trajectories to spatial and temporal relations between trajectories and landmarks in a semantic map [see Figure 8(a)]. QSTAGs ignore minor quantitative variations across trajectories but capture larger, qualitative changes. Every night, the robot created QSTAGs for a subset of all trajectories observed during the day (based on their displacement ratio). It then clustered these to create classes of movement activities. Some examples of the results can be seen in Figure 9.

During the day, an observation of a trajectory sufficiently far from any cluster center triggered a task to approach the tracked human and request confirmation of their identity using a card reader. To enable a fast response, it is important that the robot can accurately match the start of the trajectory to a cluster. Table 3 shows how the accuracy of predicting the cluster of a trajectory from an initial segment (20%) improves as more data is gathered over the robot's lifetime. This provides another example of how a robot can improve its application-specific performance once it can operate over long periods.



(a)



(b)

Figure 8. (a) The manually created semantic map from the 2015 security deployment. (b) Example human trajectories with lengths close to the average trajectory length of 2.44 m. Also pictured are the manually annotated room regions used for task planning, where the circles indicate the vertices of region polygons that were used to annotate the types of regions in the environment. The yellow regions indicate offices, the dark orange regions indicate meeting rooms, the light orange region is a kitchen, and the red region is a corridor. Blue to green lines indicate direction of movement.

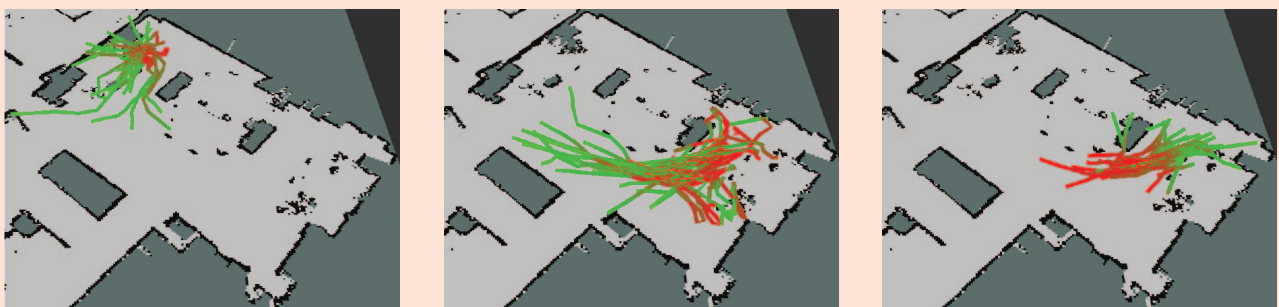


Figure 9. Trajectories belonging to three learned clusters in the region at the bottom left of Figure 8 (the direction of motion is red to green). These can be interpreted as two clusters of a desk-approaching activity and one cluster of a desk-leaving activity.

Table 3. Accuracy of activity cluster prediction on week 5 data from partial input trajectories.

Training Weeks (Number of Trajectories)	Number of Clusters	Recall	Precision	F-Score
Week 0 (342)	9	0.24	0.72	0.29
Weeks 0–1 (511)	12	0.43	0.54	0.44
Weeks 0–2 (707)	12	0.43	0.56	0.43
Weeks 0–3 (811)	10	0.43	0.71	0.49
Weeks 0–4 (1,016)	14	0.48	0.63	0.53

Conclusions and Future Work

The STRANDS Core System features a mix of design- and run-time approaches that allow it to deliver LTA in everyday environments. A key strategy for delivering long-term robustness is the monitoring of system behavior, from the individual component level up to navigation and task behaviors, as well as the ability to restart system elements on demand. This allows the system to cope with unexpected situations both internally and in the external environment. The system also uses the long-term experience of failures to learn to avoid these failures in the future. This approach improves navigation ability, and we hope to generalize this to other parts of the system. While these features provide a fundamental ability to operate autonomously for long durations in everyday environments, our robots currently have no way to manage failures that are more catastrophic, harder to predict, or both. For example, our systems have suffered from computer component failure and subtle networking issues. In the future, we would like to look at the use of redundancy and online reconfiguration, such as substituting a failing software or hardware component, coupled with more general failure detection approaches. Both of these topics have been extensively researched in robots and other systems.

Our robots are able to learn online from lengths of experiences from which no other robots to date have access. The discussed results demonstrate what we have always known from machine learning: More data improves performance. The novel element here is that a robot must be able to operate longer to gather additional data and must be able to make active choices about what data is gathered.

In the future, we will focus on the robot's ability to understand human activities, which are the major causes of environmental dynamics at most scales, and to actively close gaps in the knowledge it has already obtained from weeks of autonomous run time.

Acknowledgments

We wish to thank our project reviewers and project officers for their contributions to our research: Luc De Raedt, James Ferryman, Horst-Michael Gross, Olivier Da Costa, and Juha Heikkilä. The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement No. 600623, STRANDS.

References

- [1] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robotics Automation*, Anchorage, AK, 2010, pp. 300–307.
- [2] J. Biswas and M. Veloso, "The 1,000-km challenge: Insights and quantitative and qualitative results," *IEEE Intelligent Syst.*, vol. 31, no. 3, pp. 86–96, May 2016.
- [3] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proc. IEEE Int. Conf. Robotics Automation*, Detroit, MI, 1999, pp. 1999–2005.
- [4] R. Ambrus, J. Ekekrantz, J. Folkesson, and P. Jensfelt, "Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 5678–5685.
- [5] T. Faeulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 26–33, Jan. 2016.
- [6] P. Duckworth, Y. Gatsoulis, F. Jovan, N. Hawes, D. C. Hogg, and A. G. Cohn, "Unsupervised learning of qualitative motion behaviours by a mobile robot," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, Singapore, 2016, pp. 1043–1051.
- [7] L. Kunze, C. Burbridge, M. Alberti, A. Tippur, J. Folkesson, P. Jensfelt, and N. Hawes, "Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 2910–2915.
- [8] D. Hebesberger, C. Dondrup, T. Körtner, C. Gisinger, and J. Pripfl, "Lessons learned from the deployment of a long-term autonomous robot as companion in physical therapy for older adults with dementia: A mixed methods study," in *Proc. IEEE Int. Conf. Human-Robot Interaction*, New Zealand, 2016, pp. 27–34.
- [9] D. Hebesberger, T. Körtner, J. Pripfl, and M. Hanheide, "What do staff in eldercare want a robot for? An assessment of potential tasks and user requirements for a long-term deployment," in *Proc. Workshop on Bridging user needs to deployed applications of service robots*, Hamburg, Germany, 2015.
- [10] C. Dondrup, N. Bellotto, M. Hanheide, K. Eder, and U. Leonards, "A computational model of human-robot spatial interactions based on a qualitative trajectory calculus," *Robotics*, vol. 4, no. 1, pp. 63–102, Mar. 2015.
- [11] L. Mudrová, B. Lacerda, and N. Hawes, "An integrated control framework for long-term autonomy in mobile service robots," in *European Conf. Mobile Robots*, Lincoln, UK, 2015, pp. 1–6.
- [12] O. H. Jaffari, D. Mitzel, and B. Leibe, "Real-Time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *Proc. IEEE Int. Conf. Robotics Automation*, Hong Kong, 2014, pp. 5636–5643.
- [13] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "RGB-D object modelling for object recognition and tracking," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 96–103.
- [14] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *Proc. IEEE Int. Conf. Robotics Automation*, Hong Kong, 2014, pp. 3706–3711.
- [15] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 1511–1516.

[16] J. M. Santos, T. Krajnk, J. P. Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4-d spatio-temporal maps," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 684–691, July 2016.

Nick Hawes, School of Computer Science, University of Birmingham, United Kingdom. E-mail: n.a.hawes@cs.bham.ac.uk.

Chris Burbridge, School of Computer Science, University of Birmingham, United Kingdom. E-mail: c.j.c.burbridge@cs.bham.ac.uk.

Ferdian Jovan, School of Computer Science, University of Birmingham, United Kingdom. E-mail: fx345@cs.bham.ac.uk.

Lars Kunze, School of Computer Science, University of Birmingham, United Kingdom. E-mail: l.kunze@cs.bham.ac.uk.

Bruno Lacerda, School of Computer Science, University of Birmingham, United Kingdom. E-mail: b.lacerda@cs.bham.ac.uk.

Lenka Mudrová, School of Computer Science, University of Birmingham, United Kingdom. E-mail: lxm210@cs.bham.ac.uk.

Jay Young, School of Computer Science, University of Birmingham, United Kingdom. E-mail: j.young@cs.bham.ac.uk.

Jeremy Wyatt, School of Computer Science, University of Birmingham, United Kingdom. E-mail: jlw@cs.bham.ac.uk.

Denise Hebesberger, Akademie für Altersforschung am Haus der Barmherzigkeit, Austria, and Donau-Universitaet Krems, Austria. E-mail: Denise.Hebesberger@hausderbarmherzigkeit.at.

Tobias Körtner, Akademie für Altersforschung am Haus der Barmherzigkeit, Austria, and Donau-Universitaet Krems, Austria. E-mail: tobias.koertner@altersforschung.ac.at.

Rares Ambrus, KTH Royal Institute of Technology, Sweden. E-mail: rares.ambrus@gmail.com.

Nils Bore, KTH Royal Institute of Technology, Sweden. E-mail: nbore@kth.se.

John Folkesson, KTH Royal Institute of Technology, Sweden. E-mail: johnf@kth.se.

Patric Jensfelt, KTH Royal Institute of Technology, Sweden. E-mail: patric@kth.se.

Lucas Beyer, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: beyer@vision.rwth-aachen.de.

Alexander Hermans, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: hermans@vision.rwth-aachen.de.

Bastian Leibe, Rheinisch-Westfälische Technische Hochschule Aachen, Germany. E-mail: leibe@umic.rwth-aachen.de.

Aitor Aldoma, Technische Universität Wien, Austria. E-mail: aldoma@acin.tuwien.ac.at.

Thomas Fäulhammer, Technische Universität Wien, Austria. E-mail: faeulhammer@acin.tuwien.ac.at.

Michael Zillich, Technische Universität Wien, Austria. E-mail: zillich@acin.tuwien.ac.at.

Markus Vincze, Technische Universität Wien, Austria. E-mail: vincze@acin.tuwien.ac.at.

Eris Chinellato, Faculty of Science and Technology, Middlesex University London, United Kingdom. E-mail: e.chinellato@mdx.ac.uk.

Muhannad Al-Omari, University of Leeds, United Kingdom. E-mail: scmara@leeds.ac.uk.

Paul Duckworth, University of Leeds, United Kingdom. E-mail: scpd@leeds.ac.uk.

Yiannis Gatsoulis, University of Leeds, United Kingdom. E-mail: ygatsoulis@leeds.ac.uk.

David C. Hogg, University of Leeds, United Kingdom. E-mail: d.c.hogg@leeds.ac.uk.

Anthony G. Cohn, University of Leeds, United Kingdom. E-mail: A.G.Cohn@leeds.ac.uk.


Christian Dondrup, University of Lincoln, United Kingdom. E-mail: cdondrup@gmail.com.

Jaime Pulido Fentanes, University of Lincoln, United Kingdom. E-mail: jpulidofentanes@lincoln.ac.uk.

Tomáš Krajník, University of Lincoln, United Kingdom. E-mail: tkrajnik@lincoln.ac.uk.

João M. Santos, University of Lincoln, United Kingdom. E-mail: jsantos@lincoln.ac.uk.

Tom Duckett, University of Lincoln, United Kingdom. E-mail: tduckett@lincoln.ac.uk.

Marc Hanheide, University of Lincoln, United Kingdom. E-mail: marc@hanheide.net. 

H

KEY ARTICLE [32] - ROBOTICS AND AUTOMATION LETTERS
2017

©[2016] IEEE. Reprinted, with permission, from Jo ao Santos, Lifelong Information-Driven Exploration to Complete and Refine 4-D Spatio-Temporal Maps, 2017.

Lifelong Information-Driven Exploration to Complete and Refine 4-D Spatio-Temporal Maps

João Machado Santos, Tomáš Krajník, Jaime Pulido Fentanes, and Tom Duckett

Abstract—This letter presents an exploration method that allows mobile robots to build and maintain spatio-temporal models of changing environments. The assumption of a perpetually changing world adds a temporal dimension to the exploration problem, making spatio-temporal exploration a never-ending, life-long learning process. We address the problem by application of information-theoretic exploration methods to spatio-temporal models that represent the uncertainty of environment states as probabilistic functions of time. This allows to predict the potential information gain to be obtained by observing a particular area at a given time, and consequently, to decide which locations to visit and the best times to go there. To validate the approach, a mobile robot was deployed continuously over 5 consecutive business days in a busy office environment. The results indicate that the robot's ability to spot environmental changes improved as it refined its knowledge of the world dynamics.

Index Terms—Mapping, service robots.

I. INTRODUCTION

RECENT improvements in the ability of mobile robots to operate safely in human populated environments have allowed their deployment in households, offices and public buildings such as museums and hospitals. However, the structure of these environments is typically not known a priori, which requires the robots to build their own models of their operational environments. Moreover, natural environments tend to change over time, which means that to achieve long-term autonomous operation, robots must also update their environment models as a part of their daily routine.

While the problem of acquiring spatial representations of the environment, known as robotic mapping and exploration, has been addressed by many researchers, building and maintaining dynamic spatio-temporal environment representations has been addressed only recently. Several recent works demonstrated that explicit representation of the environment changes improves the performance of mobile robot operation in long-term scenarios [1]–[5]. However, these works were concerned with the problem of lifelong mapping, where the environment model is built in a passive way, and not lifelong exploration, where a robot decides where and when to gather data in order to complete and refine its environment model.

Our work addresses the problem of acquiring and maintaining a dense spatio-temporal environment model during

Manuscript received August 31, 2015; accepted December 18, 2015. Date of publication January 12, 2016; date of current version February 29, 2016. This paper was recommended for publication by Associate Editor J. Nieto and Editor C. Stachniss upon evaluation of the reviewers' comments. This work was supported by the EU ICT project Grant 600623 'STRANDS.'

The authors are with the Lincoln Centre for Autonomous Systems, University of Lincoln, Lincoln LN6 7TS, U.K. (e-mail: jsantos@lincoln.ac.uk).

Digital Object Identifier 10.1109/LRA.2016.2516594

long-term operation. We show that application of information-theoretic scheduling methods to time-dependent probabilistic environment representations results in a continuously improving exploratory behaviour, which evolves with the knowledge of the environment dynamics. Thus, the method allows the mobile robot to create, maintain and refine its environment models as a part of its daily routine and improve the robot's efficiency in performing other tasks at the same time. This ability is essential for long-term mobile robot operation in changing environments.

The presented work is based on a method that integrates sensory data captured at different locations and times into a dense spatio-temporal model, which represents the uncertainties of environment states as probabilistic functions of time. The probabilistic representation of the environment states allows calculation of the environment's spatio-temporal entropy, which can be used to predict the amount of information that the robot would obtain by observation of a given location at a particular time. Thus, the robot can schedule the times and locations of its observations in order to increase its chances of observing environmental changes, and thus to improve its knowledge of the environment dynamics.

To evaluate the method we compare it to a standard exploration method during 5-day-long simulated and real-world experiments performed in a human-populated environment.

II. RELATED WORK

In order to plan its actions in an intelligent way, a mobile robot needs a model of its operational environment. The quality of its internal model has a significant impact on the robot's ability to localise itself and navigate to the desired locations. Thus, most of the previous research was aimed at developing efficient environment representations.

Two of the most popular representations are metric and topological maps. Perhaps the most known and used metric map is the occupancy grid [6]. The main drawback of occupancy grids is their low-memory efficiency since they represent large, empty areas of the environment by a large number of empty cells. The aforementioned models represent the environment as a static structure, ignoring the environment dynamics. Consequently, the localization and navigation errors will accumulate over time as the environment changes. Thus, the development of world representations that can model or adapt to the environment dynamics improves the robot performance in changing environments [7], [8].

Some authors have developed approaches that can cope with dynamics without explicitly representing them [2], [4], [9],

[10]. Biswas *et al.* [11] present a novel non-Markov localization algorithm that classifies different types of unmapped objects according to their dynamics, providing local and global corrections to the environment model and thus more accurate localization than systems relying on a static world assumption. Similarly, [12] presents the concept of Dynamic Pose Graph SLAM, which aims to improve robot localization in changing environments. This method builds and maintains a pose graph of the environment by detecting changes after several measurements taken at the same locations. The pose graph is then edited in order to be more consistent with the current world state.

Other authors [3], [1] have focused on models that explicitly represent the environment changes and try to identify patterns. In [1], the robot's operational environment is modelled at multiple timescales. Churchill and Newman [2] use a vision system to group several distinct observations into 'experiences' of the same spatial locations, which improves long-term mobile robot localisation in outdoor environments. Tipaldi *et al.* [4] represent the states of the environment components (cells of an occupancy grid) with Hidden Markov Models and show that their method improves the robustness of localization. In [13], each cell in the occupancy grid stores not only the probability of it being occupied, but also the likelihood of the cell to change after a given time.

The aforementioned works focus on solving mobile robot localization in changing environments during long-term scenarios but the mapping task itself is done passively as it does not plan the locations and times to visit. Hence these approaches do not guarantee a complete model of the environment, which is vital for other abilities crucial to a robot such as planning.

Exploration approaches are aimed at guaranteeing completeness of the model by giving the robot the ability to autonomously create an accurate model of its operational environment. These methods typically focus on creating a complete and accurate world model in the shortest possible time. In frontier-based approaches [14], [15], the frontier is defined as the boundary between the known and unknown parts of the environment, and the robot plans its path in order to remove all the frontiers. While these strategies ensure the map's completeness, they do not consider map quality.

Next Best View strategies optimise several criteria, e.g., the estimated time to reach a given location and the amount of information expected to be gathered there [16]. Typically information gain is calculated as the environment uncertainty reduction achieved by incorporating the measurements from given positions. The reduction of uncertainty is typically calculated by means of entropy [17]. The lower the entropy of the environment model, the more it reflects the actual environment state.

Stachniss *et al.* [18] presented an information-gain based exploration framework that integrates not only uncertainties of the map, but also the uncertainties of the robot's localization. The method uses a Rao-Blackwellized particle filter to build the map and an entropy reduction method to plan the next location to be visited by the robot. In [19], the authors present a Next Best View approach to build a 3D model of an outdoor scenario while maximising the model's quality and optimising the robot's trajectory.

Most exploration approaches focus on building the initial map and do not deal with its maintenance over time, treating environment dynamics as unwanted noise, which means that the model loses its validity over time. To deal with this a related family of algorithms aims at creating models of the environment that allow them to predict where and when to make observations of specific phenomena within the environment. They achieve this by reasoning about the best times and paths to take. Typically, these algorithms rely on Gaussian Processes [20]–[22], which allow the robot to learn patterns in the environment. Other approaches represent the dynamics of the environment states based on the assumption that some of the environment variations observed are caused by routines performed by humans [23]. While the first approach focuses on building a model of sparse environmental phenomena as the main task of the robot, the latter focuses on updating a model of the environment that is used by the robot itself to improve its performance while executing its daily duties. The results in [23] showed that this approach could predict the environment changes, allowing the robot to better plan where and when to perform exploration. However, in this approach the topology of the environment was known *a priori*.

Our method builds on the concept of spatio-temporal exploration presented in [23], which builds frequency-enhanced spatio-temporal models from sparse and non-uniform observations and examines the performance of various exploration strategies and dynamic models. However, the prior work in [23] was based on several simplifications that make its real-world use difficult: it assumes that the topology of the environment is known *a-priori* and it neglects the fact that navigating between different locations requires different time durations. In other words, in [23], the robot simply selects which pre-defined topological locations should be visited at particular times in order to create and maintain local dynamic models on top of an *a-priori* known topological structure. The work presented here describes a complete exploration pipeline that starts without any *a-priori* knowledge about the robot's environment. The locations to be observed are not selected from a pre-defined set as in [23], but the robot infers the locations from the 3D structure itself. Thus, it considers not only the information gain obtained by visiting a given location, but also its reachability and the time it takes to navigate there. This results in a life-long exploration system that allows to create and maintain global 4D spatio-temporal representations of real, changing environments without prior knowledge of their topology. This letter also includes substantial new experimental work including extensive ground-truth-based evaluations. Nevertheless, our previous work indicated that the Monte Carlo exploration strategy outperformed other strategies in terms of the resulting model accuracy.

III. EXPLORATION SYSTEM DESCRIPTION

Robotic exploration methods usually consist of two alternating phases: planning and mapping. Considering that the environment is constantly changing, both planning and mapping have to take into account the notion of time. Thus, 3D mapping has to explicitly model the environment dynamics

and becomes “4D mapping”. The planning has to determine not only which locations to explore, but also when to perform the exploration. Other activities which form part of the robot’s daily routines may also be scheduled here, since a robot in a real-world application would have to balance its exploration activities with other activities that exploit the current spatio-temporal knowledge.

Our exploration system is composed of five main modules: the *Spatio-Temporal Model* that maintains the environment map, the *Scheduler* that determines the robot activity, the *Planner* that calculates which locations are to be explored, the *Executioner* that acts as a bridge between these modules, and the *Robot’s* navigation and sensing systems. The robot’s activity consists of separate exploration tours during which the robot leaves its charging station, navigates to a set of locations, where it uses its depth camera to observe the environment, and finally docks to its charging station using a precise marker-based localization method described in [24]. In this section, we first provide an overview of the exploration system and then details of its main modules.

A. System Overview

The overall system structure and its most important data flows are shown in Figure 2. Every 24 hours at midnight, the *Scheduler* sets up an activity plan for the upcoming day, which is partitioned into several time slots of the same duration. To determine which time slots are to be used for exploration and which ones to use for charging, it uses the *Planner* and the *Spatio-Temporal Map* to estimate how much information would be obtained by performing exploration at each of the time slots. In particular, the *Scheduler* sends the start time of a particular time slot to the *Spatio-Temporal Map* and the number of locations to visit to the *Planner*. The *Spatio-Temporal map* then predicts the probability and entropy of the environment states for the given time and passes the model to the *Planner*. The *Planner* then generates a sequence of candidate locations to visit, queries the *Spatio-Temporal Map* for the expected information gain at those positions and the *Reachability Map* for the probability that the robot will be able to navigate to those locations. The *Planner* then selects a number of locations to visit, where the number is given by the *Scheduler*, and reports the overall information gain back to the *Scheduler*. Based on the estimated information gain for each time slot, the *Scheduler* decides which time slots are to be used for exploration and which ones to use for recharging. The schedule-generation process is computationally expensive, mainly because the robot has to calculate the potential information gains across many locations and times. The entire schedule-generation process takes approximately two minutes and is performed during recharging. While the generated schedule ensures that the robot will tend to explore the environment when it is more likely to exhibit changes, the plans (sequences of points) generated by the *Planner* during the process of schedule generation might not be suitable at the time of their execution because the environment might change in a way that was not originally predicted.

Thus, at the beginning of each time slot allocated for exploration, the *Scheduler* queries the *Planner* for a new plan.

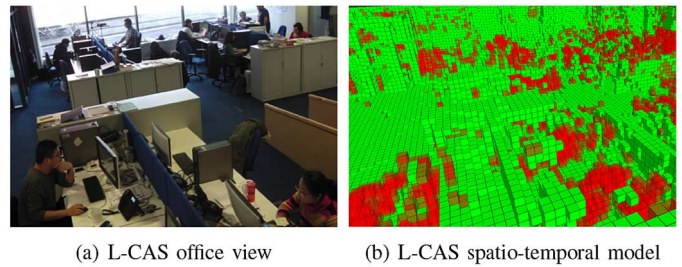


Fig. 1. Spatio-temporal occupancy grid of the Lincoln Centre for Autonomous Systems (L-CAS) office. The static cells are in green and cells that exhibit daily periodicity are in red.

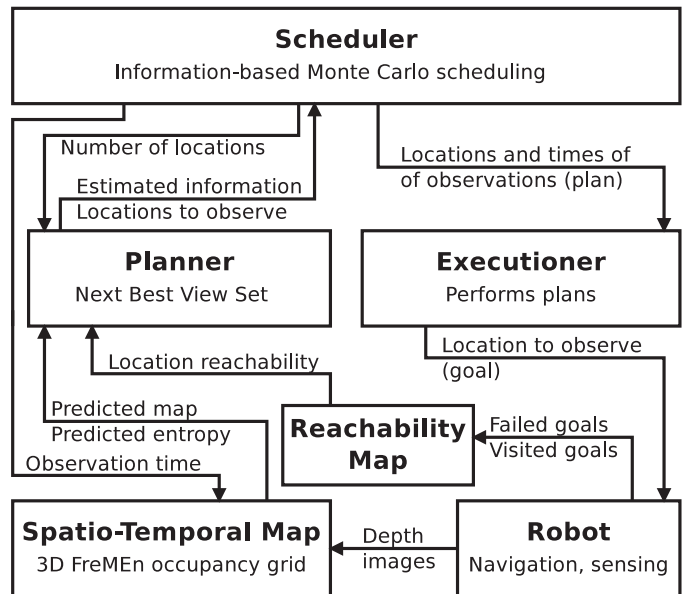


Fig. 2. Exploration system modules and main data flows.

The *Spatio-Temporal Map* predicts a temporary 3D occupancy grid, which is used to estimate the information gain and the *Reachability Map* for the given time. The *Planner* uses this information to decide which locations to visit and the *Executioner* determines their order and passes these goals one-by-one to the *Robot’s* navigation system. The *Robot* monitors whether the required locations (goals) were reached and passes this information to the *Reachability Map*. If a goal is reached successfully, the *Robot* uses its pan-tilt unit and depth camera to update the temporary 3D grid using the method in [25] and marks which cells were observed. After each 3D sweep, the updates made in the temporary 3D grid are propagated to the *Spatio-Temporal Map* using Equation (2).

B. Spatio-Temporal Map

The Spatio-Temporal map used in our work is based on a uniformly-spaced 3D occupancy grid extended by the Frequency Map Enhancement (FreMEn) concept [26]. The authors of FreMEn argue that the states of world models of human populated environments are influenced by human activities that – in a long-term perspective – tend to exhibit regular patterns, which causes some of the states’ dynamics to be regular as well. FreMEn attempts to capture and model

the periodicities by modeling the states' dynamics by their frequency spectra. In short, FreMEn uses the Fast Fourier algorithm to transform past observations of a given environment state, which is a binary function over time $s(t)$, to the spectral domain $S(\omega)$ and stores the most prominent spectral components of $S(\omega)$ in a sparsely-represented set $P(\omega)$. The inverse Fourier transform of $P(\omega)$ is then interpreted as the probability $p(t)$ of the original state $s(t)$ at time t . Application of FreMEn has been shown to improve mobile robot performance in feature-based localization [3], topological navigation [7] and robot search [8]. However, the Fast Fourier Transform algorithm used in [26] requires that the state $s(t)$ is sampled on a regular basis, which conflicts with the requirements of spatio-temporal exploration.

Thus, we propose a different scheme of transformation between the time domain $s(t)$ and the frequency domain $S(\omega)$. We assume that the spectral representation $P(\omega)$ of a state $s(t)$ consists of a small number of frequencies ω_i , phase shifts φ_i and amplitudes α_i . The probability $p(t)$ of the state $s(t)$ can be calculated as

$$p(t) = \varsigma(\alpha_0 + \sum_{i=1}^n \alpha_i \cos(\omega_i t + \varphi_i)), \quad (1)$$

where α_0 corresponds to the 'static' probability of the state $s(t)$, n is the number of periodicities modelled and $\varsigma()$ ensures that the result of Equation (1) is bounded between 0 and 1. To reflect the fact that we cannot be absolutely certain when predicting a given state, function $\varsigma()$ limits $p(t)$ between 0.05 and 0.95.

To obtain the parameters ω_i , φ_i and α_i from m measurements of the state s taken at times t_k , we first calculate the value of α_0 as the arithmetic mean of all past observations $s(t_k)$. Then we create a set of candidate frequencies Ω , which represent the periodicities of the hidden processes that affect the state $s(t)$. Finally, we establish the amplitudes α_c and phase shifts φ_c as

$$\begin{aligned} \alpha_c &= \left| \sum_{k=1}^m (s(t_k) - \alpha_0) e^{-j2\pi t_k \omega_c} \right|, \\ \varphi_c &= \arg\left(\sum_{k=1}^m (s(t_k) - \alpha_0) e^{-j2\pi t_k \omega_c} \right), \end{aligned} \quad (2)$$

where ω_c are elements of the set Ω .

Then, we order the frequencies ω_c according to their amplitude α_c , select the first n of them and store these as parameters ω_i , φ_i and α_i , which are used in Equation (1). Note that unlike the traditional Fast Fourier Transform used in [26], Equation (2) allows to update the spectral model as new observations of the state $s(t_k)$ are obtained. While faster to calculate and allowing for non-uniform sampling, the proposed representation does not ensure precise reconstruction of the original sequence $s(t)$, but typically results in $\sim 2\%$ reconstruction error. For details, see our previous work on frequency-based representations presented in [26] and [23].

Our *Spatio-Temporal Map* applies the FreMEn concept to occupancies of cells in a 3D occupancy grid. Thus, each cell contains its own set $P(\omega)$ that allows to calculate the probability of the cell's occupancy for any given time. This model is

updated by Equation (2) every time the cell is observed and its state $s(t_k)$ is measured.

Both Equations (1) and (2) are derived from the continuous formulation of the Fourier transform in [27], but unlike the classic discrete Fourier transform (DFT), Equations (1) and (2) simply do not assume time-uniform sampling of the state $s(t)$. In the case of uniform sampling with period Δt , i.e. $t_k = k\Delta t$, Equation (2) would become equivalent to the standard DFT.

1) Temporal Model Design: The set Ω of candidate frequencies in Equation (2) defines which periodicities will potentially be captured by our model. The elements of Ω can be chosen arbitrarily, but one should consider that larger Ω enables a finer representation of time at the expense of higher memory consumption of the spatio-temporal model. In our experiments, the set Ω consist of 24 elements ω_i , which are distributed in the same way as in the traditional FFT, i.e. $\omega_i = (24 \times 3600)/i$. This allows to model several periodicities ranging from one day to one hour. To model spatio-temporal dynamics of office-like environments, one could extend the set Ω by adding day-to-week periodicities as in [23]. However, this would require continuous operation of the robot for several weeks, which we hope to achieve during 2016.

The parameter n in Equation (1) determines how many periodicities of Ω are actually considered in the state prediction. Our previous work indicates that a good choice of n is 2, which typically results in modelling week- and day-long periodicities in indoor environments and year- and day-long cycles outdoors. Note that setting n to 0 means that the probability $p(t)$ becomes a constant as in traditional spatial-only representations. Similarly, non-periodic dynamics will cause the coefficients α_i calculated by Equation (2) to be close to 0, which will cause $p(t)$ to be almost constant as well.

C. Predicting the Information Gain

Since the aforementioned model can predict the probability of each cell being occupied, it also allows to estimate the amount of information that the robot obtains by observing the particular cell at a given time. The amount of information $I(t)$ obtained by observing a single cell at time t can be calculated as the difference between the cell's a-priori entropy $E(t)$ and a-posteriori entropy E_r , i.e. $I(t) = E(t) - E_r$, which are functions of the cell's occupancy probability before and after the observation. Since the cell's occupancy probability is considered as a function of time and we assume that the robot observes a given cell long enough to determine its state with certainty $p_c = 0.95$ (i.e. the probability of the cell being occupied after the observations becomes either 0.05 or 0.95), the expected information gain at time t is

$$\begin{aligned} I(t) &= -p(t) \log_2 p(t) - (1 - p(t)) \log_2 (1 - p(t)) \\ &\quad + p_c \log_2 p_c + (1 - p_c) \log_2 (1 - p_c), \end{aligned} \quad (3)$$

where $p(t)$ is the probability of occupancy of a given cell at time t calculated by Equation (2). Using the predicted occupancies and entropies, the *Spatio-Temporal map* allows to estimate the amount of information that the robot will obtain by observing a particular part of the environment at a particular time using its depth camera. Since our robot uses its

pan-tilt unit to create a 360° ‘sweep’ of its surroundings, the *Spatio-Temporal Map* implements a function that can estimate the obtained information given the robot position and the time of observation.

D. Reachability Map

Although the ability of the robot to reach individual locations of the environment can be inferred by the *Planner* from the environment’s spatio-temporal representation, some locations might not be reachable due to factors that are not included in the spatio-temporal model, such as transparent obstacles or objects with dimensions smaller than the spatio-temporal grid resolution. To reflect that, the exploration system maintains a *Reachability Map*, which is a 2D (50 × 50 cm) grid with cells that contain the robot’s success rate over the last five attempts to reach that particular location. This information is taken into account when the exploration plans are calculated.

E. Locations to Observe

We assume that moving to and observing one location takes approximately two minutes. Taking into account the time needed to dock to and leave the charging station, the robot can visit 6 locations in a 15-minute time slot.

To determine which locations are to be visited during a given time slot, the *Planner* first generates a uniform 2D grid of candidate positions $x_i, y_i \in \mathcal{C}$ that cover the operational environment. Then, it sends these positions to the *Reachability Map*, which returns the probability that the robot will be able to reach these positions, i.e. the *Planner* will obtain a reachability $p_r(x_i, y_i)$ for each candidate location (x_i, y_i) . If a position (x_i, y_i) is reachable, i.e. $p_r(x_i, y_i) > 0$, the *Planner* forwards the position (x_i, y_i) to the *Spatio-Temporal Map*, which uses the predicted 3D grid to estimate which cells are likely to be observable by the robot’s depth camera from the position (x_i, y_i) . The *Spatio-Temporal Map* sums the information gain of these cells using Equation (3) and reports it to the *Planner* as $I_C(x_i, y_i)$. This allows the *Planner* to create an evaluation $E(x_i, y_i)$ of each candidate location as

$$E(x_i, y_i) = p_r(x_i, y_i)I(x_i, y_i). \quad (4)$$

Once Equation (4) has been calculated for every (x_i, y_i) , the *Planner* starts to generate the locations to visit. First, the *Planner* finds the global maximum $E_{max}(x_j, y_j)$ of $E(x_i, y_i)$, adds (x_j, y_j) and E_{max} to the set of goals \mathcal{G} and sets $E(x_j, y_j)$ to 0. To take into account the fact that the cells observable from (x_j, y_j) are also visible from neighbouring locations but observations at locations close to (x_j, y_j) would not provide the same expected information, the values of $E(x_i, y_i)$ in the vicinity of (x_j, y_j) are decreased proportionally to their proximity to (x_j, y_j) . The aforementioned two steps, i.e. maxima search and suppression of the information gain estimates at the neighbouring locations, are repeated until the number of goals in the set \mathcal{G} equals the number of locations requested by the *Scheduler*. Then, the *Planner* calculates the sum E_G of information gains $E_{max}(x_j, y_j)$ in \mathcal{G} and reports the value of E_G to the *Scheduler* along with the locations in \mathcal{G} .

F. Generating the Schedule

Once the *Scheduler* obtains the summarised information gain E_G for every time slot using the aforementioned procedure, it uses a Monte-Carlo-based method to determine which time slots to use for exploration and which ones to use for charging. Thus, the probability that a given time slot will be selected for exploration is proportional to its expected information gain E_G . The generated schedule is then saved and the *Scheduler* is deactivated until the start of the next time slot.

At the beginning of each time slot, the *Scheduler* checks whether the time slot was allocated for exploration and eventually queries the *Planner* for an up-to-date plan for the given time. Then, it forwards the set of locations to observe to the *Executioner*.

G. Plan Execution

The *Executioner* module is responsible for carrying out the plan provided by the *Scheduler*. At first, the *Executioner* uses a 2-opt method [28] to establish the sequence in which the planned locations should be visited. Then, it ensures that the robot leaves its charging station, follows the given path while taking measurements at the given locations and returns back to recharge. If the *Executioner* fails to reach a given location, which is typically caused by the location being blocked, it first waits for the location be cleared. If the location remains unreachable, the *Executioner* simply proceeds with the following location in the plan. After each run, the *Executioner* reports the successes or failures in reaching the planned locations to the *Planner*, which updates the *Reachability map*. This causes the robot to avoid areas that are more likely to be blocked. However, the amount of obtainable information for the neighbouring cells is likely to be high, causing the robot to perform observations in nearby locations in the next exploration run.

H. The Robot

The platform used in this letter is a SCITOS-G5 mobile robot equipped with RGB-D cameras and a laser rangefinder. The robot’s navigation system is based on open-source, freely available software developed during the STRANDS project [29], which extends the navigation stack of the Robot Operating System (ROS). The sensor that was used for 4D mapping presented in this letter was the Asus Xtion RGB-D camera, which was mounted on a pan-tilt unit placed on top of the robot’s head. Using this pan-tilt unit, the robot created 360° × 90° 3D sweeps with a 4 m radius at locations it was supposed to observe.

IV. EXPERIMENTAL EVALUATION

The most popular metrics used to evaluate static exploration methods are the completeness of the robot’s environment model and the time or travel distance required to complete the exploration. Since spatio-temporal exploration is a continuous process, these metrics are not applicable. Another quality metric lies in comparison of the ground truth with the robot’s internal environment model.

Thus, we propose to evaluate the exploration algorithm by two different metrics. The first metric is based on the amount of changes observed by the exploration algorithm. This metric is calculated directly as the number of cells that change their states through direct observation during an exploration tour. One would expect that a better spatio-temporal exploration algorithm would be able to observe more changes, because it can use its predictive capabilities to determine which areas are more likely to change and direct the robot’s attention to these locations.

The second metric is based on the accuracy of the created model, which is calculated by comparing the model with the ground truth. However, obtaining a complete ground truth would require continuous observation of all environment locations, which would necessitate an extensive infrastructure. To overcome this limitation, we built a simulated environment, where ground truth can be obtained relatively easily, and performed the ground-truth comparison in the simulated environment. In the real-world experiment, the ground-truth comparison is performed on a limited set of regions around the researchers’ workplaces.

A. Experiment Description

To evaluate our spatio-temporal exploration algorithm we compare it against a method that considers a static environment model. This ‘Spatial-only’ exploration method is equivalent to state-of-the-art information-theoretic next-best-view exploration methods, such as [16].

To compare these two methods, a robot platform running the system described in Section III was deployed in both a simulated and a real-world office for 5 business days.

Every day at midnight, the *Scheduler* generated a schedule for the following day. This schedule was composed of 15-minute-long time slots, of which 48 were exclusively allocated for the spatial-only (SO) and 48 for the spatio-temporal (ST) exploration algorithm. Since each method had to use half of its allocated time slots to replenish the robot’s batteries, the robot performed 24 exploration tours guided by the spatio-temporal method and 24 tours guided by the spatial-only method per day.

Both methods operated as described in Section III. The only difference between them was that the ST method used the predicted (by the *Spatio-Temporal Model*) map while the SO method used the last obtained map. We hypothesize that the use of a predicted map should allow the *Scheduler* to determine when it is more likely to obtain more information and schedule more exploration tours at the times when the office is more likely to be occupied. Moreover, the *Planner* should be able to predict which areas of the environment are likely to change at a particular time and take this into account when generating the locations to explore.

B. Real-World Experiment

The real-world experiment was performed in an open-plan office of the Lincoln Centre for Autonomous Systems (L-CAS). The office consists of a kitchen area, a lounge area and 20

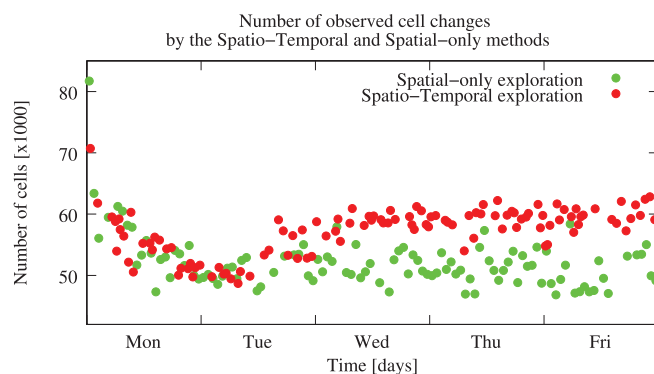


Fig. 3. The number of observed occupancy changes by the Spatio-Temporal versus the Spatial-Only exploration methods.

working places that are occupied by students and postdoctoral researchers. During the experiment, two ceiling-mounted cameras were used to capture a time-lapse video of the office dynamics, which allowed not only for a location-based ground truth comparison, but also to build a database of the office dynamics.

After five days of exploration, we calculated the number of cell changes that were observed by the two aforementioned strategies during the individual exploration tours. Figure 3 shows that at the start of the exploration process, the number of cells that changed their state was high, but gradually decreased as the environment structure became known. After the first day, the amount of changes observed by both methods tended to stabilize around a value given by noise and the environment dynamics. During the second day the Spatio-Temporal method would start to identify the daily routines and the *Planner* would guide the robot to locations that are more likely to exhibit changes – see Figures 1 and 4 for the spatio-temporal map obtained after the first two days of the experiment. After the second day, the Spatio-Temporal method would allocate more exploration tours to the afternoon, when the office is more likely to be populated. In fact, there were 30% more tours scheduled for the afternoon than for the morning.

In the last three days of the experiment, the Spatio-Temporal method observed more changes than the Spatial-Only one, due to its ability to identify the locations and times of environmental change. In other words, the Spatio-Temporal exploration method could plan better both **where** and **when** to explore.

To establish the accuracy of the models created, we selected six working locations in the office, see Figure 4, and manually established the presence of people at these locations over time. Then, we used the Spatio-Temporal models built by the two exploration strategies to predict the overall occupancy of these areas (see Figure 4) for every hour of the five-day experiment. Then, we compared these occupancies to the ground truth for people presence provided by hand. This allowed us to calculate the error of each model in the same way as in [23], i.e. as an average deviation from the ground truth during the experiment. Table I indicates that part of the dynamics of these locations can be explained by periodic processes related to human activity. The researchers working at these six places had diverse working habits, which caused the error rates to vary across the individual

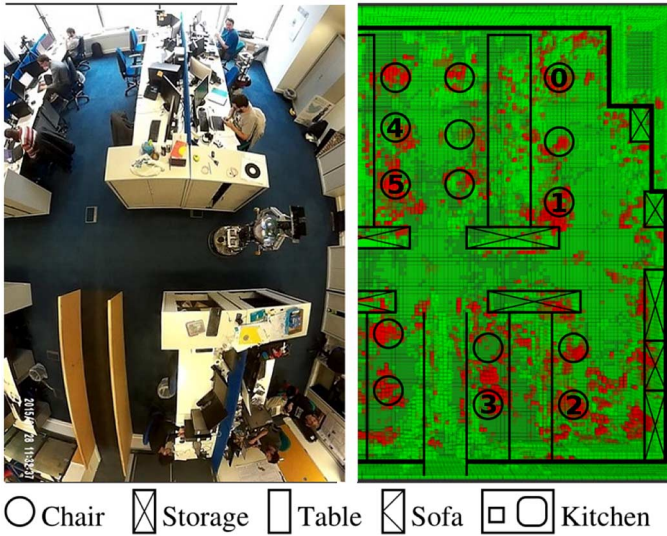


Fig. 4. The layout, the spatio-temporal occupancy grid and top camera view of the Witham Wharf office. The static cells are in green and the cells that exhibit daily periodicity are in red. The locations for ground-truth evaluation are marked with numbers.

TABLE I
OVERALL ERROR OF THE ENVIRONMENT MODEL (%)

Model type	Location						Avg	StD
	0	1	2	3	4	5		
SO	28	23	43	23	21	29	28	8
ST	20	23	25	19	17	14	20	4

locations. Performing a paired t -test indicates that the error of the ‘Spatio-Temporal’ environment model is significantly lower than the error of the ‘Spatial-Only’ method.

C. Simulated Experiment

To speed up testing and to allow for a more representative ground-truth comparison, we created a 3D MORSE-based simulation of our office.

Moreover, we created a software component that allows to reconfigure the simulated environment on the fly. Using the data gathered for five days by two ceiling cameras, we created a database that contains the positions and presence of twenty dynamic objects over time. Combination of the reconfigurable simulator with the aforementioned database allowed us to create a realistic simulation that reflects the real-world dynamics. Thus, our simulator does not only reflect the environment’s static structure, but also simulates dynamic elements, such as people, chairs, laptops and doors (see Figure 5). The experiment was performed in the same way as in the real environment. The number of changed cells captured by both the Spatio-Temporal and Spatial-Only algorithms followed a similar pattern as in the real-world experiment. The main advantage of the simulation was the possibility to obtain ground truth that spans the entire space and time of the experiment. The ground truth for a single time slot was obtained by configuring the simulation for a particular time and letting the robot perform its 3D sweeps at several locations in order to obtain a complete overview of

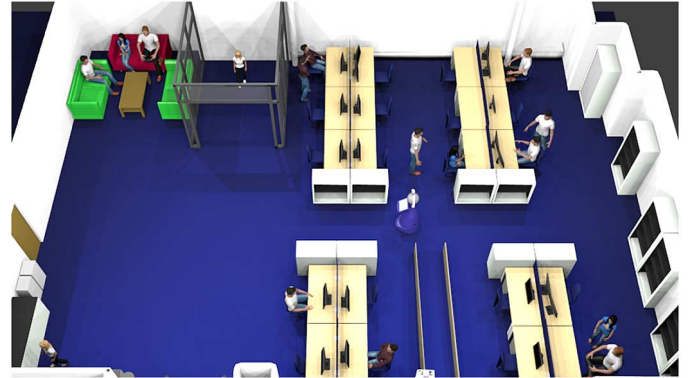


Fig. 5. Snapshot of the simulated environment.

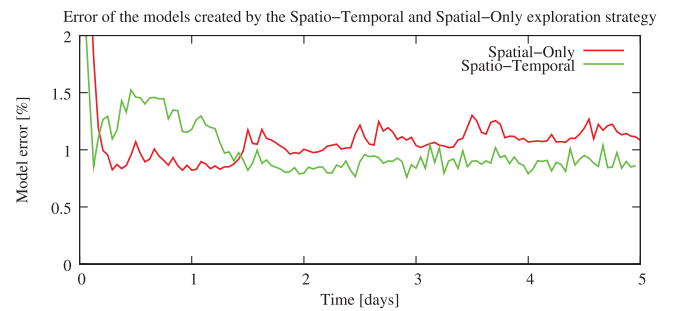


Fig. 6. The ratio of incorrectly estimated cells for the Spatial-Only and Spatio-Temporal strategies.

the environment. This was repeated for every time slot of the experiment, obtaining 480 static 3D grids that represent the environment’s evolution over time. The error of a particular model at a given time was calculated as the number of cells whose states differ from the ground truth divided by the total number of observed cells.

To compare the performance of the SO and ST models, we calculated their errors for each time slot and performed a t -test of the error values for each experiment day. The results of the t -tests indicate that during the first day, the ST model performed significantly worse than the SO one. During the second day, the ST model started learning the periodic patterns, which improved its performance, and the ST and SO model errors were not significantly different. The t -tests of the third, fourth and fifth day show that during these days, the ST model error was significantly lower than the SO one. These results are consistent with Figure 6, which illustrates the error of the ST and SO models over time.

The experimental results indicate that the Spatio-Temporal method can identify periodic patterns in the environment and take them into account when creating the schedule, which results in more changes being observed. The observed changes improve the predictive ability of the Spatio-Temporal model, which allows to construct a better exploration schedule. Note that this is due to the fact that part of the environment dynamics is periodic. If the environment was changing non-periodically, both Spatial-Only and Spatio-Temporal methods would capture a similar amount of changes.

V. CONCLUSION

We presented a 4D exploration method for dynamic environments that extends information-driven exploration into the time domain. The proposed method explicitly models the world dynamics and can predict the environment change. The predictive ability is used to reason about the most informative locations to explore for a given time. Experimental results show that taking into account the environment dynamics increases the amount of information gathered compared to approaches that represent the environment by a static structure. Thus, our method allows for efficient creation and maintenance of spatio-temporal models that increase the robot's efficiency in long-term scenarios.

As future work we intend to embed the spatio-temporal exploration strategy as part of the robot's daily routine and verify whether the ability to improve the spatio-temporal representations during long-term operation results in continuous improvement of the robot performance. Moreover, we want to study the impact of different spatio-temporal models and exploration strategies on the robot's efficiency in performing useful tasks during long-term deployment. In particular, we will examine the possibility of integrating the Markov-Model-based representation proposed in [4]. To allow the robot deployment in spatially-large environments, we also plan to use a FreMEN-based Octomap [30], [31] instead of a uniform 3D occupancy grid. We are also investigating alternative methods for calculating spectral representations from sparse observations.

REFERENCES

- [1] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proc. Robot. Sci. Syst. (RSS)*, 2005, pp. 17–24.
- [2] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1645–1661, 2013.
- [3] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localization for service robots in dynamic environments using spectral maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2014, pp. 4537–4542.
- [4] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1662–1678, 2013.
- [5] P. Neubert, N. Sünderhauf, and P. Protzel, "Superpixel-based appearance change prediction for long-term navigation across seasons," *Robot. Auton. Syst.*, vol. 69, pp. 15–27, 2014.
- [6] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Mag.*, vol. 9, no. 2, pp. 61–74, 1988.
- [7] J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide, "Now or later? Predicting and maximising success of navigation actions from long-term experience," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 1112–1117.
- [8] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, and T. Duckett, "Where's Waldo at time t? Using spatio-temporal models for mobile robot search," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2140–2146.
- [9] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *Int. J. Robot. Res.*, vol. 29, no. 9, pp. 1131–1153, 2010.
- [10] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary maps for lifelong visual localization," *J. Field Robot.*, 2015.
- [11] J. Biswas and M. Veloso, "Episodic non-Markov localization: Reasoning about short-term and long-term features," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 3969–3974.
- [12] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 1871–1878.
- [13] J. Saarinen, H. Andreasson, and A. Lilienthal, "Independent Markov chain occupancy grid maps for representation of dynamic environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 3489–3495.
- [14] S. Koenig, C. Tovey, and W. Halliburton, "Greedy mapping of terrain," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2001, vol. 4, pp. 3594–3599.
- [15] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *Proc. ISR/ROBOTIK*, 2010, pp. 1–8.
- [16] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robot. Auton. Syst.*, vol. 58, no. 5, pp. 684–699, 2010.
- [17] V. Caglioti, "An entropic criterion for minimum uncertainty sensing in recognition and localization—I: Theoretical and conceptual aspects," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, 2001, pp. 187–196.
- [18] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. Robot. Sci. Syst. (RSS)*, Cambridge, MA, USA, 2005.
- [19] J. P. Fentanes, R. F. Alonso, E. Zalama, and J. G. García-Bermejo, "A new method for efficient three-dimensional reconstruction of outdoor environments using mobile robots," *J. Field Robot.*, vol. 28, no. 6, pp. 832–853, 2011.
- [20] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser, "Modeling and decision making in spatio-temporal processes for environmental surveillance," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 5490–5497.
- [21] R. Marchant and F. Ramos, "Bayesian optimisation for intelligent environmental monitoring," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 2242–2249.
- [22] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 6136–6143.
- [23] T. Krajník, J. Santos, and T. Duckett, "Life-long spatio-temporal exploration of dynamic environments," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, 2015, pp. 1–8.
- [24] T. Krajník *et al.*, "A practical multirobot localization system," *J. Intell. Robot. Syst.*, 2014.
- [25] M. Levoy, "Efficient ray tracing of volume data," *ACM Trans. Graph.*, vol. 9, no. 3, pp. 245–261, Jul. 1990, doi: 10.1145/78964.78965.
- [26] T. Krajník, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral analysis for long-term robotic mapping," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 3706–3711.
- [27] M. Rahman, *Applications of Fourier Transforms to Generalized Functions*. WIT Press, 2011.
- [28] G. A. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, 1958.
- [29] N. Hawes *et al.* (2014). *Strands Software System* [Online]. Available: <http://strands-project.eu/software.html>
- [30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [31] T. Krajník, J. Santos, B. Seemann, and T. Duckett, "Froctomap: An efficient spatio-temporal environment representation," in *Proc. Towards Auton. Robot. Syst. (TAROS)*, vol. 8717, pp. 281–282, 2014.

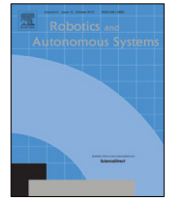
KEY ARTICLE [34] - ROBOTICS AND AUTONOMOUS SYSTEMS
2017

©[2017] Elsevier. This manuscript version is made available under the CC-BY-NC-ND 4.0 license <http://creativecommons.org/licenses/by-nc-nd/4.0/>.



Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Spatio-temporal exploration strategies for long-term autonomy of mobile robots

João Machado Santos*, Tomáš Krajník, Tom Duckett

Lincoln Centre for Autonomous Systems, University of Lincoln, Brayford Pool, Lincoln, Lincolnshire, LN6 7TS, United Kingdom



ARTICLE INFO

Article history:

Available online 17 November 2016

Keywords:

Mobile robotics
Spatio-temporal exploration
Long-term autonomy

ABSTRACT

We present a study of spatio-temporal environment representations and exploration strategies for long-term deployment of mobile robots in real-world, dynamic environments. We propose a new concept for life-long mobile robot spatio-temporal exploration that aims at building, updating and maintaining the environment model during the long-term deployment. The addition of the temporal dimension to the explored space makes the exploration task a never-ending data-gathering process, which we address by application of information-theoretic exploration techniques to world representations that model the uncertainty of environment states as probabilistic functions of time. We evaluate the performance of different exploration strategies and temporal models on real-world data gathered over the course of several months. The combination of dynamic environment representations with information-gain exploration principles allows to create and maintain up-to-date models of continuously changing environments, enabling efficient and self-improving long-term operation of mobile robots.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

As robots gradually leave the well-structured worlds of factory assembly lines and enter natural, human-populated environments, new challenges appear. One of the first problems is to operate in less structured and more uncertain environments. This challenge gave birth to the field of probabilistic mapping, which enables the representation of incomplete world knowledge obtained through noisy sensory measurements [1]. Initially, the environment models had to be created during a human-guided procedure [2], but later, the combination of probabilistic mapping and planning methods allowed robots to create the environment models themselves by means of autonomous exploration [3]. However, as robots became able to operate autonomously for longer periods of time, a new challenge appeared—their typical operating environments are subject to change.

These changes manifest themselves through sensory measurements—every perceived change causes the sensory data to disagree with the original model obtained during the exploration phase. Although probabilistic mapping methods can deal with conflicting measurements, their approach is rooted in the idea that these variations are caused by inherent sensor noise rather than by structural environment change. Thus, these conflicting measurements are generally treated as outliers caused by unwanted noise. Recently, some authors exploited these conflicting

measurements in order to obtain information about the world dynamics and proposed representations that model the environment dynamics explicitly. These dynamic representations have shown their potential by improving mobile robot localization in changing environments [4–7].

Similarly to traditional robotic mapping, introduction of spatio-temporal mapping naturally requires novel exploration strategies that allow to build and maintain spatio-temporal maps during the robot's deployment. Classic exploration strategies aim at building a spatial-only model that covers the robot's entire operational environment, ignoring the fact the environment might change after its completion. Unlike the classic exploration approaches, spatio-temporal exploration is a never-ending task, for several reasons. First, some areas of the operational environment are not exactly predictable during certain times, which requires the robot to re-observe those locations at the times when their state is uncertain. For example, even if we know the general habits of a certain person, her presence at her workplace is uncertain around the start and end of office hours, and thus, it makes sense to observe the workplace during these times. Second, the patterns in the environment dynamics might change and identification of the new patterns requires re-observation of the particular area at the right times. For example, the workplace might be occupied by a new employee with a different working pattern. Additionally, the general structure of the environment can change due to reconstruction or displacement of furniture.

Thus, the robot needs to take repeated observations of locations in its operational environment over time in order to successfully

* Corresponding author.

E-mail addresses: jsantos@lincoln.ac.uk (J.M. Santos), tkrajnik@lincoln.ac.uk (T. Krajník), tduckett@lincoln.ac.uk (T. Duckett).

build and maintain a spatio-temporal model. This requires the robot to continuously explore the environment in addition to the other tasks it was designed for. Therefore, spatio-temporal exploration must become a part of the robot's daily routine that has to be carried out along with other tasks that the robot is required to perform. The ability to build and maintain the aforementioned spatio-temporal representations allows the mobile robot to better cope with changes in the environment and to perform its daily duties efficiently. Hence, being able to build, maintain and reason over such an environment representation plays a key role in achieving long-term operation without any major human intervention, i.e., long-term autonomy.

We present an exploration method that integrates sensory data captured at different times and locations into a dynamic spatio-temporal model and uses the model to determine where and when to perform future observations, while being able to cope with the other tasks the robot needs to perform. We show that the application of information-theoretic planning principles to environment models that represent uncertainties of environment states in the frequency domain results in an intelligent exploratory behaviour, which evolves as the environment knowledge becomes more refined over time. Moreover we evaluate all possible combinations of four different spatio-temporal models and five planning strategies by their long-term performance, according to their ability to provide an accurate environment model over time. To complete this study we also evaluate the impact of different exploration versus exploitation ratios on the overall accuracy of the model. The *exploration versus exploitation dilemma* means that the robot has to find a balance between the time spent exploring and the quality of its internal model [8]. The work presented in this article extends the study presented in [9] by providing a more detailed description of the spatio-temporal models and exploration strategies and by introducing a new recency-based short-term model, as well as a novelty-driven exploration strategy that takes into account the predictions of both the recency- and periodicity-based models.

2. Related work

In order to explore the environment in an efficient way, the robot not only has to be able to create a map from its sensory inputs, but also to use the map to plan its path so that it can reach previously unknown areas of the environment. Therefore, mobile robot exploration is an iterative process in which the robot integrates its observations into its world model, interprets the world model to determine which parts of the environment are unknown, and plans a path to visit and observe these unknown areas. Therefore, an efficient exploration system consists of three essential components: mapping, goal generation and path planning. For the purpose of spatio-temporal exploration, we have to use mapping methods that allow to represent dynamic environments and goal generation methods that can determine not only the position, but also the times of observations—i.e. we have to schedule the observations in such a way that the robot can perform its other tasks as well.

2.1. Exploration methods

One of the earliest and well-known methods is frontier-based exploration [2,10,11]. This approach represents the environment as an occupancy grid, which is processed to obtain boundaries (frontiers) between the known and unknown parts of the environment. The robot movement is then planned so that these frontiers are visited and removed. The advantage of this approach is its scalability—the frontiers can be distributed among a number of robots that can explore the environment in a cooperative manner [12]. Even though these strategies ensure the completeness of

the environment model, i.e. they aim at removing all the frontiers, they do not take into account the model quality.

Another class of exploration methods is based on the notion of entropy. These methods generate a set of candidate observations and estimate the amount of information these are expected to provide [13]. The information gain is calculated as the reduction in entropy of the world model, which requires a probabilistic representation of the environment states. The lower the entropy of the environment model, the more it reflects the actual environment state.

An information-gain-based approach that integrates localization, mapping and exploration is presented in [14]. The method uses a particle filter to build the map of the environment and an entropy estimation method to plan the next location to be visited by the robot. However, the candidate observations are not evaluated simply by their information gain. Rather, the evaluation takes into account other criteria, such as the time to reach the respective location [15]. An advantage of these methods is that they not only attempt to cover the entire environment as quickly as possible, but also plan re-observations of previously visited locations to increase the quality of the resulting map [16].

Some exploration strategies aim at building maps of the environment taking into account some *a priori* knowledge instead of building it from scratch. For instance, Oßwald et al. [17] propose a novel exploration strategy that aims at decreasing the exploration time by assuming that the layout of the environment is known, such as graphs automatically obtained from floor plans. In this method a Travelling Salesman Planner generates a global plan for the exploration run, while a frontier-based strategy is used to explore the environment at each node of the graph. In [18] an exploration strategy capable of predicting how the unexplored areas may look based on previously mapped areas is proposed. This strategy combines the knowledge obtained through previous exploration tasks (in different environments) to predict which observation points might close the loop with information-driven exploration to more map the environment more efficiently.

The aforementioned exploration strategies aim at building a map of the environment in the initial stage of the robot deployment, but fail at maintaining it over time, ignoring the changes in the environment. Thus the model accuracy will decrease as the environment changes, which would eventually lead to major localization and navigation failures.

Other strategies could include intrinsic motivation systems, which drive the robot towards situations that maximize the performance of the learning process [19,20]. These strategies are able to actively identify anomalous or novel situations that might lead to decisions that provide more information and allows to deal with situations where the information-gain never decreases due to physical constraints. For example, novelty detection strategies, which involve the recognition of environmental stimuli that differ from those usually seen, allow the robot to gradually redirect its attention according to the evolution of its internal models [21].

2.2. Dynamic environment representations

Once robots have attained the ability to operate for longer periods of time, the effects of the environment changes have to be taken into account. The first approaches were aimed at short-term dynamics. These methods identify dynamic objects and remove them from the environment representations [22,23] or use them as moving landmarks [24] for self-localization. However, some dynamic objects do not move at the time of mapping and, consequently, the robot needs further observations to identify them. Ambrus et al. [25] propose to process several 3d point clouds of the same environment obtained over a period of several weeks to separate movable objects and refine the model of static environment structure at the same time.

Other approaches do not explicitly segment movable objects, but use representations that are able to model large-scale, substantial environment changes over long time periods. Some authors [26,27] represent the environment dynamics by multiple temporal models with different timescales, and Dayoub and Duckett [28] use a ranking scheme that allows to identify environmental features that are more likely to be stable in long-term. Churchill and Newman [4] propose to cluster similar observations at the same spatial locations to form ‘experiences’ which are then associated with a given place and show that this approach improves autonomous vehicle localization. Tipaldi et al. [6] represent the states of the environment components (cells of an occupancy grid) with a hidden Markov model and show that their representation also improves localization. In [29], each cell in the occupancy grid stores not only the probability of it being occupied, but also the likelihood of the cell to change after a given time. Kucner et al. [30] propose a method that learns conditional probabilities of neighbouring cells of an occupancy grid to model typical motion patterns in dynamic environments. Neubert et al. [7] proposed a method that can learn appearance changes based on a long-term dataset collected across multiple seasons and use the learned model to predict the environment appearance for a given time. Another approach that possesses the ability to predict environment changes is proposed by Rosen et al. [31], which uses Bayesian-based survivability analysis to predict which environment features will still be visible after some time and which features will disappear.

Another family of algorithms aims at creating models of the environment that allow them to predict where and when to make observations of specific phenomena within the environment. Typically, these algorithms rely on Gaussian Processes [32–34], which allow the robot to learn patterns in the environment. Even though these approaches are able to build models of given phenomena, these models are not used by the robot itself to improve essential competences such as localization.

Finally, Krajník et al. [35] propose to represent the environment dynamics in the spectral domain and apply this approach to image features to improve localization [5], to occupancy grids to reduce memory requirements [36], and to topological maps to improve both path planning [37] and robotic search [38]. While being applicable to most environment models used in mobile robotics, the aforementioned method suffers from a major drawback due to its reliance on the traditional Fast Fourier Transform (FFT) method, which requires the environment observations to be taken on a regular and frequent basis. This means that the robot’s activity has to be divided into a learning phase, when it would frequently visit individual locations to build its dynamic environment model, and a deployment phase when it would use its model to perform useful tasks. This division means that while the robot can create dynamic models, which are more suitable for long-term autonomy, it cannot maintain them during subsequent operation. Thus, the robot does not adapt to dynamics that were not present during the learning phase. This fundamental limitation is addressed by the incremental update scheme introduced in this paper.

2.3. Exploration vs exploitation

The long-term deployment of mobile robots in human-populated environments must take into account the need to balance exploitation of what the robot already knows and exploration to select better actions in the future [8]. This issue is addressed in [39], which describes the deployment of a mobile robot in a care centre. Several tasks need to be performed by the robot but there is one that directly addresses the exploration/exploitation dilemma. Here, the mobile robot has to act as an information terminal providing information services to visitors. This task is scheduled at different locations in order to increase the number of

interactions. However, the scheduler must address two different objectives: exploration and exploitation. The first one creates and maintains a spatio-temporal model of the interactions, providing interaction likelihoods for the different locations and times. The second one aims at visiting the different locations at times where the likelihood of observing interactions is uncertain. Based on the above work, Kulich et al. [40] developed several policies to schedule actions that allow to increase exploitation, or more specifically to increase the number of interactions with humans. In order to increase the interactions, the robot needs to learn human behaviours, more specifically where and when it is more likely for a human to ask for assistance. However, this needs to be achieved in parallel with the human interactions as well as the other daily tasks.

3. Spatio-temporal exploration

The primary purpose of robotic exploration is to autonomously acquire a complete and precise model of the robot’s operational environment. To explore efficiently, the robot has to direct its attention to environment areas that are currently unknown. If the world was static, these areas would simply correspond to previously unvisited locations. In the case of dynamic environments, visiting all locations only once is not enough, because they may change over time. Thus, dynamic exploration requires that the environment locations are revisited and their re-observations are used to update a dynamic environment model. However, revisiting the individual locations with the same frequency and on a regular basis is not efficient because the environment dynamics will, in general, not be homogeneous (i.e. certain areas change more often and the changes occur only at certain times). Similarly to the static environment exploration problem, the robot should revisit only the areas whose states are unknown at the time of the planned visits. Thus, the robot has to use its environment model to predict the uncertainty of the individual locations over time and use these predictions to plan observations that improve its knowledge about the world’s dynamics.

To tackle the problem of predicting environment uncertainty over time, we propose to model the probabilities and entropies of the environment states as functions of time. While the main idea is still that some of the environment’s mid- to long-term dynamics are periodic [35], the underlying mathematical representation had to be reformulated. Unlike the method in [35] that requires frequent and regular environment observations, the method proposed in this paper allows to incrementally and continuously update the spatio-temporal model from sparse observations taken at different locations and times. This eliminates the need for a separate training and deployment phase, and allows integration of spatio-temporal exploration into the robot’s daily routine. Thus, the robot can continuously refine its internal environment model and improve its efficiency from the experience gathered over long periods of time.

3.1. Problem definition

Let us represent the environment as a set S of n discrete non-stationary independent binary states $s_i(t)$ that are observable by a mobile robot through its sensors. The states $s_i(t)$ might represent the occupancy of individual cells in an occupancy grid, the traversability of edges in a topological map, the visibility of environmental features, etc. Since these states are dynamic and the robot cannot observe all the states all the time, it maintains an internal environment model that we denote as a set S' , where each element $s'_i(t)$ corresponds to the real-world state $s_i(t)$. To represent the fact that the currently unobserved states are uncertain, we associate each state with a probability value $p_i(t)$ such that $p_i(t) =$

$P(s_i(t) = 1)$. We refer to the probability function $p_i(t)$ and the way it is calculated from the past observations of $s_i(t)$ as a **temporal model**.

Let us define a **location** as a set of environment states that can be observed simultaneously, i.e. a location \mathcal{L}_j is a subset of \mathcal{S} such that by visiting location \mathcal{L}_j at time t , observations of the states that belong to \mathcal{L}_j are obtained. Given that the robot location at time t is $l(t)$, the robot can directly observe only the states s_i of location $\mathcal{L}_{l(t)}$ and states observable at other locations have to be estimated. Thus, the states of the robot's internal environment model are

$$s'_i(t) = \begin{cases} s_i(t) & \text{if } s_i \in \mathcal{L}_{l(t)} \\ p_i(t) \geq 0.5 & \text{otherwise.} \end{cases} \quad (1)$$

The purpose of the exploration process is to obtain and maintain as faithful an environment model as possible, i.e. to minimize the difference between the states of the real environment \mathcal{S} and its model \mathcal{S}' . Technically, this corresponds to minimization of the model error $\epsilon(T)$ calculated as the difference between the real and estimated states over the time period $[0, T)$ as

$$\epsilon(T) = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^n |s'_i(t) - s_i(t)|. \quad (2)$$

Although the reduction of the error $\epsilon(T)$ can be partially achieved by visiting the relevant locations as often as possible, the robot has to perform other tasks and the number of observations is typically limited. Thus, the robot has to carefully plan where and when to perform observations so that it obtains the relevant data to create, maintain and refine its spatio-temporal models of the environment. From a technical point of view, the robot has to use its internal temporal models $p_i(t)$ to determine a sequence of locations $l(t)$. We refer to the way the robot plans the sequence of $l(t)$ from the $p_i(t)$ as its **exploration strategy**.

4. Spatio-temporal models

The underlying spatial environment representations that we will use to test our approach are occupancy grids, topological and feature-based maps. The elementary states of these models represent the occupancy of individual cells, the presence of people at the particular areas and the visibility of image features. Unlike classic environment models that represent the probabilities of the elementary states $s(t)$ by constant values p , we represent the probability of each elementary state as a function of time $p(t)$. In particular, we model each $p(t)$ as a combination of harmonic functions that correspond to hidden periodic processes in the environment.

4.1. Spectral maps

The idea of identifying periodic patterns in the measured states and using them for future predictions was originally presented in [35]. These methods process the sequence of the measured state $s(t)$ by the Fast Fourier Transform (FFT) to obtain the corresponding frequency spectrum $s(\omega)$ and extract its most prominent spectral components $s'(\omega)$. Then, they employ the Inverse Fast Fourier Transform (IFFT) to recover the sequence of state probabilities $p(t)$, which can be used for anomaly detection [35] or state prediction [5]. However, the reliance of these methods on the Fast Fourier Transform (FFT) algorithm makes their real-world application impractical. First, the FFT can transform only the complete sequence of a state $s(t)$ or its full spectral representation $s(\omega)$. Thus, updating the spectral representation with new measurements or prediction of a single probability requires to recalculate the entire sequence of observations, which becomes computationally expensive as the observations accumulate. Most importantly, the FFT algorithm requires that the environment observations are sampled at regular

intervals, which imposes an inefficient exploration scheme and goes against the concept of spatio-temporal exploration that aims at deciding when and where to observe the environment in a non-regular way.

4.1.1. Frequency map enhancement (FreMEen)

Similarly to the aforementioned spectral representation [35], our method still aims to identify the periodic patterns of the environment states and use them for predictions. Unlike the previous representation in [35], the method proposed here allows to update the underlying dynamic models incrementally from sparse, irregular observations. The proposed method represents each state by the number of performed measurements n , its mean probability μ , and two sets \mathcal{A}, \mathcal{B} of complex numbers α_k and β_k that correspond to the set Ω of periodicities ω_k that might be present in the modelled environment. The set Ω was chosen to cover periodicities ranging from 4 weeks to 2 h with distribution similar to the traditional FFT, i.e. $\omega_k = \frac{2\pi k}{4 \times 7 \times 24 \times 3600}$, where $k \in \{1, 2, \dots, 4 \times 7 \times 12\}$. Initially, the mean value μ is set to 0.5 and all α_k, β_k are set to 0, which corresponds to a completely unknown state.

4.1.2. Addition of a new measurement

Each time a state $s(t)$ is observed at time t , we update its representation, i.e. the number of measurements n , the mean μ and values of \mathcal{A}, \mathcal{B} , which are actually a sparse spectral representation of $s(t)$, as follows:

$$\begin{aligned} \mu &\leftarrow \frac{1}{n+1} (n\mu + s(t)), \\ \alpha_k &\leftarrow \frac{1}{n+1} (n\alpha_k + s(t)e^{-jt\omega_k}) \quad \forall \omega_k \in \Omega, \\ \beta_k &\leftarrow \frac{1}{n+1} (n\beta_k + \mu e^{-jt\omega_k}) \quad \forall \omega_k \in \Omega, \\ n &\leftarrow n+1. \end{aligned} \quad (3)$$

The proposed update step is analogous to incremental averaging—the absolute values of $|\alpha_k - \beta_k|$ correspond to the average influence of a periodic process (with a frequency of ω_k) on the values of $s(t)$. Note that the size of the representation of the state (i.e. the number of elements in \mathcal{A}, \mathcal{B}) is independent of the number of observations, which means that the memory requirements of the proposed representation do not grow with time. Note also that if the times of observations t are equally spaced and the frequencies ω_k are selected as described in Section 4.1.1, then (3) corresponds closely to the traditional Discrete Fourier Transform.

4.1.3. Performing predictions

To predict the value of state $s(t)$ for a future time t , we first create a set \mathcal{C} consisting of $\gamma_k = \alpha_k - \beta_k$ and then sort it descending according to the absolute values $|\gamma_k|$. Then, we extract the first m elements γ_l along with their corresponding frequencies ω_l and calculate the state's probability over time as

$$p(t) = \zeta \left(\mu + \sum_{l=1}^m 2|\gamma_l| \cos(\omega_l t + \arg(\gamma_l)) \right), \quad (4)$$

where $\zeta(\cdot)$ ensures that $p(t) \in [0, 1]$. The choice of m determines how many periodic processes are considered for prediction. Setting m too low would mean that we might omit some environment processes that actually influence the state, while setting m too high might include components of \mathcal{C} that are caused by sensor noise. To estimate the optimal value of m , we compare the predictions performed by (4) to the measured values by means of (2), and select the value of m that minimizes the prediction error ϵ . This choice of m is performed automatically during the robot operation—initially, m equals 0 and is increased only after the robot obtains enough data to verify the prediction accuracy of its

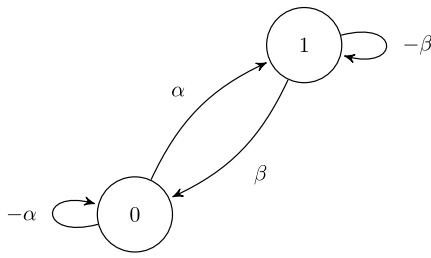


Fig. 1. The underlying Markov chain in the short-term memory model.

spatio-temporal models. Since the most prominent periodicities in human-populated environments are related to the day/night cycle, the value of m typically equals to zero for the first two days of exploration, because inferring a day-long periodicity requires two days of data gathering—one day to build the model and one day to verify it.

One of the main advantages of the proposed representation is that the state is modelled probabilistically. This allows to calculate the time intervals when the particular states are uncertain, which is crucial to direct the robot's attention during exploration.

4.2. Short-term memory

We propose to model the short-term dynamics using a similar model to [29]. This model is based on a Markov chain and aims not only at representing the environment states but also how likely they are to change given the last observed state and the time it was observed. Assuming that each measured state s can be occupied or free, the goal of this method is to estimate the conditional probabilities that represent the transition from one state to another, which are $p(s = 0|s = 1)$ and $p(s = 1|s = 0)$. These probabilities are estimated by means of a Poisson process, i.e., these probabilities can be approximated by the ratio between the number of state changes observed and the total number of observations. However, as described in Section 3, due to the nature of spatio-temporal exploration the observations of states are not performed uniformly in time, and consequently the discrete Markov chain described in [29] as well as the estimation of the aforementioned probabilities do not apply in our case. Thus, we propose a continuous Markov chain to model the recency of the environment states, as shown in Fig. 1. In this case, the transition rates between the states 0 and 1, α and β , are inversely proportional to the average time that an observed state remains at 0 or 1. From the Markov chain shown in Fig. 1, we infer the equations

$$\begin{aligned} \dot{p}_0(t) &= -\alpha p_0(t) + \beta p_1(t), \\ \dot{p}_1(t) &= -\beta p_1(t) + \alpha p_0(t). \end{aligned} \quad (5)$$

Since we only have two states for any time t , we have $p_0(t) + p_1(t) = 1$. Thus, by differentiating and substituting the previous set of equations we obtain Eq. (6), which allows us to predict the probability of the state $s(t)$ for a given future time t , where T is the time of the most recent observation.

$$p(t) = \frac{\alpha}{\alpha + \beta} + \left(p(T) - \frac{\alpha}{\alpha + \beta} \right) e^{-(\alpha + \beta)(t - T)}. \quad (6)$$

4.3. Alternative temporal models

The most popular way to deal with the uncertainty of the environment is based on Bayesian filtering, which updates the state estimates based on the sensor noise characteristics. The typical measurement rate of the robot sensors exceeds the mid-to long-term environment dynamics, therefore the Bayesian update scheme causes the probabilities of the observed states to

quickly converge towards the latest observed values. Typically, the traditional environment representations tend to reflect the latest state measurements, discarding older measurements. However, for long-term deployment it is sensible to use representations that somehow reflect the prior environment states since the initial deployment stage. To strengthen our study, we describe in this section two additional environment representations that take into account all the previous observations, a **long-term memory** model and **Gaussian Mixture Models** (GMM).

4.3.1. Long-term memory

A way to reflect the uncertainty of the observed states in the long-term is to implement a **long-term memory** (LM). The model that we propose works as a memory that takes into account all the observations and calculates the probability of a given state simply as the arithmetic mean of all its past observations.

4.3.2. Gaussian mixture models

Gaussian Mixture Models that can approximate multi-dimensional functions as a weighted sum of Gaussian component densities are a well-established method of function approximation. A Gaussian Mixture Model of a function $f(t)$ is a weighted sum of m Gaussian functions:

$$f(t) = \frac{1}{\sqrt{2\pi}} \sum_{j=1}^m \frac{w_j}{\sigma_j} e^{-\frac{(t - \mu_j)^2}{2\sigma_j^2}}. \quad (7)$$

GMMs find their applications in numerous fields ranging from botany to psychology [41]. The parameters of individual components of GMMs, i.e. the weights w_k , means μ_j and variances σ_j are typically estimated from training data using the iterative Expectation Maximization (EM) or Maximum A-Posteriori (MAP) algorithms. While GMMs can model arbitrarily-shaped functions, their limitation rests in the fact that they cannot naturally represent functions that are periodic.

To deal with this issue, we simply assume that people perform most of their activities on a daily basis and thus we consider the object presence in individual areas as being the same for every day. While this assumption is not entirely correct (as working days will typically be different from weekends), such a temporal model might still be better than a 'static' model where the probability of object presence is a constant.

Prior knowledge of the periodicity allows to transform the measured sequence of states $s(t)$ into a sequence $p'(t)$ by

$$p'(t) = \frac{k}{\tau} \sum_{i=1}^{k/\tau} s(t + i\tau), \quad (8)$$

where τ is the assumed period and k is the $s(t)$ sequence length. After calculating $p'(t)$, we employ the Expectation Maximization algorithm to find the means μ_j , variances σ_j and weights w_j of its Gaussian Mixture approximation. Thus, the probability of occupancy of a room at time t is given by

$$p(t) = \frac{1}{\sqrt{2\pi}} \sum_{j=1}^m \frac{w_j}{\sigma_j} e^{-\frac{(\text{mod}(t, \tau) - \mu_j)^2}{2\sigma_j^2}}, \quad (9)$$

where τ is the a priori known period of the function $p(t)$ and mod is a modulo operator. The periodic-GMM-based (PerGaM) model is complementary to the FFT-based one. It can approximate even short, multiple events, but it can represent only one period (τ) that has to be known a priori. Since the dominating periodicity of human populated environments is 1 day, we chose $\tau = 86\,400$ s.

5. Exploration strategies

As noted in Section 3.1, an exploration strategy is defined as a process that determines both which locations to visit and when

to visit them. One has to assume that a real mobile robot has to perform other tasks as well and can spend only a fraction of the total time on actual exploration. We refer to this fraction as the exploration ratio e , e.g. $e = 0.2$ means that the robot can spend 20% of its operational time on exploration.

Thus, given an exploration ratio e and a set \mathcal{T} of time intervals $[t_s, t_{s+1})$, the exploration algorithm has to determine a sequence $l(t_s)$ of locations to visit. To represent situations where the time slot $[t_s, t_{s+1})$ is allocated to an unrelated activity, the value of $l(t_s)$ is set to zero, whereas a non-zero value of $l(t_s)$ signifies the location to be observed during $[t_s, t_{s+1})$.

5.1. Information-gain strategies

The information-gain strategies take into account the experiences the robot has gathered so far to plan when and which location to visit. These strategies attempt to reduce the uncertainty of the environment models by planning the observations that maximize the potential information gain. To estimate how much information is gained by a particular observation, we will use the notion of entropy. We assume that direct observation of particular states at a given time reduces the entropy of these states to zero. Thus, the information gained by a particular observation can be estimated as the sum of the entropies of the states observed at a given location as

$$I(\mathcal{L}, t) = - \sum_{i \in \mathcal{L}} (p_i(t) \ln(p_i(t)) + (1 - p_i(t)) \ln(1 - p_i(t))). \quad (10)$$

The **Greedy** strategy calculates the potential information gains for all given time slots and locations, then assigns the best location to visit at each time slot. Then, it selects a subset \mathcal{T}' of time slots with the highest information gain such that $e = |\mathcal{T}'|/|\mathcal{T}|$. The remaining time slots are assigned to other tasks. Thus, this strategy maximizes the potential information gain obtained over the time slots in the set \mathcal{T} .

The **Monte Carlo** strategy chooses the locations randomly, but the probability of selecting a given location at a given time is proportional to the estimated information gain. At first, it estimates the $I(l, t_s)$ for all given time slots and locations and sums these values to I' . Then, it calculates the value of $I(0, t_s) = I'(1 - e)/(ne)$. Finally, it calculates the probabilities of each $l(t_s)$ as

$$P(l(t_s) = j) = \frac{I(j, t_s) + \iota}{\sum_{i \in \mathcal{L}} I(i, t_s) + \iota}. \quad (11)$$

Here, the value of $I(0, t_s)$ does not represent actual information gain, but is added to ensure that the exploration ratio e is satisfied by ensuring sufficient chance of assigning the time slots to exploration-unrelated tasks. The positive constant ι ensures that the locations will be occasionally visited even at times when the spatio-temporal model predicts their state with absolute certainty. This allows the robot to detect unexpected changes in the environment dynamics.

The **Novelty-driven** strategy follows the same principle as the Monte Carlo one. However, unlike the Monte-Carlo strategy, which strictly follows a schedule determined by Eq. (11), the novelty-driven strategy uses a combination of temporal models to identify situations where a change in the Monte-Carlo schedule would result in a high amount of information obtained. To identify such situations, the novelty-driven strategy predicts the amount of information obtainable in the following time slot by:

$$I'(\mathcal{L}, t) = - \sum_{i \in \mathcal{L}} (p_{i_{\text{expc}}}(t) \ln(p_{i_{\text{info}}}(t)) + (1 - p_{i_{\text{expc}}}(t)) \ln(1 - p_{i_{\text{info}}}(t))), \quad (12)$$

where $p_{i_{\text{expc}}}(t)$ is calculated by the short-term memory model (see Section 4.2) and serves as a measure of expectation, whereas

$p_{i_{\text{info}}}$ is provided by another model and represents the amount of information expected. If $I'(\mathcal{L}, t) \gg I(\mathcal{L}, t)$, i.e. the amount of information predicted by Eq. (12) is significantly higher than the value calculated by Eq. (10), then the location to visit in the following time-slot is changed accordingly. Thus, if the observed states at a recently visited location did not match their predictions, the robot re-observes the location again to obtain more information about this unexpected event.

5.2. Uninformed strategies

For comparison purposes, we include strategies which select the places to visit regardless of the environment dynamics. These strategies calculate the sequence of visits $l(t_s)$ simply from the values of the ratio e , number of locations n and number of time slots m .

The **Round-Robin** strategy visits all areas of the environment with the same frequency, interleaving the observations with other tasks so that the exploration ratio e is satisfied.

The **Random** strategy also attempts to visit all areas with the same frequency, but the sequence of $l(t_s)$ is not deterministic, but random. The probability of a given slot being assigned to a non-exploration task is equal to $1 - e$ and the probability of visiting the individual locations is uniform and equal to e/n .

6. Evaluation datasets

To evaluate the ability of the various temporal models and exploration strategies, we performed a comparison on two datasets gathered over several weeks. The first, 'Aruba' dataset was gathered by a team of the Center for Advanced Studies in Adaptive Systems (CASAS) to support their research concerning smart environments [42]. The second, 'Brayford' dataset was created at the Lincoln Centre for Autonomous System Research (LCAS) for their research on long-term mobile robot autonomy [5]. The aforementioned datasets were processed so that the dynamics of these environments are represented as visual-feature-based, topological and metric maps.

6.1. The aruba dataset

The 'Aruba' dataset consists of maps capturing 16 week long dynamics of a large apartment that was occupied by a single, house-bound person who occasionally received visitors. An occupancy grid and a topological map were created for every minute of a 16 week long period—the resulting dataset contains over 160 000 metric and topological maps. Since the original dataset [42] is simply a year-long collection of measurements from 50 different sensors spread over an eight-room apartment, these maps had to be created by means of simulation.

First, we processed the events from the original dataset's motion detectors to establish the location of people in the flat for every minute of the 16 weeks. Then, we partitioned the flat into ten different areas, where eight areas represent the rooms and two correspond to corridors. This allowed us to create a topological map that indicates the presence of people in these locations. To obtain the metric representation, we created a simulated environment with the same structure as the 'CASAS' apartment, see Fig. 2. Then the simulation was provided with a sequence of person locations recovered in the previous step. As a result, the simulated environment contains physical models of people at locations provided by the real-world dataset, and thus it reflects the dynamics of the real apartment. A virtual, RGB-D camera equipped robot was also introduced into the virtual environment. Every time the configuration of the simulated environment (i.e. locations of the people) changed, the robot used its 3D sensors to create occupancy grids of the flat's individual rooms. Thus, we obtained occupancy grids that reflect the real environment dynamics minute-by-minute for 16 weeks.



Fig. 2. Aruba environment simulation.

6.2. The Brayford dataset

The Brayford dataset was originally collected for the purpose of benchmarking long-term mobile robot localization algorithms in dynamic environments [5]. The data collection was performed by a human-sized robot equipped with an RGB-D camera in a large, open-space office of the Lincoln Centre for Autonomous Systems. The robot was set up to obtain RGB-D images of eight designated areas every 10 min for a period of one week. Representative examples of the captured images are shown in Fig. 3. While the high-level environment model of this dataset contains information about people presence at the individual locations, the states of the low-level model represent the visibilities of image features [43] established by the method presented in our earlier work on visual localization in changing environments [5]. The resulting dataset contains more than 8000 feature-based and 8000 semantic maps collected over a period of one week.

6.3. Dataset summary

Both datasets contain a high-level model representing people presence at different locations and a low-level model based on RGB-D sensing. In the following sections, we will refer to these models as ‘symbolic’ and ‘metric’, respectively. The aforementioned datasets are available as a part of the long-term dataset collection [44].

7. Experimental results

We assume that the aforementioned datasets reflect the real state of the environments they have been captured in and thus we use the sequence of the observations in the datasets as ground truth. To evaluate how the various temporal models and exploration strategies affect the robot’s ability to create and update its internal environment models, we emulate the exploration process using the datasets gathered. We assume that exploration can be performed during only half of the robot’s operational time (i.e. $e = 0.5$) and that a single observation takes 10 min. While 10 min might seem like a long time, creation of a 3D occupancy grid of a given location means that the robot has to position itself precisely, and capture and process approximately 50 RGB-D images from different viewpoints. This time also includes navigation to the given spot, leaving the charging station, etc.

This exploration procedure corresponds to the situation when the robot updates its spatio-temporal model and generates a new observation schedule every 24 h at midnight. The robot starts with an empty environment model that has all probabilities constant and equal to 0.5.

First, the entropy functions of the individual locations are calculated and 72 observations for the following day are scheduled. Then, these 72 observations are retrieved from the given dataset and the temporal models of the environment states are updated. The updated temporal models are used to recalculate the spatio-temporal entropy and the next day’s observation schedule is then generated. These steps are repeated for every day of the given dataset.

7.1. Evaluating environment model error

To compare the performance of the temporal models and exploration strategies described in Sections 4 and 5, the resulting world model is compared to the actual dataset using Eq. (2), which estimates the error in the environment model. Since there are 4 temporal models and 5 exploration strategies, each comparison considers 20 values that characterize the ratio of incorrectly estimated states to the total number of environment states. One dataset evaluation consist of two comparisons, each corresponding to the given environment representation.

The results of the ‘Aruba’ dataset summarized in Table 1 show that the combination of FreMEn with the novelty-driven or Monte-Carlo strategies reduces the model error by more than 40%. Nevertheless, the combination of FreMEn and the novelty-driven strategy performs slightly better than the combination of the same model with the Monte-Carlo one. One may think that the greedy strategy would be the best performer since it always chooses the room with higher entropy, but in most situations this strategy fails to maintain an up-to-date model. For example, in the case of noisy and unpredictable signals in a given room, the robot will attempt to focus its attention mainly in that room. While this is a logical behaviour—not being able to model the location, the robot will gather the data about it through direct observation, it might not be really desirable, because the robot might not be getting valuable data at all. Also, this behaviour would mean that the robot would not observe the remaining rooms, since the entropy of the current room is higher due to the higher uncertainties.

Table 1

Aruba dataset results: Model errors for different exploration strategies and spatio-temporal models [%].

Strategy	Spatio-temporal model							
	Symbolic				Metric			
	SM	LM	FT	GM	SM	LM	FT	GM
Round Robin	09.3	09.7	06.5	07.5	08.9	09.3	05.6	05.8
Random	08.9	09.5	09.2	07.5	08.7	09.0	08.3	07.2
Greedy	08.5	08.7	07.0	09.4	07.7	10.9	06.2	07.1
Monte-Carlo	08.5	08.9	05.8	06.4	08.0	08.3	05.0	05.7
Novelty-driven	08.5	08.9	05.7	06.1	08.0	08.4	04.9	05.4

Table 2

Brayford dataset results: Model errors for different exploration strategies and spatio-temporal models [%].

Strategy	Spatio-temporal model							
	People presence				Visual features			
	SM	LM	FT	GM	SM	LM	FT	GM
Round Robin	23.7	23.7	16.3	20.2	25.7	27.0	12.7	17.9
Random	23.7	23.8	23.0	23.8	25.9	27.0	25.2	20.3
Greedy	20.2	22.3	19.2	20.1	29.9	29.3	24.4	18.6
Monte-Carlo	23.5	23.5	16.4	19.3	25.6	27.0	12.3	16.9
Novelty-driven	23.4	23.5	15.2	19.4	25.6	27.0	12.1	17.1

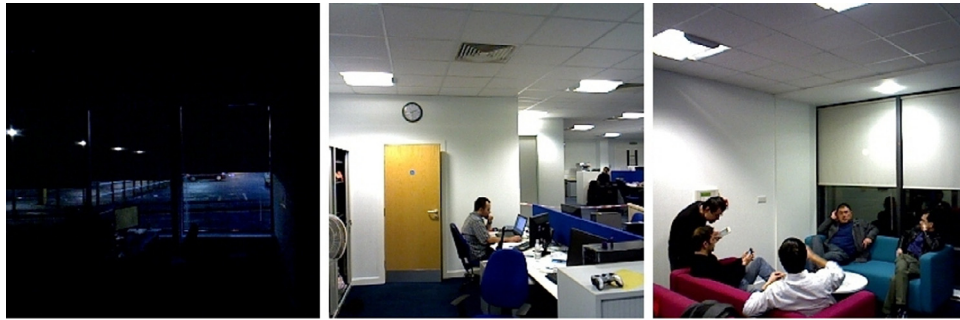
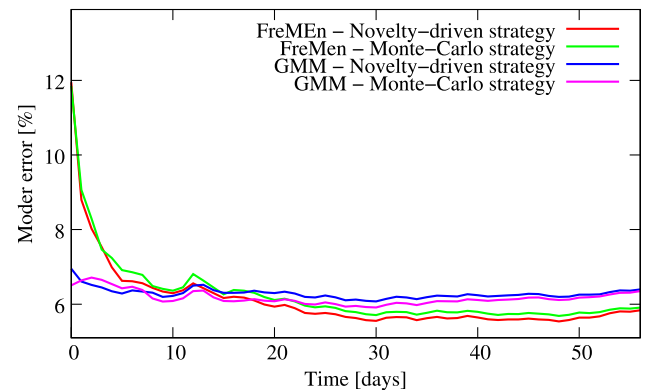
**Fig. 3.** Examples of Brayford dataset images.

Fig. 4 shows that the FreMEn model error is lower during the first day, showing that this strategy allows quicker identification of the environment patterns. Since more than 99% of the cells in the ‘Aruba’ occupancy grids represent empty space or static objects, the model error (Eq. (2)) is calculated for the cells that change their occupancy at least once.

The model errors of the ‘Brayford’ dataset as shown in Table 2 again indicate that the most faithful environment representation is based on frequency-enhanced temporal models (see Section 4.1.1) in combination with the novelty-driven strategy. The improvement is more prominent in the case of people presence models. The reason for this might be that the visibility of image features tends to follow regular patterns given by the daily illumination cycle, whereas the presence of people can be influenced by unexpected events. Note that the model errors of the feature-based maps are higher than the ones reported in [9] because we used a higher number of visual features in our model.

Fig. 4 shows that initially, the GMM model achieves the lowest error, but in the long-term, it is outperformed by FreMEn. This is caused by the fact that the GMM model is tailored to represent daily periodicities, while the FreMEn model has to identify the patterns of changes from the data by itself. After several days, FreMEn identifies several important periodicities (not only the daily one) and its prediction capability improves, allowing it to better schedule observations and decrease the model error. Fig. 4 also shows that the novelty-driven strategy performs slightly, but consistently better than the Monte-Carlo one. In the experiments performed, we observe that the novelty-driven strategy identifies one or two unexpected observations per day.

**Fig. 4.** Comparison of the average error of the novelty-driven and Monte Carlo exploration strategies. The remaining models are not displayed due to their similar performance.

7.2. Exploration vs. exploitation

In the above experiments, the robot’s exploration ratio e was set to 0.5. Thus, the robot could spend 50% of its time gathering data about its operational environment. However, such a ratio is unrealistic—the robot has to spend some time replenishing its batteries and we have to assume that it should perform other tasks as well depending on the application. Moreover, we have to assume that the purpose of the robot is not in creating precise environment models, but to perform useful tasks. Thus, exploration is

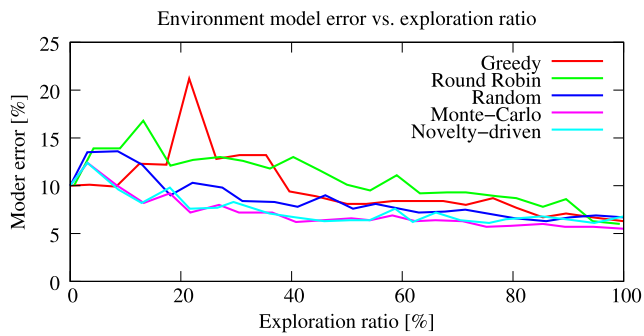


Fig. 5. Exploration vs. exploitation analysis: The influence of the fraction of time spend with exploration on the performance of the exploration strategies.

just an instrument to obtain and maintain knowledge to improve the robot's performance. If the robot spends too much time on exploration, it would not be able to exploit the obtained knowledge in its everyday activities.

We evaluate the efficiency of the individual exploration strategies with different exploration ratios for predicting person presence on the Aruba dataset. We combine the Frequency Map Enhancement models with four different exploration strategies, fix the exploration ratio to a value between 0 and 1, and let the robot explore the Aruba environment for two consecutive weeks. The resulting error of the model obtained is shown in Fig. 5. The results indicate that if the fraction of the time that the robot can spend on actual exploration is low, the dynamic models might make wrong assumptions about the environment changes and perform worse than their static counterparts—this is especially notable with the Greedy and Round Robin strategies. However, this effect can be mitigated by a proper exploration strategy—the graph shows that both Monte Carlo and novelty-based strategies improve the model even if the robot cannot spend too much time on exploration.

Note that the initial model error is 10%—this is caused by the fact that the Aruba dataset represents the presence of people in 10 different areas and the flat has only one inhabitant. Without any observations, the robot simply assumes that the flat is empty, which results in 10% error.

7.3. Qualitative evaluation

To gain an insight into the robot's exploratory behaviour, we interpret the data gathered during the exploration of the 'Aruba' topological map. Here, the robot's task was to create a spatio-temporal model of person presence in the individual rooms of a small apartment. For the purpose of this explanation, let us focus on the dynamics of three rooms only—the bedroom, the kitchen and a storage room. Let the robot use the best-performing exploration method that combines the FreMEn temporal models and the Monte Carlo exploration strategy. Applying the proposed spatio-temporal exploration method to this dataset produced the behaviour in Fig. 6. The top part of Fig. 6 shows the real state of the environment, where the three binary functions $s_i(t)$ represent the room's occupancies over time. The second part shows the robot's internal model of the environment, i.e. the probabilities $p_i(t)$. The third graph displays the information that is expected to be obtained by visiting these three locations at a given time. Finally, the bottom graph shows which locations have been visited at a particular time—we assume that the exploration ratio $e = 0.5$, which reflects the situation where the robot has to spend half of its time on its charging station. Now let us explain how the robot's understanding of the environment changes over time and how this affects its exploratory behaviour day by day.

7.3.1. Day one

Initially, the robot has no knowledge of the environment and therefore the probabilities $p_i(t)$ of the world states $s(t)$ are equal to 0.5. This means that the expected information gain from visiting any of the rooms equals 1 bit at any time of the first day. Thus, the robot has no room or time preference when scheduling the first day's observations.

7.3.2. Day two

After performing the first day's observations, the environment models provide enough evidence that the three rooms are not occupied with the same probability. This is reflected in the second day's environment model—see the probability functions $p_i(t)$ of the second day in Fig. 6. Thus the robot expects to gain more information by visiting the bedroom and kitchen than by going to the storage room. This is reflected in the second day's observation schedule—the last row of Fig. 6 shows that the first two rooms are visited more often.

7.3.3. Day three

The additional observations obtained during the second day provide information about the rooms' dynamics: the robot assumes that the bedroom has a daily periodicity and that the kitchen is visited five times per day. This causes the expected information gain to be time-dependent—the third day of the third row of Fig. 6 shows that evening and morning observations of the bedroom provide more information than in the afternoon. This fact is rather intuitive: visiting the room at the time of its state transition allows to refine the room's state periodicity. Thus, on the third day, the bedroom is visited mostly in the evening and morning, while the afternoon visits are scheduled to the kitchen.

7.3.4. Days four and five

Based on the data gathered during the third day, the robot modifies its hypothesis about the periodicity of activities in the kitchen and assumes that it is visited three times per day. During the following days, the robot tends to visit the kitchen and bedroom more often, and checks the storage room only occasionally. While the kitchen is visited mostly in the early afternoon, the bedroom is visited late evenings and mornings, which allows to refine the robot's model of the person's daily habits.

This example indicates that the combination of a probabilistic temporal model with an information-based strategy not only allows the robot to obtain knowledge about the environment dynamics, but the observations are scheduled in a seemingly logical way: at first, all the locations are visited often and with the same frequency. As the spatio-temporal environment model becomes more refined, the robot tends to visit particular locations only at times when their states are uncertain.

8. Conclusion

In this paper, we presented a method for life-long spatio-temporal exploration of dynamic environments. We assume that the robot's operational environment is subject to perpetual change, which requires a method that can model and predict these variations. The purpose of spatio-temporal exploration is not only to obtain the environment structure and keep it up-to-date with any changes, but also to allow the robot to observe and understand the world dynamics.

We hypothesize that the problem of spatio-temporal exploration can be tackled by combining information-gain-based exploration strategies with probabilistic dynamic environment models. To verify our approach, we compare the performance of five exploration strategies and four temporal models on real-world data gathered over the course of several months. We show that the

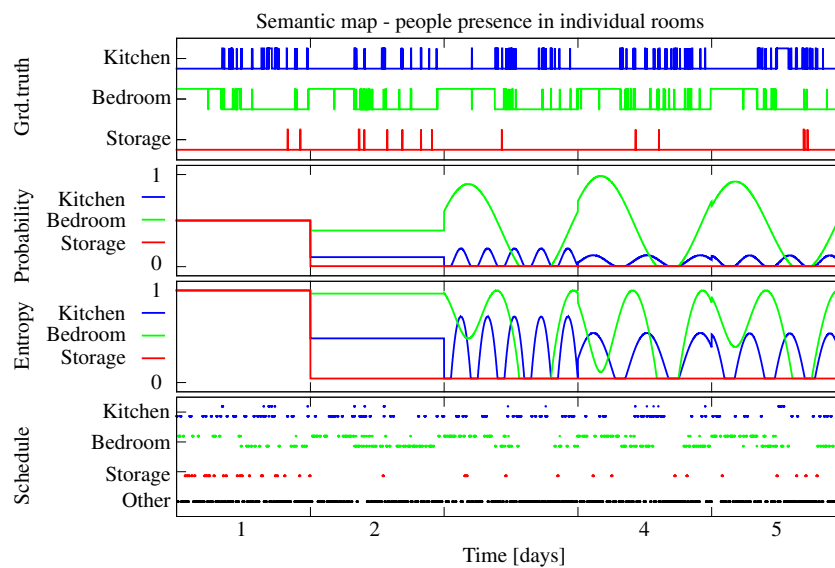


Fig. 6. Spatio-temporal exploration behaviour: The robot uses its probabilistic world model (second row) and spatio-temporal entropy estimates (third row) to schedule its observations (bottom graph) and learn the environment dynamics (top). As the environment knowledge improves over time, the scheduled observations provide more information which allows for further refinement of the environment model.

combination of spectral-based temporal models with information-gain-based novelty-driven strategies results in an intelligent exploration behaviour that improves as the environment knowledge becomes more refined.

Analysis of the robot behaviour shows that when introduced to a new environment, the robot prefers to explore unknown locations. After it has obtained the spatial models, it starts to revisit these locations in order to learn about their dynamics. Finally, the learned dynamics allow the robot to schedule which locations to visit at which times and adapt this schedule in the case of unexpected observations.

The evaluations performed in this paper involved several assumptions to simplify the problem. The first assumption was that the time the robot spends moving to a particular location is negligible compared to the time it takes to make an observation. The second assumption was that the locations of observations were predefined and that the robot could position itself with perfect accuracy. The third assumption is that the observations are error-free, i.e. there is no noise on the sensory data. While these assumptions were needed for validation purposes in this work due to the known difficulties of ground-truthing when comparing exploration strategies, more recent work has overcome these limitations and achieved full 4D metric-based spatio-temporal exploration [45].

The analysis presented here opens several questions for further investigation, which we would like to address in the future. In particular, we will investigate not only the impact of exploration on the quality of the spatio-temporal models, but its impact on the efficiency of the robot operation over time. We will investigate how much time the robot should spend on exploration (represented by ‘exploration ratio’ e) during the initial stages of deployment, when the environment model is created, and what is the optimal e later on, when the model is just maintained or when the model needs to be re-built due to changes in the environment dynamics. We will also investigate which situations in our datasets influenced the novelty-driven strategy, so that it performed better than the Monte-Carlo strategy.

Acknowledgements

The work has been supported by the EU ICT project 600623 ‘STRANDS’.

References

- [1] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press, 2005.
- [2] B. Yamauchi, A frontier-based approach for autonomous exploration, in: *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, 1997.
- [3] B. Kuipers, Y.-T. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, *Robot. Auton. Syst.* 8 (1) (1991) 47–63.
- [4] W.S. Churchill, P. Newman, Experience-based navigation for long-term localisation, *Int. J. Robot. Res.* (2013). <http://dx.doi.org/10.1177/0278364913499193>.
- [5] T. Krajník, et al., Long-term topological localization for service robots in dynamic environments using spectral maps, in: *Proc. of Int. Conference on Intelligent Robots and Systems, IROS*, 2014.
- [6] G.D. Tipaldi, D. Meyer-Delius, W. Burgard, Lifelong localization in changing environments, *Int. J. Robot. Res.* (2013). <http://dx.doi.org/10.1177/0278364913502830>.
- [7] P. Neubert, N. Sünderhauf, P. Protzel, Superpixel-based appearance change prediction for long-term navigation across seasons, *Robot. Auton. Syst.* (0) (2014). <http://dx.doi.org/10.1016/j.robot.2014.08.005>.
- [8] R.S. Sutton, A.G. Barto, *Introduction To Reinforcement Learning*, MIT Press, 1998.
- [9] T. Krajník, J. Santos, T. Duckett, Life-long spatio-temporal exploration of dynamic environments, in: *Mobile Robots (ECMR)*, 2015 European Conference on, 2015, pp. 1–8. <http://dx.doi.org/10.1109/ECMR.2015.7324052>.
- [10] S. Koenig, C. Tovey, W. Halliburton, Greedy mapping of terrain, in: *Proc. of Int. Conference on Robotics and Automation, ICRA*, 2001.
- [11] D. Holz, N. Basilico, F. Amigoni, S. Behnke, Evaluating the efficiency of frontier-based exploration strategies, *ISR/ROBOTIK*, 2010.
- [12] B. Yamauchi, Frontier-based exploration using multiple robots, in: *Proc. of the 2nd Int. Conf. on Autonomous Agents*, 1998.
- [13] V. Caglioti, An entropic criterion for minimum uncertainty sensing in recognition and localization. i. Theoretical and conceptual aspects, *IEEE Trans. Syst. Man Cybern. B* 31 (2) (2001) 187–196. <http://dx.doi.org/10.1109/3477.915342>.
- [14] C. Stachniss, G. Grisetti, W. Burgard, Information gain-based exploration using Rao-Blackwellized particle filters, in: *Proc. of Robotics: Science and Systems, RSS*, Cambridge, MA, USA, 2005.
- [15] C. Stachniss, W. Burgard, Exploring unknown environments with mobile robots using coverage maps, in: *Proceedings of the International Conference on Artificial Intelligence, IJCAI*, 2003.
- [16] J. Fentanes, R.F. Alonso, E. Zalama, J.G. García-Bermejo, A new method for efficient three-dimensional reconstruction of outdoor environments using mobile robots, *J. Field Robot.* (2011).
- [17] S. Oßwald, M. Bennewitz, W. Burgard, C. Stachniss, Speeding-Up robot exploration by exploiting background information, *IEEE Robot. Autom. Lett.* 1 (2) (2016) 716–723. <http://dx.doi.org/10.1109/LRA.2016.2520560>. ISSN: 2377-3766.
- [18] D. Perea Strom, F. Nenci, C. Stachniss, Predictive exploration considering previously mapped environments, in: *Robotics and Automation (ICRA)*, 2015.

- IEEE International Conference on, 2015, pp. 2761–2766. <http://dx.doi.org/10.1109/ICRA.2015.7139574>.
- [19] P.Y. Oudeyer, F. Kaplan, V.V. Hafner, Intrinsic motivation systems for autonomous mental development, *IEEE Trans. Evol. Comput.* 11 (2) (2007) 265–286. <http://dx.doi.org/10.1109/TEVC.2006.890271>.
- [20] S. Thrun, Exploration in active learning, *Handb. Brain Sci. Neural Netw.* (1995) 381–384.
- [21] S. Marsland, Novelty detection in learning systems, *Neural Comput. Surv.* 3 (2003).
- [22] D. Hähnel, D. Schulz, W. Burgard, Mobile robot mapping in populated environments, *Adv. Robot.* (2003).
- [23] D. Wolf, G. Sukhatme, Mobile robot simultaneous localization and mapping in dynamic environments, *Auton. Robots* (2005).
- [24] C.C. Wang, et al., Simultaneous localization, mapping and moving object tracking, *Int. J. Robot. Res.* (2007).
- [25] R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, Meta-rooms: Building and maintaining long term spatial models in a dynamic world, in: *Proceedings of the International Conference on Intelligent Robots and Systems, IROS, 2014*.
- [26] P. Biber, T. Duckett, Dynamic maps for long-term operation of mobile service robots, in: *Proc. of Rob.: Science and Systems, 2005*.
- [27] D. Arbutkule, A. Howard, M. Mataric, Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment, in: *Proc. of Int. Conference on Intelligent Robots and Systems, IROS, vol. 1, 2002*, pp. 409–414. <http://dx.doi.org/10.1109/IRDS.2002.1041424>.
- [28] F. Dayoub, T. Duckett, An adaptive appearance-based map for long-term topological localization of mobile robots, in: *Proc. of Int. Conference on Intelligent Robots and Systems, IROS, 2008*.
- [29] J. Saarinen, H. Andreasson, A. Lilienthal, Independent Markov chain occupancy grid maps for representation of dynamic environment, in: *Intelligent Robots and Systems, IROS, 2012 IEEE/RSJ International Conference on, 2012*, pp. 3489–3495. <http://dx.doi.org/10.1109/IROS.2012.6385629>.
- [30] T. Kucner, et al., Conditional transition maps: Learning motion patterns in dynamic environments, in: *Proc. of Int. Conf. on Intelligent Robots and Systems, IROS, 2013*.
- [31] D.M. Rosen, J. Mason, J.J. Leonard, Towards lifelong feature-based mapping in semi-static environments, in: *International Conference on Robotics and Automation, ICRA, IEEE, 2016*, pp. 1063–1070. <http://dx.doi.org/10.1109/ICRA.2016.7487237>. in press.
- [32] A. Singh, F. Ramos, H.D. Whyte, W.J. Kaiser, Modeling and decision making in spatio-temporal processes for environmental surveillance, in: *Proc. IEEE Int. Conf. Robot. Autom.* 2010, pp. 5490–5497.
- [33] R. Marchant, F. Ramos, Bayesian optimisation for intelligent environmental monitoring, in: *Intelligent Robots and Systems, IROS, 2012 IEEE/RSJ International Conference on, 2012*, pp. 2242–2249. <http://dx.doi.org/10.1109/IROS.2012.6385653>.
- [34] R. Marchant, F. Ramos, Bayesian optimisation for informative continuous path planning, in: *Robotics and Automation (ICRA), 2014 IEEE International Conference on, 2014*, pp. 6136–6143. <http://dx.doi.org/10.1109/ICRA.2014.6907763>.
- [35] T. Krajník, J.P. Fentanes, G. Cielniak, C. Dondrup, T. Duckett, Spectral analysis for long-term robotic mapping, in: *Proc. of Int. Conference on Robotics and Automation, ICRA, 2014*.
- [36] T. Krajník, J. Santos, B. Seemann, T. Duckett, FROctomap: An efficient spatio-temporal environment representation, in: *Advances in Autonomous Robotics Systems*, Springer, 2014, pp. 281–282. <http://dx.doi.org/10.1007/978-3-319-10401-0>.
- [37] J. Pulido Fentanes, et al., Now or later? Predicting and maximising success of navigation actions from long-term experience, in: *International Conference on Robotics and Automation, ICRA, 2015*.
- [38] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, T. Duckett, Where's Waldo at time t? Using spatio-temporal models for mobile robot search, in: *Int. Conf. on Robotics and Automation, ICRA, 2015*.
- [39] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J.L. Wyatt, D. Hebesberger, T. Körtner, R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, L. Beyer, A. Hermans, B. Leibe, A. Aldoma, T. Faulhammer, M. Zillich, M. Vincze, M. Al-Omari, E. Chinellato, P. Duckworth, Y. Gatsoulis, D.C. Hogg, A.G. Cohn, C. Dondrup, J.P. Fentanes, T. Krajník, J.M. Santos, T. Duckett, M. Hanheide, The STRANDS project: Long-term autonomy in everyday environments, *Robotics and Automation Magazine* (2017).
- [40] M. Kulich, T. Krajník, L. Přeucil, T. Duckett, To explore or to exploit? learning humans' behaviour to maximize interactions with them, in: *Modelling and Simulation for Autonomous Systems Workshop, MESAS, 2016*.
- [41] D.M. Tittertoning, A.F. Smith, U.E. Makov, et al., *Statistical Analysis of Finite Mixture Distributions*, vol. 7, Wiley, New York, 1985.
- [42] D.J. Cook, Learning setting-generalized activity models for smart spaces, *IEEE Intell. Syst.* (99) (2010) 1.
- [43] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: Binary robust independent elementary features, in: *Proc. of European Conference on Computer Vision, ECCV, Springer, 2010*, pp. 778–792.
- [44] T. Krajník, J. P. Fentanes, C. Dondrup, M. Hanheide, T. Duckett, L-CAS datasets for long-term autonomy of mobile robots, <http://lcas.lincoln.ac.uk/owncloud/shared/datasets/>.
- [45] J.M. Santos, T. Krajník, J.P. Fentanes, T. Duckett, Lifelong information-driven exploration to complete and refine 4D spatio-temporal maps, *IEEE Robot. Autom. Lett.* 1 (2) (2016) 684–691. <http://dx.doi.org/10.1109/LRA.2016.2516594>. ISSN: 2377-3766.



João Machado Santos is Doctoral candidate at the Lincoln Centre for Autonomous Systems (L-CAS) in the School of Computer Science at the University of Lincoln, UK. Currently, he is involved with the project STRANDS within the FP7 framework, which aims to enable a robot to achieve robust and intelligent behaviour over long periods of time. He holds an M.Sc. degree in Electrical Engineering and Computers, specialization in Automation, from the Faculty of Sciences and Technology of the University of Coimbra obtained in September, 2013. His research interests include mobile robotics, mapping, localization and explo-

ration.



Tomáš Krajník is a research fellow at the Lincoln Center of Autonomous Systems, UK. He received the Ph.D. degree in Artificial Intelligence and Biocybernetics from the Czech Technical University, Prague, Czech Republic, in 2012. His research interests include life-long autonomous navigation, spatio-temporal modelling, and aerial robots.



Tom Duckett is a Professor of Computer Science at the University of Lincoln, UK, where he also leads the Lincoln Centre for Autonomous Systems. His research interests include autonomous robots, artificial intelligence and machine perception, with applications including service robotics and assistive technologies. Tom has co-authored over 100 scientific publications and held peer-reviewed grants worth over 1.5 million at the University of Lincoln.

KEY ARTICLE [34] - ROBOTICS AND AUTOMATION LETTERS
2018 (IN REVIEW)

©[2018] IEEE. Reprinted, with permission, from Tomáš Krajník, Warped Hypertime Representations for Long-term Autonomy of Mobile Robots, 2018.

Warped Hypertime Representations for Long-term Autonomy of Mobile Robots

Tomáš Krajník, Tomáš Vintr, Sergi Molina, Jaime P. Fentanes, Gregorz Cielniak, Tom Duckett

Abstract—This paper presents a novel method for introducing time into discrete and continuous spatial representations used in mobile robotics, by modelling long-term, pseudo-periodic variations caused by human activities. Unlike previous approaches, the proposed method does not treat time and space separately, and its continuous nature respects both the temporal and spatial continuity of the modeled phenomena. The method extends the given spatial model with a set of wrapped dimensions that represent the periodicities of observed changes. By performing clustering over this extended representation, we obtain a model that allows us to predict future states of both discrete and continuous spatial representations. We apply the proposed algorithm to several long-term datasets and show that the method enables a robot to predict future states of representations with different dimensions. The experiments further show that the method achieves more accurate predictions than the previous state of the art.

I. INTRODUCTION

Advances in autonomous robotics are gradually enabling deployment of robots in human-populated environments [1]. Human activity tends to cause changes to the environments it takes place in, and the mobile robots that share these environments need to be able to cope with such never-ending changes. However, merely coping is not enough – ideally, a mobile robot should not only be able to comprehend the environment structure, but also understand how it changes over time. Many authors have shown that even simple models of the environment that adapt to changes improve the overall ability of mobile robots to operate over longer time periods [2], [3], [4], [5], [6]. Moreover, the ability of long-term autonomous operation improves the chances of observing the temporal changes, and thus to extract more valuable information about the environment dynamics, resulting in more accurate spatio-temporal models [1].

Our approach proposes to extend spatial representations by adding several dimensions representing time, with each pair of temporal dimensions representing a given periodicity observed in the gathered data. In particular, we employ the Frequency Map Enhancement [5] concept to identify (temporally) periodic patterns in the data gathered. This approach represents periodic processes in the environment using Fourier analysis, and the resulting spectral models obtained from long-term experience can be used to predict

T. Krajník and T.Vintr are with Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University, CZ tomas.krajnik@fel.cvut.cz

S.Molina, J.P.Fentanes, G.Cielniak and T.Duckett are with Lincoln Centre for Autonomous Systems, University of Lincoln, UK

The work has been supported by the Czech Science Foundation project 17-27006Y STRoLL and EU H2020 agreement 732737 ILIAD.

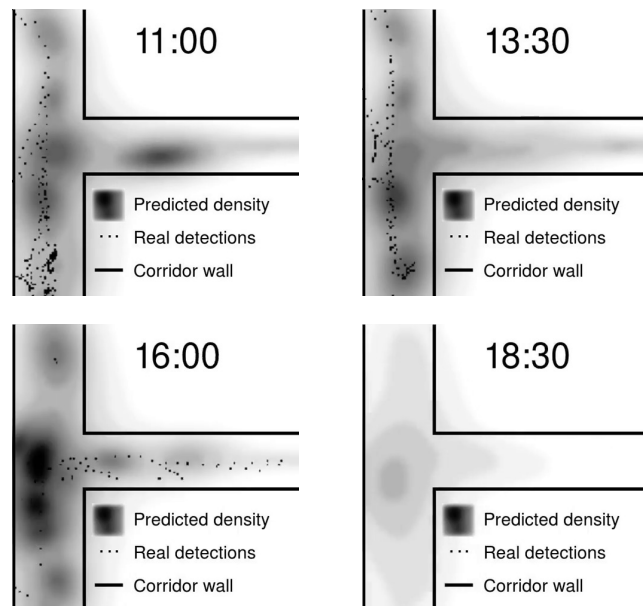


Fig. 1. Spatio-temporal model of people presence in a corridor of the University of Lincoln, UK at various times of a day. See a video at [7].

future environment states. Then, we transform each time periodicity into a pair of dimensions that form a circle in $2d$ space and add these dimensions to the vectors that represent the spatial aspects of the modelled phenomena. The model itself is then built using traditional techniques like clustering or expectation-maximisation over the warped space-hypertime representation. The resulting multi-modal model represents both the structure of the space and temporal patterns of the changes or events. We hypothesize that since the proposed model respects the spatio-temporal continuity of the modelled phenomena, it provides more accurate predictions than models that partition the modeled space into discrete elements. In this paper, we provide a description of the proposed method, and provide experimental evidence of its capability to efficiently represent spatio-temporal data and to predict future states of the environment. Unlike the previous works [3], [8], [4], [5], which can only introduce time into models that represent the environment by a discrete set of binary states, such as visibility of landmarks or cell occupancy in grids, our method is able to work with continuous and higher-dimensional variables, e.g. robot velocities, object positions, pedestrian flows, etc. Moreover, the method explicitly represents and predicts not only the value of a given state, but also its probabilistic distribution

at a particular time and location, which can be useful for task scheduling and planning [9].

Our experiments, based on real world data gathered over several weeks, confirm that the method achieved more accurate predictions than both static models and models that aim to represent time over a discretised space only.

II. RELATED WORK

In mobile robotics, the effects of environment variations were studied mainly from the perspective of localisation and mapping, because neglecting the environment change gradually deteriorates the ability of the robot to determine its position reliably and accurately. Assuming that the world is non-stationary, the authors of [2], [10], [11], [12], [6] proposed approaches that create, refine and update world models in a continuous manner. Furthermore, Ambrus et al. [13] demonstrated that the ability of continuous remapping not only allows to refine models of the static environment structure, but also opens up the possibility to learn object models from the spatial changes observed [14].

Unlike the aforementioned works, which focused on the spatial structure and appearance aspects of the changes observed, other authors [2], [15], [16], [3], [5] focused on modelling the temporal aspects. For example, [2] and [15] represent the environment dynamics by multiple temporal models with different timescales, where the best map for localisation is chosen by its consistency with current readings. Dayoub et al. [16] and Rosen et al. [8] used statistical methods to create feature persistence models and reasoned about the stability of the environmental states over time. Tipaldi et al. [3] proposed to represent the occupancy of cells in a traditional occupancy grid with a Hidden Markov Model. Krajník et al. [5] represent the probability of environment states in the spectral domain, which captures cyclic (daily, weekly, yearly) patterns of environmental changes as well as their persistence.

The aforementioned approaches demonstrated that considering temporal aspects (and especially their persistence and periodicity) in robotic models improves not only mobile robot localisation [5], [3], [2], but also planning [17], [18] and exploration [19]. However, these temporal representations were tailored to model the probability of a single state over time, and thus were applied only to individual components of the discretised models, e.g. cells in an occupancy grid [3], [19], visibility of landmarks [5], traversability of edges in topological maps [17] or human presence in a particular room [18]. Since the spatial interdependence of these components was neglected, the above models were actually considering only temporal and not spatial-temporal relations of the represented environments. This results not only in memory inefficiency (because of the necessity to model a high number of discrete states separately) but also in the inability of the representation to estimate environment states at locations where no measurements were taken, e.g. if a certain cell in an occupancy grid is occluded, its state is unknown even if the neighbouring cells are occupied, because the cell is part of a wall or ground.

Spatio-temporal relations of discrete environment models were investigated in [20], [21]. Kuczner et al. [20] proposed to model how the occupancy likelihood of a given cell in a grid is influenced by the neighbouring cells and showed that this representation allows to model object movement directly in an occupancy grid. A similar approach was proposed in [21], where the direction of traversal over each cell is obtained using an input-output Hidden Markov Model connected to neighboring cells. However, these models represent only local spatial dependencies and suffer from a major disadvantage of the discretised models – memory inefficiency. Therefore, in their latest work, [22] model a given set of spatio-temporal phenomena (the motion of people and wind flow) in a continuous domain, building their model by means of Expectation Maximisation. Moreover, [23] shows how to use this representation for robot motion planning in crowded environments.

O’Callaghan and Ramos [24] also argue in favour of continuous models, showing the advantages of Gaussian Mixture-based representations in terms of memory efficiency and utility for mobile robot navigation. In their latest work, [25] speed up building and updating of the proposed models by using an elegant combination of kernels and optimization methods. The speed-up achieved allows to recalculate the model relatively quickly, which keeps the model updated with the changes in the robot’s operational environment.

Unlike the work of Ramos et al. [25], which is aimed primarily at modelling the spatial structure, and [22], which aims to make short-term predictions of the motion of people, our aim is to create universal, spatial-temporal models capable of long-term predictions of various phenomena. Inspired by the ability of the continuous models [25], [22] to represent spatio-temporal phenomena and the predictive power of spectral representations [5], we propose a novel method which allows to introduce the notion of time into state-of-the-art spatial models used in mobile robotics. Unlike our previous work [5], which treats environmental states as independent despite their spatial proximity and is applicable to binary states only, the proposed method can be applied to continuous, multi-dimensional representations, e.g. object positions, movements of people, gas concentration, etc. Moreover, the method explicitly represents and predicts not only the value of a given state, but also its probabilistic distribution at a particular time and location, which can be useful for task scheduling and planning.

III. METHOD

A. Example scenario

As an example, let us consider a robot providing an information terminal service in an office building or a conference guide that has to provide directions to certain events at locations where people commonly gather before the events. To provide the service efficiently, the robot has to position itself close to areas with a high level of pedestrian traffic. However, to avoid causing nuisance to people and improve robot performance, the robot should arrive at these locations before they become congested. Thus, the robot

has to anticipate the likelihood of occurrence of people at a given time and place based on a model, which is built from data gathered by the robot's people detection system. Such a system provides the robot with $x, y \in \mathfrak{R}^2$ positions of the people in the robot's field of view. To be able to navigate to this spot efficiently, the robot should know how to predict if, and how quickly, it is able to reach the desired destination from the current position. To do so, it has to predict which of the potential paths will be blocked (e.g. due to doors being closed) and how quickly it will be able to traverse them. Furthermore, the robot has to reason about its ability to localise itself reliably along the path it plans to traverse. This means that the robot needs to use past data from its navigation system, including its velocity $v \in \mathfrak{R}$ along a given path and success of the path traversal $s \in \{0, 1\}$. The number of people o , robot speed v and path traversal success s around the position (x, y) might be influenced by time, since people gather in different areas at different times (meeting rooms, cafeterias, etc.), and the robot speed is influenced by the congestion of the corridors it moves through and the doors on the path that are more likely to be open during busy hours than otherwise.

Thus, one can assume that the values of o , v and s , which form the state of the environment, will appear with a different frequency depending on the time t and location x, y . In particular, they are likely to be influenced by patterns of human activity, which are strongly influenced by the time of day and the day of the week. Our method provides a unified way to identify the dependence of these variables on time by modelling the spatio-temporal distribution of their occurrences. The problem of finding such distributions is that while the modelled space is constrained, and thus one can gather an arbitrary number of measurements from a given location, time unfolds indefinitely and it is not possible to obtain measurements with the same t , which makes calculation of the temporal density of some phenomena difficult. In our case, we assume that the time domain exhibits certain periodic properties, and project the entire time-line into a multi-dimensional, but constrained warped space, where the notion of event density makes sense.

B. Method Overview

Let us assume that a robot already gathered a training set containing l measurements of a given phenomenon, obtaining tuples (a_i, \mathbf{x}_i, t_i) , where $i \in \{1 \dots l\}$, the vector \mathbf{x}_i describes the location of the measurement (e.g. position of a detected person or obstacle), t_i corresponds to the time of the measurement and a_i represents the measurement's value, e.g. the number of detected people, likelihood of an obstacle or robot velocity in the vicinity of (\mathbf{x}_i, t_i) .

Our method aims to find a $p(a|\mathbf{x}, t)$, which would represent the conditional probability density function of the variable a given the position \mathbf{x} and time t . The proposed method is composed of five stages, which are performed in an iterative manner:

- 1) initialization;
- 2) spatio-temporal clustering;

- 3) model error estimation;
- 4) identification of periodicities;
- 5) and hypertime space extension.

To initialize the algorithm, we first set an index h , which characterises the number of periodicities taken into the temporal model, to zero and we store all measurements (a_i, \mathbf{x}_i) in ${}^h\mathbf{x}_i$. In the *spatio-temporal clustering* stage, we cluster the vectors $({}^h\mathbf{x}_i)$, obtaining a Gaussian mixture model, which represents the spatio-temporal distribution of the given phenomenon and allows to calculate conditional probability function $p_h(a|\mathbf{x}, t)$. In the *model error estimation*, we calculate the mean μ_i of $p(a|\mathbf{x}_i, t_i)$ for all training samples. Then we calculate the time series ${}^h\epsilon(t_i)$ as ${}^h\epsilon(t_i) = \mu_i - a_i$ and its mean squared value E_h . Then, during the *identification of periodicities*, we perform spectral analysis of ${}^h\epsilon(t_i)$, extract the most prominent spectral component, and store its period as T_{h+1} . After that, we perform the *hypertime space extension*, which extends each vector ${}^h\mathbf{x}_i$ by 2 dimensions representing a given periodicity of the temporal domain, i.e.

$${}^{h+1}\mathbf{x}_i \leftarrow ({}^h\mathbf{x}_i, \cos(2\pi t_i/T_{h+1}), \sin(2\pi t_i/T_{h+1})). \quad (1)$$

Then, we increment h by one and repeat the steps of *spatio-temporal clustering* and *model error estimation* on the now extended vectors ${}^h\mathbf{x}_i$, obtaining a new error E_h . We compare the model error E_h calculated with the error obtained in the previous iteration E_{h-1} and if $E_h < E_{h-1}$, we proceed with *identification of periodicities* and *hypertime space extension*, extending the vector ${}^h\mathbf{x}_i$ with another two dimensions representing another potential periodicity of the modeled phenomena. In cases where the model error starts to increase, i.e. if $E_h \geq E_{h-1}$, we store the model $p_{h-1}(a, \mathbf{x}, t)$ from the previous iteration as $p(a, \mathbf{x}, t)$ and terminate the method.

The resulting model allows to estimate the likelihood of each value a of a given phenomena at location \mathbf{x} and time t . In our experiments, we show that the function $p(a, \mathbf{x}, t)$ allows to predict the visibility of image features, door states, robot velocity and number of people occurrences within a given spatio-temporal volume.

C. Spatio-Temporal Clustering

We represent the probability density function $p(a, \mathbf{x}, t)$ by a mixture of Gaussian models in the hypertime space as follows:

$$p(a, \mathbf{x}, t) = \gamma \sum_{j=1}^n w_j u_j(a, \mathbf{x}, {}^h\mathbf{t}), \quad (2)$$

where $u_j(a, \mathbf{x}, {}^h\mathbf{t})$ is a multivariate Gaussian function of the j^{th} cluster, w_j is the cluster weight, ${}^h\mathbf{t} = (\cos(2\pi \frac{t}{T_1}), \sin(2\pi \frac{t}{T_1}), \dots, \cos(2\pi \frac{t}{T_h}), \sin(2\pi \frac{t}{T_h}))$ is the projection of time in the hypertime space and γ is a scaling constant. Calculation of the model, i.e. computation of weights, means and covariances of the Gaussian mixture model is performed by an Expectation Maximisation scheme, discussed in detail in Chapter IV.

D. Model error estimation

Projecting the linear time t onto the circular hypertime space (or its inverse) inevitably changes the scale of the calculated spatio-temporal density. This is because several time instants t can project into the same area of hypertime. Thus, we first need to determine the scaling factor γ in such a way that the mean value of a calculated from the model (2) over the training set vectors (\mathbf{x}_i, t_i) is equal to the average value of a_i on the training set:

$$\gamma = \frac{\sum_{i=1}^l a_i}{\sum_{i=1}^l \int_a \sum_{j=1}^n w_j u_j(a, \mathbf{x}_i, {}^h \mathbf{t}_i) da} \quad (3)$$

After calculating the scaling factor γ , we compute an estimate of a_i at each training test point defined by location \mathbf{x}_i and time t_i by calculating the mean μ_i :

$$\mu_i = \int a p(a, \mathbf{x}_i, t_i) da, \quad (4)$$

Then, we calculate the error ${}^h \varepsilon(t_i)$ as the difference between the mean and the measured values a_i

$${}^h \varepsilon(t_i) = \mu_i - a_i. \quad (5)$$

Finally, we calculate the mean squared error of the current model as

$$E_h = \sqrt{\sum_{i=1}^l {}^h \varepsilon^2(t_i)} = \sqrt{\sum_{i=1}^l (\mu_i - a_i)^2}. \quad (6)$$

We compare the mean squared error E_h with the one calculated in the previous iteration E_{h-1} . If E_h is smaller than E_{h-1} , then we increase the h by one and perform another iteration of the method.

E. Identification of periodicities and hypertime extension

To identify the periodicities in the error, we use a Fourier-transform scheme. However, since the data collections for the experiments were performed by a system operating in real-world conditions, they were not collected in a (temporally) regular manner. Thus, we process the time series ${}^h \varepsilon(t_i)$ by the FreMEn method [5], which is able to find periodicities in non-uniform and sparse data. In particular, we calculate the most prominent periodicity T_h in the error time-series as:

$$T_h = \arg \max_{T_k} \sum_{i=1}^l |({}^h \varepsilon(t_i) - {}^h \hat{\varepsilon}) e^{-j2\pi t_i/T_k}|, \quad (7)$$

where ${}^h \hat{\varepsilon}$ is an average error ${}^h \varepsilon(t_i)$. After establishing the T_h , we extend all vectors of the training set ${}^{h-1} \mathbf{x}(t_i)$ by adding another two components $(\cos(2\pi t_i/T_h), \sin(2\pi t_i/T_h))$, i.e.

$${}^h \mathbf{x}_i \leftarrow ({}^{h-1} \mathbf{x}_i, \cos(2\pi t_i/T_h), \sin(2\pi t_i/T_h)). \quad (8)$$

Thus, at the start of our method, each vector ${}^0 x_i$ contains only the spatial information, i.e. ${}^0 x_i = (a_i, \mathbf{x}_i)$, but at the end, the vector contains $2h$ additional dimensions modeling the periodicities observed in the training data.

IV. DISCUSSION OF CLUSTERING

While the hypertime extension, error estimation and periodicity estimation steps of the method are quite straightforward, deterministic and computationally inexpensive, the way we build the model of the probability distribution over the hypertime space is key to the method's predictive efficiency. The main issue of the hypertime space is its sparsity, because the time, which is linear and one-dimensional, is projected onto a single-dimensional curve lying on a multi-dimensional hypersphere. This causes problems with the numerical stability of many algorithms that we tested. Thus, we dedicated a significant effort into testing various clustering methods, their initialisations and metrics. Since this letter is concerned with the idea of using the hypertime space to allow robots to take into account cyclic environment variations during their long-term operation, we will give a short overview of the methods tested on the scenarios described in Section V. A thorough comparison of the aforementioned methods and settings is beyond the scope of this article and will be presented in a separate paper.

We evaluated different clustering methods and metrics to model distributions in the spatio-temporal hyperspace. The Gustafson–Kessel algorithm [26] with fuzzy membership degrees [27], which uses Mahalanobis distance, worked well on a space with two temporal and two spatial dimensions. However, as the number of modelled periodicities (and thus, the number of temporal dimensions) grew, the method became unstable and did not provide meaningful results. This confirms our previous results [28], which showed that Gustafson–Kessel does not achieve good performance in high-dimensional spaces. We also evaluated other distribution modeling methods described in [29]. In particular, we thoroughly evaluated the performance of fuzzy k-means clustering [30] and classical k-means [31]. Surprisingly, the only clustering method able to partition the hypertime-space into meaningful clusters was classical k-means [31]. We also attempted to model the spatio-temporal hyperspace by a mix of Gaussian distributions, determined by an Expectation-Maximisation scheme similar to [25], [22]. While these achieved good results in most cases, they sometimes exhibited numerical instabilities similar to the Gustafson–Kessel algorithm [26]. However, the effects of numerical instability are possible to detect through eigenvalue analysis of the covariance matrices and if needed, the EM can be restarted with different initialization or with the covariances matrices restricted to diagonal-only.

We also evaluated a variety of different metrics, e.g. Euclidean, Minkowski, Chebyshev, etc., as well as their squared and square rooted variants. The Minkowski metric is recommended for clustering in higher dimensions [32], but it performed no better than Euclidean distance when compared on hypertime space. As mentioned earlier, we also evaluated Mahalanobis distance without satisfactory results. Inspired by [33][34], we experimented with the mixtures of cosine distance for hypertimes and the aforementioned metrics for other variables.

Finally, in our experiments we compare two clustering methods, which provided the most satisfactory results, the HyperTime Expectation Maximisation (*HyT-EM*) and Hyper-Time K-Means (*HyT-KM*).

The *HyT-EM* method is based on the Expectation Maximisation scheme implementation from the OpenCV library. As the method requires to specify the number of clusters, we indicate the the method name as *HyT-EM.k*, where k is the number of clusters used during the experiments (Section V). As mentioned before, the only problem of the method is its occasional numerical instability, which, if detected, is solved by automated restart with different initial positions of the clusters.

The *HyT-KM* method is based on the NumPy package, popular for scientific computing in the Python language. *HyT-KM* first initialises the cluster centres using k-means, while using the cosine distance for temporal and Euclidean distance for spatial dimensions. After initialisation, it calculates the covariace matrices of the clusters and then it proceeds with the standard Expectation Maximisation procedure, while using the cosine distance for temporal dimensions. Unlike *HyT-EM*, *HyT-KM* does not require to specify the number of clusters in advance. Instead, the algorithm tries to analyse the temporal structure of the hypertime-space prior to the *hypertime expansion* step. In particular, it starts with $n = 1$ clusters and calculates the sum of amplitudes $T_{\Sigma}(n)$ and $T_{\Sigma}(n + 1)$ of the frequency spectrum of the error ${}^h\epsilon(t_i)$

$$T_{\Sigma}(n) = \sum_{k=1}^K \sum_{i=1}^l |({}^h\epsilon(t_i) - {}^h\hat{\epsilon}) e^{-j2\pi t_i/T_k}|. \quad (9)$$

If $T_{\Sigma}(n) > T_{\Sigma}(n + 1)$, the model with $n + 1$ clusters is stored and the number of clusters n is incremented by 1. If $T_{\Sigma}(n) \leq T_{\Sigma}(n + 1)$, the method simply proceeds with the *hypertime expansion* step.

V. EXPERIMENTS

The purpose of the experimental evaluation is to assess the predictive capability of the proposed method and its utility for different robotic tasks. The performance of the method is evaluated in four different scenarios, which require predictions of variables of different dimensionalities. The data for these experiments were collected by robotic sensors in real world conditions over periods of several weeks. These scenarios correspond to our original motivational example from Section III-B, where we discussed how a long-term operating robot will benefit from the predictive capabilities of models that explicitly represent temporal behaviour of environment states with different dimensions. To evaluate the efficiency of our method, we compare five different temporal models: *Mean*, which predicts a value as an average of its past measurements, *Hist_n*, which divides each day into n intervals and predicts the given variable as an average in a relevant time of a day, *FreMEn_m*, which extracts m periodic components from the variable's history and uses these periodicities for prediction, *HyT-EM.k*, which uses the expectation-maximisation of k -component Gaussian Mixture Model over the hypertime space, and finally *HyT-KN*, as

described in Section IV. The experimental evaluation is performed by an automated system [35], which first optimises each method's parameters (number of intervals n , number of periodicities m , and number of clusters k) and then runs a series of pairwise t-tests to determine which methods perform statistically significantly better than other ones. To enable the reproducibility of the results, the evaluation system and the datasets are available online [36].

A. Door state

The first scenario concerns a single binary variable, which corresponds to the state of a university office door. The door was continuously observed by an RGB-D camera for 10 weeks to obtain the training set, and for another 10 weeks to obtain 10 testing sets, each one week long. Since the RGB-D data processing was rather simple, the data contains noise, because people moving through the door caused the system to indicate incorrectly that the door was closed.

To compare the efficiency of the predictions, we calculated the mean squared error ϵ of the various temporal models' predictions $p(t)$ to the ground truth $s(t)$ as $\epsilon = \sqrt{\sum_T (p(t) - s(t))^2 / |s(t)|}$. The results indicate that both

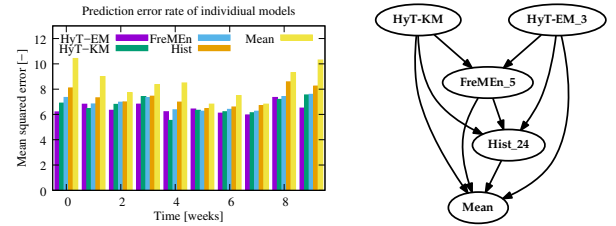


Fig. 2. Door state prediction error. The left figure shows the MSE for the training (week 0) and testing (weeks 1-9) datasets. An arrow from model A to model B in the right figure indicates that A's prediction error is statistically significantly lower than prediction error of model B.

hypertime-based models outperformed the other ones, including FreMEn [5]. Both methods indicated that the most prominent periodicities corresponded to one day, four hours and one week.

B. Topological localisation

In this scenario a robot has to determine its location in an open-plan university office based on the current image from its onboard camera and a set of pre-learned appearance models of several locations. Since the appearance of these locations changes over time, it is beneficial to utilise appearance models that explicitly represent the appearance variations [37], [5], [8], [6]. This experiment compares the impact of different temporal models, which predict the visibility of environmental features at these locations, on the robustness of robot localisation. To gather data about the changes in feature visibility, a SCITOS-G5 robot visited eight different locations of the university office every 10 minutes for one week, collecting a training dataset with more than 8000 images. After one week, the robot visited the same locations every 10 minutes for one day, collecting 1152 time-stamped images used for testing. The training set

images were then processed by the BRIEF method [38], which shows good robustness to appearance changes [39]. The extracted features belonging to the same locations were matched and we obtained their visibility over time, which was then processed by the temporal models evaluated. Thus, we obtained a dynamic appearance-based model of each location that can predict which features are likely to be visible at a particular time.

During testing, the robot uses these models to calculate the likelihood of the features' visibility at each of the locations at the time it captured an image by its onboard camera (or extracted a time-stamped image from the testing set). In particular, it selects the n most likely-to-be-visible features at each location and time, matches these features to the features extracted from its onboard camera (or testing set) image, and determines the model with the most matches as its current location. The localisation error is calculated as the ratio of cases when the robot incorrectly estimated its location to the total number of images in the testing set. The dependence of the average localisation error on the particular temporal model and number of features n used for localisation is shown in Figure 3. The results indicate that the

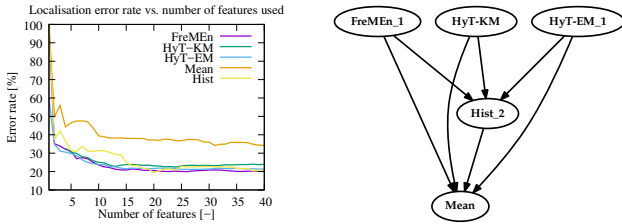


Fig. 3. Temporal model performance for feature-based topological localisation. The left figure shows the dependence of localisation error rate on the number of features predicted by a given temporal model. An arrow from A to B in the right indicates that A's localisation error rate is statistically significantly lower than localisation error rate of model B.

localisation robustness of the methods that take into account the rhythmic nature of the appearance changes outperform the *Mean* method, which relies on the most stable image features. Moreover, the methods that model these cyclic changes in a continuous manner perform better than the *Hist* method which models different times of the day in separate, as shown in Figure 3.

C. Velocity prediction

This scenario concerns the ability of our representation to predict the velocity of a robot while navigating through a given area, which depends on how cautiously it has to move due to the presence of humans. Thus, this experiment is concerned with the ability of our method to predict a one-dimensional continuous variable (robot velocity) for a given time and location.

The velocities and times of navigation for our evaluation were obtained from a database obtained with a SCITOS-G5 mobile platform, which gathered data in an open plan

research office for more than 10 weeks. Typically, the average velocity of the robot did not show much variation, but in cases it had to navigate close to workspaces and through doors, the velocity varied significantly. To evaluate the ability of our approach to predict the robot velocity, we split the dataset into an 8-weeks long training set and two testing sets of 1-week duration. As in the case of door

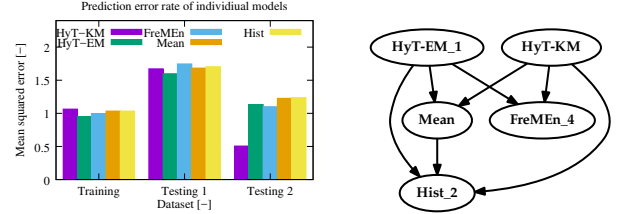


Fig. 4. Navigation velocity prediction error. The left figure shows the mean squared error for the training and testing sets. An arrow from A to B in the right indicates that A's prediction error is statistically significantly lower than velocity prediction error of model B.

state prediction, we calculated the mean square error of the predictions provided by our models, and compared them to find out which of the methods provide the most accurate predictions. Our results indicated that both Hypertime-based methods outperformed the other ones, as shown in Figure 4.

D. Human presence

Finally, we validated the proposed approach on 2-dimensional data indicating the positions of people in several corridors of the Isaac Newton Building at the University of Lincoln. Data collection was performed by a mobile robot equipped with a Velodyne 3d laser rangefinder, which was placed at a T-shaped junction so that its laser rangefinder was able to scan the three connecting corridors simultaneously. To detect and localize people in the 3d point clouds provided by the scanner, we used an efficient and reliable person detection method [40]. Since we needed to recharge the robot occasionally, we did not collect data on a 24/7 basis and recharged the robot batteries during nights, when the building is vacant and there are no people in the corridors. Thus, our dataset spanned from early mornings to late evening over several weekdays. Each day contains approximately 28000 entries, which correspond to hundreds of walks by people through the monitored corridor. To

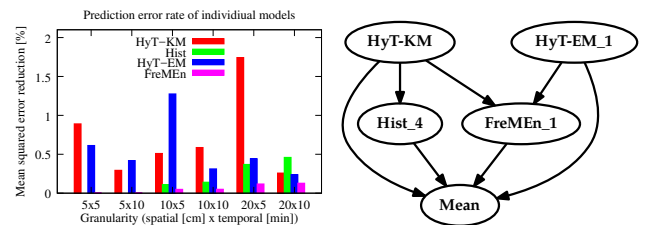


Fig. 5. Human presence prediction results. The left figure shows how the mean squared error reduced for a particular model compared to the *Mean* model. An arrow from A to B in the right indicates that A's prediction error is statistically significantly lower than velocity prediction error of model B.

quantitatively evaluate the model quality, we again split the gathered data into training and test sets, and learn the model from the training set only. Then, we partition the timeline of the test data into a spatio-temporal 3d grid. For each cell g , we count the number of detections d_g that occurred and compare this value with the value p_g predicted by a given spatio-temporal model. To better visualise the methods' prediction improvements, we show the reduction of the mean square error compared to the *Mean* model in Figure 5. To make a comparison with other models, we apply the FreMEN method on each of the grid cells independently and then predict the most likely number of events at a given time in a particular cell. Since the error is dependent on the partitioning used, we tested the method for grids of various cell sizes ranging from 5 to 20 cm and 5 to 30 minutes.

To demonstrate the model's ability to estimate the spatio-temporal distribution over time, we let it predict the most likely occurrence of people for different times. Figure 1 and video [7] show that the predicted distributions of the people depend on time and follow the shape of the corridor (which is not part of the training data).

VI. CONCLUSION

We presented a novel approach for spatio-temporal modeling for robots that are required to operate for long periods of time in changing environments. The method models the time domain in a multi-dimensional hyperspace, where each pair of dimensions represents one periodicity observed in the data. This multi-dimensional, warped time model is used to extend the state space representing a given phenomenon. By projecting the robot's observations into this space-hypertime and clustering them, we create a continuous, spatio-temporal model (distribution) of the phenomenon observed by the robot. Knowledge of the spatio-temporal distribution is then used to predict the occurrence of a given phenomenon at a given time.

Using data collected by a mobile robot over several weeks, we show that the method can represent the spatio-temporal dynamics of binary and continuous variables, and use the representation to make predictions of the future environment states, resulting in significantly better performance than the previous state of the art.

One of the major problems we have encountered is the instability of clustering methods on the multi-dimensional hypertime space. Thus, in the future, we will evaluate the performance of different clustering algorithms on the proposed representation. Furthermore, the observed distribution is heavily influenced by the way in which the robot samples the environment, i.e. where and when it takes the measurements. Thus, we will study spatio-temporal exploration methods, which will allow a mobile robot to automatically select a location and time to obtain data useful to refine and improve the spatio-temporal model.

Finally, to allow use of the method by other researchers, we provide its baseline open source code and datasets at <http://fremen.uk>.

REFERENCES

- [1] N. Hawes *et al.*, "The strands project: Long-term autonomy in everyday environments," *IEEE Robotics and Automation Magazine*, 2016.
- [2] P. Biber and T. Duckett, "Experimental analysis of sample-based maps for long-term slam," *International Journal of Robotics Research*, 2009.
- [3] G. D. Tipaldi *et al.*, "Lifelong localization in changing environments," *The International Journal of Robotics Research*, 2013.
- [4] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments," in *IROS*, 2013.
- [5] T. Krajník, J. P. Fentanes, J. Santos, and T. Duckett, "FreMEN: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, 2017.
- [6] W. S. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *IJRR*, 2013.
- [7] T. Krajník, "Warped hypertime representations for long-term mobile robot autonomy," 2018. [Online]. Available: <https://youtu.be/4SW4j7DDxYE>
- [8] D. M. Rosen, J. Mason, and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016, pp. 1063–1070.
- [9] L. Mudrová, B. Lacerda, and N. Hawes, "An integrated control framework for long-term autonomy in mobile service robots," in *Proc. of the 7th European Conf. on Mobile Robotics (ECMR)*, Lincoln, United Kingdom, 2015.
- [10] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *The International Journal of Robotics Research*, vol. 29, no. 9, pp. 1131–1153, 2010.
- [11] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *International Conference on Intelligent Robots and Systems*, 2009.
- [12] S. Hochdorfer and C. Schlegel, "Towards a robust visual SLAM approach," in *International Conference on Advanced Robotics*, 2009.
- [13] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [14] T. Faelulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *Robotics and Automation Letters*, 2016.
- [15] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2002, pp. 409–414 vol.1.
- [16] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, 2011.
- [17] J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide, "Now or later? predicting and maximising success of navigation actions from long-term experience," in *ICRA*, 2015.
- [18] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, and T. Duckett, "Where's waldo at time t? using spatio-temporal models for mobile robot search," in *ICRA*, 2015.
- [19] J. M. Santos, T. Krajník, J. Pulido Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4d spatio-temporal maps," *Robotics and Automation Letters*, 2016.
- [20] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1196–1201.
- [21] Z. Wang, R. Ambrus, P. Jensfelt, and J. Folkesson, "Modeling motion patterns of dynamic objects by IOHMM," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1832–1838.
- [22] T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal, "Enabling flow awareness for mobile robots in partially observable environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1093–1100, 2017.
- [23] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on gaussian mixture fields," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 6176–6181.

- [24] S. T. OCallaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.
- [25] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.
- [26] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*. IEEE, 1979, pp. 761–766.
- [27] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, 1973.
- [28] T. Vintr, L. Pastorek, and H. Rezankova, "Autonomous robot navigation based on clustering across images," *Research and Education in Robotics-EUROBOT 2011*, pp. 310–320, 2011.
- [29] R. Kruse, C. Döring, and M.-J. Lesot, "Fundamentals of fuzzy clustering," *Advances in fuzzy clustering and its applications*, pp. 3–30, 2007.
- [30] J. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.
- [31] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. California, USA, 1967, pp. 281–297.
- [32] P. J. Groenen, U. Kaymak, and J. van Rosmalen, "Fuzzy clustering with minkowski distance functions," *Fuzzy Clustering and its Applications*, Wiley, pp. 53–68, 2007.
- [33] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on gaussian mixture fields," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 6176–6181.
- [34] A. Roy, S. K. Parui, and U. Roy, "Swgmm: a semi-wrapped gaussian mixture model for clustering of circular-linear data," *Pattern Analysis and Applications*, vol. 19, no. 3, pp. 631–645, 2016.
- [35] T. Krajník, M. Hanheide, T. Vintr, K. Kusumam, and T. Duckett, "Towards automated benchmarking of robotic experiments," in *ICRA Workshop on Reproducible Research in Robotics*, 2017.
- [36] T. Krajník, "The frequency map enhancement (FreME) project repository," <http://fremen.uk>.
- [37] P. Neubert, N. Sünderhauf, and P. Protzel, "Superpixel-based appearance change prediction for long-term navigation across seasons," *RAS*, 2014.
- [38] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Proceedings of the ICCV*, 2010.
- [39] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, 2016.
- [40] Z. Yan, T. Duckett, N. Bellotto *et al.*, "Online learning for human classification in 3d lidar-based tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

K

KEY ARTICLE [40] - JOURNAL OF INTELLIGENT AND ROBOTIC SYSTEMS

©[2014] Springer. This is a post-peer-review, pre-copyedit version of an article published in Journal of Intelligent and Robotic Systems. The final authenticated version is available online at: <http://dx.doi.org/10.1007/s10846-014-0041-x>

Tomáš Krajník · Matías Nitsche · Jan Faigl · Petr Vaněk · Martin Saska · Libor Přeučil · Tom Duckett · Marta Mejail

A practical multirobot localization system

Received: date / Accepted: date

Abstract We present a fast and precise vision-based software intended for multiple robot localization. The core component of the software is a novel and efficient algorithm for black and white pattern detection. The method is robust to variable lighting conditions, achieves sub-pixel precision and its computational complexity is independent of the processed image size. With off-the-shelf computational equipment and low-cost cameras, the core algorithm is able to process hundreds of images per second while tracking hundreds of objects with millimeter precision. In addition, we present the method's mathematical model, which allows to estimate the expected localization precision, area of coverage, and processing speed from the camera's intrinsic parameters and hardware's processing capacity. The correctness of the presented model and performance of the algorithm in real-world conditions is verified in several experiments. Apart from the method description, we also make its source code public at <http://purl.org/robotics/whycon>; so, it can be used as an enabling technology for various mobile robotic problems.

The European Union supported this work within its Seventh Framework Programme project ICT-600623 "STRANDS". The Ministry of Education of the Czech Republic and Argentina have given support through projects 7AMB12AR022 and ARC/11/11. The work of J. Faigl was supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P. Christian Dondrup and David Mullineaux are acknowledged for help with experiments.

T. Krajník · T. Duckett
Lincoln Centre for Autonomous Systems,
School of Computer Science, University of Lincoln
E-mail: tkrajnik,tduckett@lincoln.ac.uk

M. Nitsche · M. Mejail
Laboratory of Robotics and Embedded Systems,
Fac. of Exact and Natural Sciences, Univ. of Buenos Aires
E-mail: mnitsche,marta@dc.uba.ar

J. Faigl · P. Vaněk · M.Saska · L. Přeučil · T. Krajník
Faculty of Electrical Engineering,
Czech Technical University in Prague
E-mail: faigl,j.saskam,vanekpe5,preucil,krajnt1@fel.cvut.cz

1 Introduction

Precise and reliable position estimation remains one of the central problems of mobile robotics. While the problem can be tackled by Simultaneous Localization and Mapping approaches, external localization systems are still widely used in the field of mobile robotics both for closed-loop mobile robot control and for ground truth position measurements. These external localization systems can be based on an augmented GPS, radio, ultrasound or infrared beacons, or (multi-)camera systems. Typically, these systems require special equipment, which might be prohibitively expensive, difficult to set up or too heavy to be used by small robots. Moreover, most of these systems are not scalable in terms of the number of robots, i.e., they do not allow to localize hundreds of robots in real time. This paper presents a fast vision-based localization system based on off-the-shelf components. The system is precise, computationally efficient, easy to use, and robust to variable illumination.

The core of the system is a detector of black-and-white circular planar ring patterns (roundels), similar to those used for camera calibration. A complete localization system based on this detector is presented. The system provides estimation of the roundel position with precision in the order of millimeters for distances in the order of meters.

The detection with tracking of a single roundel pattern is very quick and the system is able to process several thousands of images per second on a common desktop PC. This high efficiency enables not only tracking of several hundreds of targets at a camera frame-rate, but also implementation of the method on computationally restricted platforms. The fast update rate of the localization system allows to directly employ it in the feedback loop of mobile robots, which require precise and high-frequency localization information.

The system is composed of low-cost off-the-shelf components only – a low-end computer, standard webcam, and printable patterns are the only required elements.

The expected coverage, precision, and image processing speed of the system can be estimated from the camera resolution, computational power, and pattern diameter. This allows the user to choose between high-end and low-end cameras, estimate if a particular hardware platform would be able to achieve the desired localization frequency, and calculate a suitable pattern size for the user’s specific application.

Ease of the system setup and use are also driving factors of the proposed implementation, which does not require user-set parameters or an intricate set-up process. The implementation also contains an easy tool for camera calibration, which, unlike other calibration tools, does not require user interaction. At the same time, the implementation is proposed as a library, which can be integrated into commonly used computer vision frameworks, such as OpenCV.

The main intention of this paper is to present the system principle, its theoretical properties and real performance characteristics with respect to the intended application. Therefore, we present a model of the localization arising from theoretical analyses of the vision system and experimental evaluation of the system performance in real scenarios with regard to its practical deployment.

2 Related work

External localization systems are widely used in the field of mobile robotics, either for obtaining ground truth pose data or for inclusion in the control loop of robots. In both scenarios, it is highly desirable to have good precision and high-frequency measurements. Here, both of these aspects are analysed in related works and are specifically addressed in the proposed system.

Localization systems for mobile robots comprise an area of active research; however, the focus is generally on internal localization methods. With these methods, the robot produces one or more estimates of its position by means of fusing internal sensors (either exteroceptive or proprioceptive). This estimation can also be generally applied when either a map of the environment exists *a priori* or when the map is being built simultaneously, which is the case of SLAM approaches [1]. When these internal localization systems are studied, an external positioning reference (i.e., the ground truth) without any cumulative error is fundamental for a proper result analysis. Thus, this research area makes use of external localization systems.

While the most well-known external localization reference is GPS, it is also known that it cannot be used indoors due to signal unavailability. This fundamental limitation has motivated the design of several localization principles, which can be broadly divided into two major groups by means of the type of sensors used: active or passive.

In the former group, several different technologies are used for the purpose of localization. One example [2] of active sensing is the case of a 6DoF localization system comprised of target modules, which include four LED emitters and a monocular camera. Markers are detected in the image and tracked in 3D, making the system robust to partial occlusions and increasing performance by reducing the search area to the vicinity of the expected projection points. Experiments with this system were performed using both ground and aerial robots. The mean error of the position estimation is in the order of 1 cm, while the maximum error is around 10 cm. The authors note that for uncontrolled lighting scenarios passive localization systems appear to be more suitable.

Another active sensor approach is the NorthStar [3] localization system, which uses ceiling projections as a non-permanent ambient marker. By projecting a known pattern, the camera position can be obtained by reprojection. The authors briefly report the precision of the system to be around 10 cm.

In recent works, the most widely used approach is the commercial motion capture system from ViCon [4]. This system is comprised of a series of high-resolution and high-speed cameras, which also have strong infrared (IR) emitters. By placing IR reflective markers on mobile robots, sub-millimeter precision can be achieved with updates up to 250Hz. Due to these qualities, ViCon has become a solid ground-truth information source in many recent works and, furthermore, has allowed development of closed-loop aggressive maneuvers for UAVs inside lab environments [5]. However, this system is still a very costly solution, and therefore, it is not applicable to every research environment. This issue has motivated several works proposing alternative low-cost localization systems.

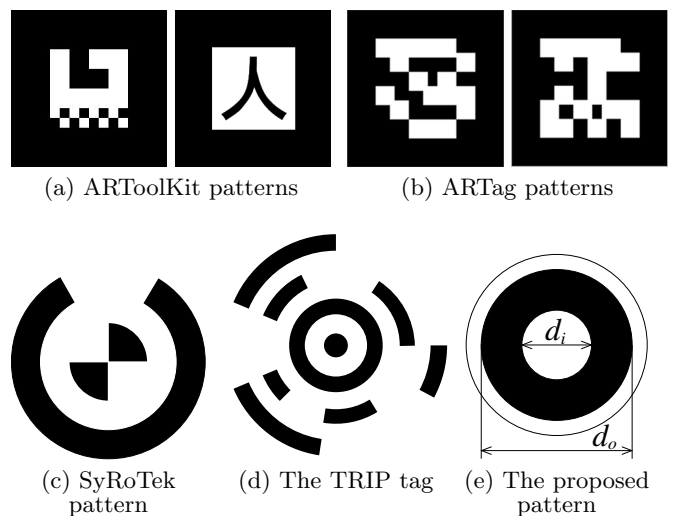


Fig. 1: Patterns used in passive vision-based global localization systems.

Several passive vision-based localization methods were also proposed in recent literature, using simple planar printable patterns, which reduce significantly the cost and difficulty of use and setup. Several of these works employ augmented-reality oriented markers, which not only permit obtaining the pose of the target but can also encode additional information like target ID. In this area, the software libraries most widely used for this purpose are ARTag [6] and ARToolkit+ [7], both based on its predecessor ARToolkit [8], see examples of patterns in Figure 1. These target detectors were used in several works in order to obtain localization information about mobile robots, either explicitly as a part of a pose estimation system [9, 10] or as ground-truth data [11].

In [9], ARToolkit markers are used for obtaining the pose of several ground robots. The homography from 3D-to-2D space (ground floor) is computed by defining the work area by placing four ad-hoc markers, which are manually detected in the image. In more recent work, the authors proposed the ARTag [6] system that was later extensively analysed in [12]. However, the analysis is focused on detection and confusion rates, and it does not report the real accuracy in position estimation. Similar systems are explored in [13], but details of their precision are not reported.

One particular system, which is based on AR markers similar to ARTag and ARToolkit, is ArUco [14]. The main aspects of this method are: easy integration into C++ projects, based exclusively on OpenCV and a robust binary ID system with error correction which can handle up to 1024 individual codes. The detection process of AR markers in ArUco consists of: an adaptive thresholding step, contour extraction and filtering, projection removal and code identification. When the intrinsic camera parameters are known, the extrinsic parameters of the target can be obtained. Due to the free availability of the implementation and lack of performance and precision reports, this system is analyzed in the presented work, see Section 6.5.

Since the previous pattern detectors were conceived for augmented-reality applications, other works propose alternative target shapes, which are specifically designed for vision-based localization systems with high precision and reliability. Due to several positive aspects, circular shaped patterns appear to be the best suited as fiducial markers in external localization systems. This type of pattern can be found (with slight variations) in several works [15–18].

The SyRoTek e-learning platform [19] uses a ring shaped pattern with a binary tag (see Figure 1c) to localize up to fourteen robots in a planar arena. The pattern symmetry is exploited to perform the position and orientation estimation separately, which allows to base the pattern localization on a two-dimensional convolution. Although this convolution-based approach has proven to be reliable enough to achieve 24/7 operation, its computational complexity still remains high, which

lead to its implementation on alternative platforms such as FPGA [20].

In [16], a planar pattern consisting of a ring surrounding the letter “H” is used to obtain the relative 6DoF robot pose with an on-board camera and IMU (Inertial Measurement Unit) to resolve angular ambiguity. The pattern is initially detected by binarization using adaptive thresholding and later processing for connected component labeling. For classifying each component as belonging to the target or not, a neural network (multilayer perceptron) is used. The input to the neural network is a resized 14×14 pixel image. After testing for certain geometric properties, false matches are discarded. Positive matches corresponding to the outer ring are processed by applying the Canny edge detector and ellipse fitting, which allows computation of the 5DoF pose. Recognition of the “H” letter allows to obtain the missing yaw angle. The precision in 3D position is in the order of 1 cm to 7 cm depending on the target viewing angle and distance, which was at the maximum around 1.5 m.

Probably the most similar approach to the proposed system in this work is the TRIP localization system [17]. In TRIP, the pattern comprises of a set of several concentric rings, broken into several angular regions, each of which can be either black or white. The encoding scheme, which includes parity checking, allows the TRIP method to distinguish between 3^9 patterns. For detecting the tags, adaptive thresholding is performed and edges are extracted. TRIP only involves processing edges corresponding to projections of circular borders of the ring pattern, which are detected using a simple heuristic. These edges are used as input to an ellipse fitting method and then the concentricity of the ellipses is checked. TRIP achieves a precision similar to [16] in position estimation (the relative error is between 1% and 3%), but only a moderate performance (around 16 FPS at the resolution 640×480) is achieved using an 1.6 GHz machine. The authors report that the adaptive thresholding step is the most demanding portion of the computation. To the best of our knowledge, there is no publicly available implementation.

Finally, a widely used, simple and freely available circular target detector can be found in the OpenCV library. This “SimpleBlobDetector” class is based on traditional blob detection methods and includes several optional post-detection filtering steps, based on characteristics such as area, circularity, inertia ratios, convexity and center color. While this implementation is originally aimed for circular target detection, by tuning the parameters it is possible to find elliptical shapes similar to the ones proposed in the present work and thus it is compared to the proposed implementation.

In this work, a vision-based external localization system based on a circular ring (roundel) pattern is proposed. An example of the pattern is depicted in Figure 1e. The algorithm allows to initiate the pattern search anywhere in the image without any performance penalty.

Therefore, the search is started from the point of the last known pattern position. Since the algorithm does not contain any phase that processes the entire image, successful tracking causes the method to process only the area occupied by the pattern. Therefore, the algorithm's computational complexity is independent of the image size. This provides a significant performance boost, which allows to track thousands of patterns in real-time using a standard PC. By performing an initial unattended calibrating step, where the reference frame is defined, pose computation of ground robots moving on a plane is performed with millimeter precision using an off-the-shelf camera.

The real-world performance of the proposed method makes it highly competitive with the aforementioned state-of-the-art methods. Moreover, its computational complexity is significantly lower, which makes the method superior for scenarios with embedded computational resources and real-time constraints. These findings are supported by the experimental results and a comparison with the selected localization methods presented in Section 6.

3 Pattern detection

The core of the proposed computationally efficient localization system is based on pattern detection. Fast and precise detection is achieved by exploiting properties of the considered pattern that is a black and white roundel consisting of two concentric annuli with a white central disc, see Figure 1.

The low computational requirements are met by the pattern detection procedure based on on-demand thresholding and flood fill techniques, and gathering statistical information of the pattern on the fly. The statistical data are used in consecutive tests with increasing complexity, which determine if a candidate area represents the desired circular pattern.

The pattern detection starts by searching for a black segment. Once such a segment is detected and passes the initial tests, the segment detection for a white inner disc is initiated at the expected pattern center.

Notice, that at the beginning, there is no prior information about the pattern position in the image; hence, the search for the black segment is started at a random position. Later, in the subsequent detections, when a prior pattern position is available, the algorithm starts detection over this area. For a successfully re-detected (tracked) pattern, the detection processes only pixels belonging to the pattern itself, which significantly reduces the computation burden. Since the method is robust (see following sections for detection limits), tracking is generally successful and thus the method provides very high computational performance.

After the roundel is detected, its image dimensions and coordinates are identified. Then, its three-dimensional

position with respect to the camera is computed from its known dimensions and camera re-projection techniques, and its coordinates are transformed to a coordinate frame defined by the user, see Section 4.

In this section, a detailed description of the pattern detection based on an efficient thresholding is presented together with an estimation of the pattern center and dimensions and a compensation of the incorrect diameter estimation, which has a positive influence to the localization precision. Moreover, a multiple pattern detection capability is described in Section 3.6.

3.1 Segmentation

The pattern detection is based on an image segmentation complemented with on-demand thresholding that searches for a contiguous set of black or white pixels using a flood-fill algorithm depicted in Algorithm 1. First, a black circular ring is searched for in the input image starting at an initial pixel position p_0 . The adaptive thresholding classifies the processed pixel using an adaptively set value τ as either black or white. If a black pixel is detected, the queue-based flood-fill algorithm procedure is initiated to determine the black segment. The queue represents the pixels of the segment and is simply implemented as a buffer with two pointers q_{start} and q_{end} .

Once the flood fill is complete, the segment is tested for a possible match of the outer (or inner) circle of the pattern. At this point, these tests consist of a minimum size (in terms of the number of pixels belonging to the segment) and a roundness measure within acceptable bounds. Notice, that during the flood-fill search, extremal pixel positions can be stored. This allows to establish the bounding box of the segment (b_u and b_v) at any time. Besides, after finding a segment, the queue contains positions of all the segment's pixels. Hence, initial simple constraints can be validated quickly for a fast rejection of false positives.

In the case where either test fails, the detection for further segments continues by starting from the next pixel position (i.e., a pixel at the position $p_0 + 1$). However, no redundant computation is performed since the previous segment is labeled with a unique identifier.

The first roundness test is based on the pattern's bounding box dimensions and number of pixels. Theoretically, the number of pixels s of an elliptic ring with outer and inner diameters d_o, d_i and dimensions b_u, b_v should be

$$s = \frac{\pi}{4} b_u b_v \frac{d_o^2 - d_i^2}{d_o^2}. \quad (1)$$

Therefore, the tested segment dimensions and area should satisfy the inequality

$$\rho_{tol} > \left| b_u b_v \frac{\pi}{4} \frac{\rho_{exp}}{s} - 1 \right|, \quad (2)$$

Algorithm 1: Flood-fill segmentation

Input: $(p, \rho_{exp}, class)$: p – starting pixel position; ρ_{exp} – expected area to bounding box dimensions ratio; $class$ – searched segment type (white or black)

Output: $(u, v, b_u, b_v, \mu, valid)$: (u, v) – segment center; (b_u, b_v) – bounding box; μ – average brightness; $valid$ – validity

```

 $s_{id} \leftarrow s_{id} + 1$  // increment segment ID
 $q_{old} \leftarrow q_{end}$  // store previous queue end
 $pixel\_class[p] \leftarrow s_{id}$  // mark pixel as processed and
 $queue[q_{end} + +] \leftarrow p$  // push its position to the queue
// #1 perform the flood fill search
while  $q_{end} > q_{start}$  do
   $q \leftarrow queue[q_{start} + +]$  // pull pixel from the queue
  // and check its neighbours
  foreach  $offset \in \{+1, -1, +w, -w\}$  do
     $r \leftarrow q + offset$ 
    if  $pixel\_class[r] = unknown$  then
       $pixel\_class[r] \leftarrow classify(Image[r], \tau)$ 
    if  $pixel\_class[r] = class$  then
       $queue[q_{end} + +] \leftarrow r$ 
       $pixel\_class[r] \leftarrow s_{id}$ 
      update  $u_{min}, u_{max}, v_{min}, v_{max}$  from  $r_u, r_v$ 
  valid  $\leftarrow false$ 
// #2 test for the pattern size and roundness
 $s \leftarrow q_{end} - q_{old}$ 
if  $s > min\_size$  then
   $u \leftarrow (u_{max} + u_{min})/2$  // segment center x-axis
   $v \leftarrow (v_{max} + v_{min})/2$  // segment center y-axis
   $b_u \leftarrow (u_{max} - u_{min}) + 1$  // estimate segment width
   $b_v \leftarrow (v_{max} - v_{min}) + 1$  // estimate segment height
   $\rho \leftarrow \rho_{exp} \pi b_u b_v / (4s) - 1$  // calculate roundness
  if  $-\rho_{tol} < \rho < \rho_{tol}$  then
     $\mu \leftarrow \frac{1}{s} \sum_{j=q_{old}}^{q_{end}-1} Image[j]$  // mean brightness
    valid  $\leftarrow true$  // mark segment as valid

```

where ρ_{exp} equals 1 for white and $1 - d_i^2/d_o^2$ for black segments. The value of ρ_{tol} represents a tolerance range, which depends on the camera radial distortion and possible pattern deformation and spatial orientation.

If a black segment passes the roundness test, the second flood-fill search for the inner white segment is initiated from the position corresponding to the segment centroid. If the inner segment passes the minimum size and roundness tests, further validation tests are performed. These involve the concentricity of both segments, their area ratio, and a more sensitive circularity measure (discussed in the following sections). If the segments pass all these complex tests, the pattern is considered to be found and its centroid position will be used as a starting point p_0 for the next detection run. The overall pattern detection algorithm is depicted in Algorithm 2.

3.2 Efficient thresholding

Since the segmentation looks only for black or white segments, the success rate of the roundel detection depends

Algorithm 2: Pattern detection

Input: $(p_0, \tau, Image)$: p_0 – position to start search; τ – threshold; $Image$ being processed

Output: (c, p_0, τ) : c – the pattern data; p_0 – position to start the next search; τ – an updated threshold

```

 $s_{id} \leftarrow 0; i \leftarrow p_0$  // initialize
// #1 search throughout the image
repeat
  if  $pixel\_class[i] = unknown$  then
    if  $classify(Image[i], \tau) = black$  then
       $pixel\_class[i] \leftarrow black$ 
  // initiate pattern search
  if  $pixel\_class[i] = black$  then
    // search for outer ring
     $q_{end} \leftarrow q_{start} \leftarrow 0$ 
     $C_{outer} \leftarrow flood\_fill\_seg(i, \rho_{outer}, black)$ 
    if  $valid(C_{outer})$  then
      // search for inner ellipse
       $j \leftarrow center(C_{outer})$ 
       $C_{inner} \leftarrow flood\_fill\_seg(j, \rho_{inner}, white)$ 
      if  $valid(C_{inner})$  then
        // test area ratio (no. of pixels):
         $s_{outer} = |C_{outer}|, s_{inner} = |C_{inner}|$ 
        if  $\frac{s_{outer}}{s_{inner}} \approx \frac{d_o^2 - d_i^2}{d_i^2}$  then
          check segments for concentricity
          compute ellipse semiaxes  $e_0, e_1$ 
          if  $q_{end} \approx \pi |e_0 e_1|$  then
            assign segment ID or
            compensate illumination
            mark segment as valid
            break
       $i \leftarrow (i + 1) \bmod sizeof(Image)$  // go to next pixel
until  $i \neq p_0$ ;
// #2 set the thresholding value
if  $valid(C_{inner})$  then
   $\tau \leftarrow (\mu_{outer} + \mu_{inner})/2$ 
  // hide pattern in multiple pattern detection
  paint over all inner ellipse pixels as black;
else
   $\tau \leftarrow binary\ search\ sequence$ 
// #3 perform the cleanup
if only two segments examined then
  reset  $pixel\_class[]$  inside bounding box of  $C_{outer}$ 
else
  reset entire  $pixel\_class[]$ 

```

on the threshold parameter τ , especially under various lighting conditions. Therefore, we proposed to adaptively update τ whenever the detection fails according to a binary search scheme over the range of possible values. This technique sets the threshold τ consecutively to values $\{1/2, 1/4, 3/4, 1/8, 3/8, 5/8 \dots\}$ up to a pre-defined granularity level, when τ is reset to the initial value.

When the pattern is successfully detected, the threshold is updated using the information obtained during detection in order to iteratively improve the precision of

segmentation:

$$\tau = \frac{\mu_{outer} + \mu_{inner}}{2}, \quad (3)$$

where μ_{outer}, μ_{inner} correspond to the mean brightness value of the outer and inner segments, respectively.

The computationally intensive full image thresholding is addressed by on-demand processing over each pixel analyzed during the detection. At the very first access, the RGB values of the image are read and a pixel is classified as either black or white and the classification result is stored for further re-use in the subsequent steps. Moreover, whenever the tracking is successful, only the relevant pixels are thresholded and processed by the two-step flood fill segmentation. Clearing the per-pixel classification memory area is also efficiently performed by only resetting the values inside the pattern's bounding box. As a result, the detection step is not directly dependent on the input image resolution, which provides a significant performance gain. If the tracking is not successful, extra memory accesses resulting from this on-demand strategy are negligible compared to a full-image thresholding approach.

3.3 Pattern center and dimensions

After the black and white segments pass all the initial tests, a more sophisticated roundel validation is performed. The validation is based on a more precise roundness test using estimation of the ellipse (pattern) semi-axes. All the information to calculate the ellipse center u, v and the semi-axes $\mathbf{e}_0, \mathbf{e}_1$ is at the hand, because all the pattern pixels are stored in the flood-fill queue. Hence, the center is calculated as the mean of the pixel positions. After that, the covariance matrix \mathbf{C} , eigenvalues λ_0, λ_1 , and eigenvectors $\mathbf{v}_0, \mathbf{v}_1$ are established. Since the matrix \mathbf{C} is two-dimensional, its eigen decomposition is a matter of solving a quadratic equation. The ellipse semi-axes e_0, e_1 are calculated simply by

$$\mathbf{e}_i = 2\sqrt{\lambda_i}\mathbf{v}_i. \quad (4)$$

The final test verifying the pattern roundness is performed by checking if the inequality

$$\rho_{prec} > \left| \pi \frac{|\mathbf{e}_0||\mathbf{e}_1|}{s} - 1 \right| \quad (5)$$

holds, where s is the pattern size in the number of pixels. Unlike in the previous roundness test (2), the tolerance value of ρ_{prec} can be much lower because (4) establishes the ellipse dimensions with subpixel precision.

Here, it is worth mentioning that if the system runs on embedded hardware, it might be desirable to calculate \mathbf{C} using integer arithmetic only. However, the integer

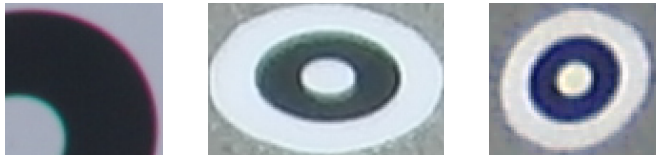


Fig. 2: Undesired effects affecting the pattern edge.

arithmetic might result in a loss of precision, therefore \mathbf{C} should be calculated as

$$\mathbf{C} = \frac{1}{s} \sum_{i=0}^{s-1} \begin{pmatrix} u_i u_i & u_i v_i \\ u_i v_i & v_i v_i \end{pmatrix} - \begin{pmatrix} uu & uv \\ uv & vv \end{pmatrix}, \quad (6)$$

where u_i and v_i are the pattern's pixel coordinates stored in the queue and u, v denote the determined pattern center.

3.4 Pattern identification

The ratio of the patterns' inner and outer diameters does not have to be a fixed value, but can vary between the individual patterns. Therefore, the variable diameter ratio can be used to distinguish between individual circular patterns. If this functionality is required, the system user can print patterns with various diameter ratios and use these ratios as ID's.

However, this functionality requires to relax the tolerance ranges for the tests of inner/outer segment area ratio, which might (in an extreme case) cause false positive detections. Variable inner circle dimensions also might mean a smaller inner circle or a thinner outer ring, which might decrease the maximal distance at which the pattern is detected reliably. Moreover, missing a priori knowledge of the pattern's diameter ratio means that compensation for incorrect diameter estimation is not possible, which might slightly decrease the method's precision.

3.5 Compensation of incorrect diameter estimation

The threshold separating black and white pixels has a significant impact on the estimation of the pattern dimensions. Moreover, the pixels on the black/white border are affected by chromatic aberration, nonlinear camera sensitivity, quantization noise, and image compression, see Figure 2. As a result, the borderline between the black ring and its white background contains a significant number of misclassified pixels.

The effect of pixel misclassification is observed as an increase of the ratio of white to black pixels with increasing pattern distance. The effect causes the black ring to appear thinner (and smaller), which has a negative impact on the distance estimation. However, the inner and outer diameters of the pattern are known, and therefore, the knowledge of the true d_o and d_i can be used

to compensate for the aforementioned effect. First, we can establish the dimensions of the inner white ellipse e_{0i} and e_{1i} in the same way as in Section 3.3. We assume the pixel misclassification enlarges the inner ellipse semi-axes $\mathbf{e}_{0i}, \mathbf{e}_{1i}$ and shrinks the outer semi-axes $\mathbf{e}_{0o}, \mathbf{e}_{1o}$ by a value of t . Since the real inner d_i and outer d_o pattern diameters are known, the true ratio of the areas can be expressed as

$$\frac{d_i^2}{d_o^2} = r = \frac{(e_{0i} - t)(e_{1i} - t)}{(e_{0o} + t)(e_{1o} + t)}, \quad (7)$$

where t can be calculated as a solution of the quadratic equation

$$(1 - r)t^2 - t(e_{0i}e_{1i} + re_{0o}e_{1o}) + e_{0i}e_{1i} - re_{0o}e_{1o} = 0. \quad (8)$$

The ambiguity of the solution can be resolved simply by taking into account that the corrected semi-axes lengths $e_{0i} - t, e_{1i} - t$ must be positive. The compensation of the pattern diameter reduces the average localization error by approximately 15 %.

3.6 Multiple target detection

The described roundel detection method can also be used to detect and track several targets in the scene. However, a single threshold τ is not well suited to detecting more patterns because of illumination variances. Besides, other differences presented across the working area may affect the reflectance of the pattern and thus result in different gray levels for different patterns, which in turn requires a different τ value for each pattern. Individual thresholding values not only provide detection robustness but also increase precision by optimizing pixel classification for each target individually.

Multiple targets can be simply detected in a sequence one by one, and the only requirement is to avoid detection of the already detected pattern. This can be easily avoided by modifying the input image after a successful detection by painting over the corresponding pixels, i.e., effectively masking out the pattern for subsequent detection runs.

Detection of multiple targets can also be considered in parallel, e.g., for obtaining additional performance gain, using multi core processor. In this case, it is necessary to avoid a possible race condition and mutual exclusion has to be used for accessing the classification result storage.

An initial implementation of the parallel approach using OpenMP and multi-processor system did not yield a significant speedup. Furthermore, due to the high performance of detection of a single pattern, the serial implementation provides better performance than the parallel approach. Therefore, all the presented computational results in this paper are for the serial implementation.

4 Pattern localization

The relative pattern position to the camera module is calculated from the parameters established in the previous step. We assume that the radial distortion of the camera is not extreme and the camera intrinsic parameters can be established by the method [21] or similar. With this assumption, the pattern's position is computed as follows:

1. The ellipse center and semi-axes are calculated from the covariance matrix eigenvectors and transformed to a canonical camera coordinate system.
2. The transformed parameters are then used to establish coefficients of the ellipse characteristic equation, which is a bilinear form matrix (also called a cubic).
3. The pattern's spatial orientation and position within the camera coordinate frame is then obtained by means of eigen analysis of the cubic.
4. The relative coordinates are transformed to a two- or three-dimensional coordinate frame defined by the user.

A detailed description of the pattern position estimation is presented in the following sections.

4.1 Ellipse vertices in the canonical camera system

The ellipse center u'_c, v'_c and semi-axes $\mathbf{e}'_0, \mathbf{e}'_1$ are established in a canonical camera form. The used canonical form is a pinhole camera model with unit focal lengths and no radial distortion. The transformation to a canonical camera system is basically a transform inverse to the model of the actual camera.

First, we calculate the image coordinates of the ellipse vertices $\mathbf{a}_{0,1}$ and co-vertices $\mathbf{b}_{0,1}$, and transform them to the canonical camera coordinates $\mathbf{a}'_{0,1}, \mathbf{b}'_{0,1}$. The canonical coordinates of the (co)vertices are then used to establish the canonical center and canonical semi-axes. This rather complicated step is performed to compensate for the radial distortion of the image at the position of the detected ellipse.

Since the ellipse center \mathbf{u} and semi-axes $\mathbf{e}_0, \mathbf{e}_1$ are known, calculation of the canonical vertices $\mathbf{a}'_{0,1}$ and co-vertices $\mathbf{b}'_{0,1}$ is done simply by adding the semi-axes to the ellipse center and transforming them:

$$\begin{aligned} \mathbf{a}'_{0,1} &= g'((u \pm e_{0x} - c_x)/f_x, (v \pm e_{0y} - c_y)/f_y) \\ \mathbf{b}'_{0,1} &= g'((u \pm e_{1x} - c_x)/f_x, (v \pm e_{1y} - c_y)/f_y), \end{aligned}$$

where g' is the radial undistortion function and $f_{x,y}, c_{x,y}$ are the camera focal lengths and optical center, respectively. Using the canonical position of the ellipse vertices, the ellipse center u', v' and axes $\mathbf{e}'_0, \mathbf{e}'_1$ are then calculated as

$$\begin{aligned} \mathbf{e}'_0 &= (\mathbf{a}'_0 - \mathbf{a}'_1)/2 \\ \mathbf{e}'_1 &= (\mathbf{b}'_0 - \mathbf{b}'_1)/2 \\ \mathbf{u}'_c &= (\mathbf{a}'_0 + \mathbf{a}'_1 + \mathbf{b}'_0 + \mathbf{b}'_1)/4 \end{aligned}.$$

After this step, we have all essential variables to calculate the ellipse characteristic equation.

4.2 Ellipse characteristic equation

Notice that each point u, v lying on an ellipse satisfies the characteristic equation of an ellipse:

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}^T \begin{pmatrix} q_a & q_b & q_d \\ q_b & q_c & q_e \\ q_d & q_e & q_f \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \mathbf{X}^T \mathbf{Q} \mathbf{X} = 0, \quad (9)$$

where \mathbf{Q} is called a conic. Thus, the parameters of the matrix \mathbf{Q} are calculated from the ellipse center and axes as follows:

$$\begin{aligned} q_a &= +e'_{0u}e'_{0u}/|\mathbf{e}'_0|^2 + e'_{0v}e'_{0v}/|\mathbf{e}'_1|^2 \\ q_b &= +e'_{0u}e'_{0v}/|\mathbf{e}'_0|^2 - e'_{0u}e'_{0v}/|\mathbf{e}'_1|^2 \\ q_c &= +e'_{0u}e'_{0u}/|\mathbf{e}'_1|^2 + e'_{0v}e'_{0v}/|\mathbf{e}'_0|^2 \\ q_d &= -u'_c q_a - v'_c q_b \\ q_e &= -u'_c q_b - v'_c q_c \\ q_f &= +q_a u'^2_c + q_c v'^2_c + 2q_b u'_c v'_c - 1 \end{aligned} \quad (10)$$

4.3 Pattern position

Once the conic parameters \mathbf{Q} are known, the position and orientation of the pattern can be obtained by means of eigenvalue analysis [22]. Let the \mathbf{Q} matrix eigenvalues and eigenvectors be $\lambda_0, \lambda_1, \lambda_2$ and $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2$, respectively. Since \mathbf{Q} represents an ellipse, its signature is $(2, 1)$ and we assume that $\lambda_0 \geq \lambda_1 > 0 > \lambda_2$. According to [16], the position of the circle can be calculated as:

$$\mathbf{x}_c = \pm \frac{d_o}{\sqrt{-\lambda_0 \lambda_2}} \left(\mathbf{q}_0 \lambda_2 \sqrt{\frac{\lambda_0 - \lambda_1}{\lambda_0 - \lambda_2}} + \mathbf{q}_2 \lambda_0 \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_0 - \lambda_2}} \right),$$

where d_o is the circular pattern diameter. The ambiguity of the sign can be resolved by taking into account that the pattern is located within the camera field of view. Thus, if the first component of the \mathbf{x}_c vector is negative, the vector \mathbf{x} is simply inverted.

4.4 Transformation to the global coordinates

The position \mathbf{x}_c of the circular pattern established in the previous step is in a camera centered coordinate frame. Depending on the particular application scenario, our system allows to transform the pattern coordinates to a 3D or 2D coordinate frame defined by the user. The user just places four circular patterns in the space covered by the camera and provides the system with their real positions.

4.4.1 Global coordinate frame – 3D case

In the case of the 3D localization, the three patterns at positions $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ define the coordinate origin and x and y axes, respectively. The transformation between the global $\mathbf{x} = (x, y, z)^T$ and camera centered $\mathbf{x}_c = (x_c, y_c, z_c)^T$ coordinate systems can be represented as

$$\mathbf{x} = \mathbf{T}(\mathbf{x}_c - \mathbf{t}_0),$$

where \mathbf{t}_0 equals \mathbf{x}_0 and \mathbf{T} is a similarity transformation matrix.

The user can define the coordinate system simply by putting three “calibration” patterns in the camera field of view and designating the pattern that defines the coordinate system origin \mathbf{t}_0 and the x and y axes. Using the pattern positions (let us define them as $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$, respectively), the system calculates the transformation between the camera and global coordinate systems, i.e., the vector \mathbf{t}_0 and matrix \mathbf{T} . Establishing the translation vector \mathbf{t} is straightforward – it corresponds to the camera coordinates of the pattern at the global coordinate origin, i.e., $\mathbf{t} = \mathbf{x}_0$.

The x and y axes of the coordinate frame are defined by vectors $\mathbf{t}_1 = \mathbf{x}_1 - \mathbf{x}_0$ and $\mathbf{t}_2 = \mathbf{x}_2 - \mathbf{x}_0$, respectively. Since we assume an orthonormal coordinate system, the z axis vector can be simply calculated as a cross product $\mathbf{t}_3 = \mathbf{t}_1 \times \mathbf{t}_2$. From an algebraic point of view, the matrix \mathbf{T} represents a transformation of the vector $\mathbf{x}' = \mathbf{x} - \mathbf{t}$ to a coordinate system defined by the basis $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3$. Therefore, the matrix \mathbf{T} can be calculated simply as

$$\mathbf{T} = \begin{pmatrix} t_{1x} & t_{2x} & t_{3x} \\ t_{1y} & t_{2y} & t_{3y} \\ t_{1z} & t_{2z} & t_{3z} \end{pmatrix}^{-1}. \quad (11)$$

Having established the vector \mathbf{t} and matrix \mathbf{T} , any point in the camera coordinate frame can be transformed to the coordinate frame defined by the user.

When the user places four patterns in the camera field of view, four independent coordinate transformations are calculated using each pattern triplet. The pattern position x' is then calculated as their mean, which results in increased system accuracy.

4.4.2 Global coordinate frame – 2D case

Two-dimensional localization can be generally more precise than full three-dimensional localization. This is because the estimation of the pattern position depends mainly on the pattern distance, especially in cases when the pattern image is small. Estimation of the pattern distance can be simply avoided if all the patterns are located only in a plane, e.g., ground robots operating on a floor.

In this case, the transformation from the image coordinates to an arbitrary world plane is a homography, and (homogeneous) spatial coordinates \mathbf{x} of the patterns can

be calculated directly from their canonical coordinates \mathbf{u}' simply by $\mathbf{x} = \mathbf{H}\mathbf{u}'$, where \mathbf{H} is a 3×3 homography matrix. Similarly to the case of three-dimensional localization, the user can define \mathbf{H} just by placing four patterns in the camera field of view and providing the system with their positions in the desired coordinate frame.

5 Sensor Model

In this section, we present three mathematical models that can be used to estimate the expected performance of the system. The main purpose of these models is to support the selection of the most suitable camera, processing hardware, and pattern size according to the particular application scenario. The first model calculates the localization system coverage from the pattern dimensions, camera resolution, and field of view. The second set of equations provides estimation of the localization precision based on the camera parameters, pattern dimensions, and required coverage. Finally, the third model estimates the necessary computational power to track the given number of patterns at the desired frame rate.

5.1 Localization system coverage

Regarding the practical deployment of the localization system, its most critical property is its coverage or “operational space”, i.e., the space where the pattern is reliably detected and localized. The dimensions of the operational space are affected by the camera focal length and radial distortion, image resolution, pattern diameter, and pattern spatial orientation.

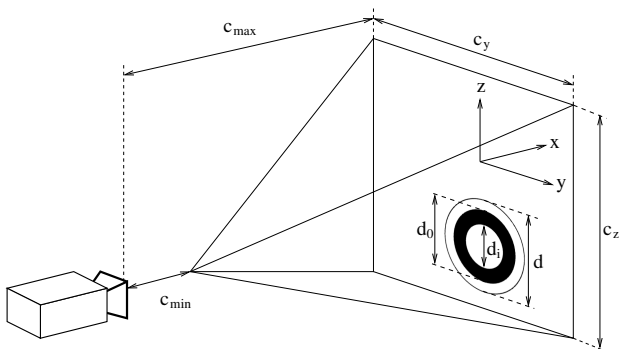


Fig. 3: Geometry of the operational space.

For the sake of simplicity, the effect of radial distortion on the shape of the operational space is neglected and an ideal pinhole camera is assumed. Considering this ideal model, the operational space has a pyramidal shape with its apex close to the camera, see Figure 3.

The parameters of the operational space are the minimal and maximal detectable distances v_{min}, v_{max} and base dimensions v_y, v_z .

A pattern can be detected if it “fits” in the image and its central part and black ring are recognizable. Therefore, the pattern image dimensions must be lower than the camera resolution, but higher than a certain value. To estimate the dimensions, we assume the camera focal lengths f_x, f_y and radial distortion parameters have been established by a calibration tool¹, e.g., MATLAB calibration toolbox or similar software based on [21]. Then, the width and height w_p, h_p of the pattern in pixels can be calculated by

$$w_p = f_x \frac{d_o}{x} \cos(\varphi), \quad h_p = f_y \frac{d_o}{x} \cos(\psi), \quad (12)$$

where x is the pattern distance from the image plane, d_o is the pattern diameter, and φ and ψ represent the pattern tilt.

5.1.1 Minimal localization distance

The minimal distance v_{min} , at which the pattern can be detected regardless of its orientation, is given as

$$v_{min} = d_o \max\left(\frac{f_x}{w}, \frac{f_y}{h}\right), \quad (13)$$

where w and h is the image horizontal and vertical resolution in pixels, respectively. One has to realize that the fractions f_x/w and f_y/h correspond to the camera field of view. Hence, the camera field of view remains the same regardless of the current resolution settings and the distance v_{min} can be considered as independent of the camera resolution.

5.1.2 Maximal localization distance

The pattern has to be formed from a sufficient number of pixels to be detected reliably. Therefore, the pattern pixel dimensions have to exceed a certain value that we define as D . The value of D has been experimentally established as 12. We also found that D might be lower than this threshold for exceptionally good lighting conditions; however, $D = 12$ represents a conservative value. Having D , the maximal detectable distance v'_{max} of the pattern can be calculated as

$$v'_{max} = \frac{d_o}{D} \min(f_x \cos(\varphi), f_y \cos(\psi)). \quad (14)$$

Notice a higher camera resolution increases the focal lengths f_x and f_y ; so, setting the camera resolution as high as possible maximizes the area covered by the localization system.

On the other hand, (14) does not take into account the camera radial distortion and it is applicable only

¹ Such a tool is also a part of the proposed system available online at [23].

when the pattern is located near the optical axis. The radial distortion causes the objects to appear smaller as they get far away from the optical axis. Thus, the distance v'_{max} at which the pattern is detected along the optical axis is higher than the maximal detectable distance v_{max} of the pattern located at the image corners. Therefore, the dimension v_{max} of the operational space is smaller than v'_{max} by a certain factor and v_{max} can be calculated as

$$v_{max} = \frac{d_o}{D} \min(k_x f_x \cos(\varphi), k_y f_y \cos(\psi)), \quad (15)$$

where k_x and k_y represent the effect of the radial distortion. The values of k_x and k_y can be estimated from the differential of the radial distortion function close to an image corner:

$$\begin{aligned} k_x &= 1 + \frac{dg(r_x, r_y)}{dx} = 1 + 2k_1 r_x + 4k_2(r_x^3 + r_x r_y^2) + \dots \\ k_y &= 1 + \frac{dg(r_x, r_y)}{dy} = 1 + 2k_1 r_y + 4k_2(r_y^3 + r_y r_x^2) + \dots \end{aligned}$$

where r_x and r_y can be obtained from the camera optical axis and focal lengths as c_x/f_x and c_y/f_y . For a consumer grade camera, one can assume that the radial distortion would not shrink the pattern more than by 10 %; so, a typical value of $k_{x,y}$ would be between 0.9 and 1.0.

5.1.3 Base dimensions

Knowing the maximal detectable distance v_{max} , the dimensions of the localization area “base” v_y and v_z can be calculated as

$$v_y = w \frac{v_{max}}{f_x} - 2d_o, \quad v_z = h \frac{v_{max}}{f_y} - 2d_o, \quad (16)$$

where w and h are the horizontal and vertical resolutions of the camera used, respectively. Considering a typical pattern, the value of d_o is much smaller than the localization area and can be omitted.

With Equations (13), (15), and (16) the user can calculate the diameter of the pattern and camera parameters from the desired coverage of the system. It should be noted that the presented model considers a static configuration of the module and the detected pattern. Rapid changes of the pattern’s relative position may cause image blur, which might affect v_{max} and restrict the operational space.

5.2 Localization system precision

Another important property of the localization system is the precision with which the system provides estimation of the pattern position. The precision of the localization is directly influenced by the amount of noise in the image and uncertainty in the camera parameters. The position

estimation error also depends on the system operational mode, i.e., it is different for the three-dimensional and two-dimensional position estimations. The expected localization precision is discussed in the following sections for both the 2D and 3D cases.

5.2.1 Two-dimensional localization by homography

For the 2D localization, the pattern position is estimated simply from its center image coordinates. In the case of an ideal pinhole camera, the calibration procedure described in Section 4.4 should establish the relation between the image and world planes. Therefore, the precision of the position estimation is affected mainly by the image radial distortion. Since the uncertainties of the radial distortion parameters are known from the camera calibration step, the error of radial distortion for x and y can be estimated from the differential of the radial distortion function

$$\begin{aligned} \eta_x &= x(\epsilon_1 r + \epsilon_2 r^2 + \epsilon_5 r^3 + 2\epsilon_3 y) + \epsilon_4(r + 2x^2) \\ \eta_y &= y(\epsilon_1 r + \epsilon_2 r^2 + \epsilon_5 r^3 + 2\epsilon_4 x) + \epsilon_3(r + 2y^2), \end{aligned} \quad (17)$$

where η_x and η_y are the position relative errors, k_i are camera distortion parameters, ϵ_i are their uncertainties, and $r = x^2 + y^2$. The overall relative error of the two-dimensional localization can be expressed as

$$\eta_{hom} = \eta_{rad} = \sqrt{\eta_x^2 + \eta_y^2}. \quad (18)$$

Note that (17) does not take into account the camera resolution. Therefore, the model suggests that higher resolution cameras will not necessarily achieve better localization precision. This is further investigated in Section 6.2.2, where experimental results are presented. Also, note that in the standard camera calibration implementations, values of ϵ_i are meant as 99.7 % confidence intervals. To calculate the average error, i.e., the standard deviation, one has to divide η_{hom} by three.

5.2.2 Full three-dimensional localization

In the full 3D localization, the main source of the localization imprecision is incorrect estimation of the pattern distance. Since the pattern distance is inversely proportional to its diameter in pixels, smaller patterns will be localized with a higher error. The error in the diameter measurement is caused by quantization noise and by the uncertainty in the identification of the camera’s intrinsic parameters, especially in the parameters of the image radial distortion. One can roughly estimate the expected error in the pattern distance estimation as

$$\eta_{3D} = \frac{\Delta f}{f_x} + \Delta e_0 \frac{x f_x}{d_0} + \eta_{rad}, \quad (19)$$

where Δf is the error of the focal length estimation, Δe represents the error of the ellipse axis estimation due to

image noise, and η_{rad} is the relative error of the radial distortion model. While Δ_f and η_{rad} can be calculated from the camera calibration parameters, Δe_0 is influenced by a number of factors that include camera thermal noise, lighting, motion blur etc. However, its current value can be estimated on the fly from the variance of the calibration (see Section 4.4.1) patterns' diameters.

In our experiments, the typical value of Δe_0 was around 0.15 pixels. This means that for a well-calibrated camera, the major source of distance estimation error is the ratio of image noise to the pattern projection size. Since the pattern image size (in the number of pixels) grows with the camera resolution, the precision of localization can be increased simply by using a high resolution camera or a larger pattern.

5.3 Computational requirements

From a practical point of view, it is also desirable to estimate the necessary computational hardware needed to achieve a desired frame rate, especially for an embedded solution. The time needed to process one image can be roughly estimated from the number of patterns, their expected size, image dimensions, tracking failure rate, and the computer speed. For the sake of simplicity, we can assume that the time to process one frame is a linear function of the amount of processed pixels:

$$t = (k_0 + k_1(s_p(1 - \alpha) + s_i\alpha)) no, \quad (20)$$

where k_0 represents the number of operations needed per pattern regardless of its size (e.g., a coordinate transformation), k_1 is a constant corresponding to the number of operations per pixel per pattern, s_p is the average size of the pattern in pixels, α is the expected failure rate of the tracking, s_i is the image size in pixels, n is the number of tracked patterns, and o is the number of operations per second per processor core given as a ratio $o = c/m$ of the entire processor MIPS (Million Instructions Per Second) m and the number of processor cores c . The constant k_0 has been experimentally estimated as $5 \cdot 10^5$ and k_1 as 900. The average size s_p of a pattern can be calculated from the camera parameters, pattern diameter, and average distance from the camera by (12). Thus, if the user wants to track 50 patterns with 30 pixel diameter using a machine with two cores and 53 GIPS (Giga Instructions Per Second), the expected processing time per image would be 1.2ms, which would allow to process about 800 images per second.

The speed of the localization algorithm depends on the failure rate of the tracking α . Typically, if the pattern displacement between two frames is smaller than the pattern radius, the tracking mechanism causes the method to process only the pixels belonging to the pattern. This situation corresponds to α being equal to zero. Thus, assuming that the pattern is not moving erratically, the method's computational complexity is independent of

the processed image size. Moreover, the smaller the pattern image dimension, the faster the processing rate. Of course, equation (20) gives only a coarse estimate, but it might give the user a basic idea of the system processing speed. Equation (20) has been experimentally verified and the results are presented in Section 6.3.

6 Experiments

This section is dedicated to presentation of the experimental results verifying the mathematical models established in Section 5. First, the model of the operational space defined by (15) and (16) is tested to see if it corresponds to a real situation. After that, the real achievable precision of the localization is evaluated according to the model (17) and (19). Then, the real computational requirements of the algorithm are measured using different computational platforms and the model in (20) is validated. Finally, the performance of the proposed localization system is also evaluated according to the precise motion capture system and compared with the AR tag based approach ArUco [14] and the simple OpenCV circle detector.

6.1 Operational space for a reliable pattern detection

The purpose of this verification is to validate the model describing the area covered by the localization system. In Section 5.1, the covered space is described as a pyramid with base dimensions v_y, v_z and a height denoting the maximal detectable distance v_{max} .

6.1.1 Maximal detection distance

A key parameter of the operational space is the maximal distance for reliable pattern detection v_{max} that is described by (14). The following experimental setup has been used to verify the correctness of this model. Two different cameras have been placed on a mobile platform SCITOS-5 with precisely calibrated odometry. The proposed localization system was set up to track three circles, each with a different diameter. The platform was set to move away from the circles at a constant speed and its distance from the patterns was recorded whenever a particular pattern was not detected. The recorded distances are considered as the limit v'_{max} of the system operational space. The same procedure was repeated with the patterns being slanted by forty degrees.

During this experiment, the patterns were located approximately at the image center. As previously noted in Section 5.1.2, additional correction constants k_x, k_y have been introduced in Section 5.1.2 to take into account radial distortion effects, which cause the detected pattern to appear smaller when located at the image edges. The augmented model considering the radial distortion

Table 1: Maximal distance for a reliable pattern detection

Camera type	Pattern d_o [cm]	Distance [m]			
		Measured		Predicted	
		0°	40°	0°	40°
Logitech QC Pro	2.5	1.4	1.3	1.6	1.3
	5.0	3.3	2.8	3.2	2.5
Olympus VR-340	2.5	6.8	6.2	6.7	5.1
	5.0	13.2	11.4	13.4	10.3
	7.5	19.8	16.8	20.1	15.4

was verified in an additional experiment using a pattern with diameter 2.5 cm positioned at the image corner. In this case, the maximal detected distance was reduced by 7% for a Logitech QuickCam Pro camera and by 5% for an Olympus VR-340. These results are in a good accordance with the model introduced in Section 5.1.2, where the values of k_x and k_y were estimated to be between 0.9 and 1.0.

6.1.2 Base dimensions

The dimensions of the coverage base are modeled by (16), which provides the dimensions of the expected coverage v_y and v_z . This model was verified using a similar setup to the previous experiment. The camera was placed to face a wall at a distance established in the previous experiment and four patterns were placed at the very corners of the image. This procedure was repeated for three different sizes of the pattern. The operation space dimensions, both measured and calculated by (16), are summarized in Table 2.

Table 2: Dimensions of the operational space

d_o [cm]	Dimensions [m]					
	Measured			Predicted		
	v_{max}	v_y	v_z	v_{max}	v_y	v_z
2.5	1.6	2.1	1.6	1.5	1.9	1.4
5.0	3.0	3.9	3.0	3.0	4.0	3.0
7.5	4.5	5.9	4.5	4.4	5.8	4.4

6.2 Localization precision

The real localization precision, which is probably the most critical parameter of the localization system, was established experimentally using a dataset collected in

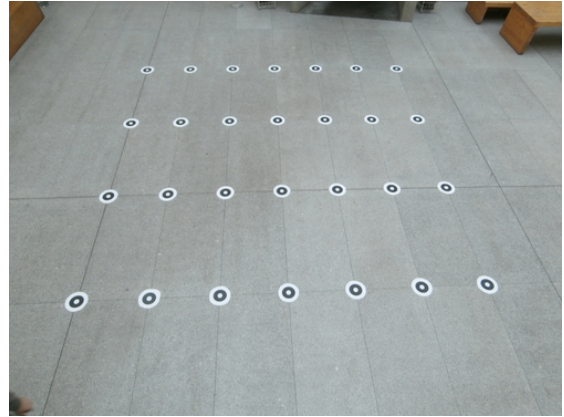


Fig. 4: Side view of the experiment

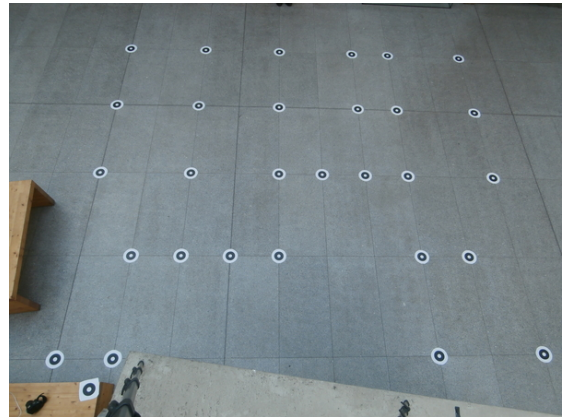


Fig. 5: Top view of the experiment

the main entrance hall of the Faculty of Mechanical Engineering at the Charles square campus of the Czech Technical University. The entrance hall offers enough space and its floor tiles form a regular rectangular grid with dimensions 0.625×1.250 m. The regularity of the grid was verified by manual measurements and the established precision of the tile placement is around 0.6 mm.

We placed several patterns on the tile intersections and took five pictures with three different cameras from two different viewpoints (see Fig. 4 and 5). The cameras used were a Creative Live! webcam, Olympus VR-340, and Canon 550D set to 1280×720 , 4608×3456 , and 5184×3456 pixel resolutions, respectively. The viewpoints were chosen at two different heights; so, the images of the scene were taken from a “side” and a “top” view.

First, three or four of the patterns in each image were used to define the coordinate system. Then, the resulting transformation was utilized to establish the circle global positions. Since the circles were placed on the tile corners, their real positions were known precisely. The Euclidean distances of these known positions to the ones

Table 3: Precision of 3D position estimation

Image		Abs. [cm]		Rel. [%]		
camera	view	ϵ_{avg}	ϵ_{max}	η_{pred}	η_{avg}	η_{max}
Webcam	side	5.7	19.5	1.04	0.90	2.96
Webcam	top	3.7	12.1	0.68	0.61	1.83
VR-340	side	1.9	6.5	0.47	0.35	1.02
VR-340	top	3.2	11.0	0.54	0.50	1.39
C-550D	top	2.5	7.4	0.30	0.43	1.46

Table 4: Precision of 2D position estimation

Image		Abs. [cm]		Rel. [%]		
camera	view	ϵ_{avg}	ϵ_{max}	η_{pred}	η_{avg}	η_{max}
Webcam	side	0.23	0.62	0.03	0.04	0.08
Webcam	top	0.18	0.68	0.04	0.03	0.09
VR-340	side	0.64	1.40	0.11	0.12	0.22
VR-340	top	0.68	2.08	0.19	0.11	0.32
C-550D	top	0.15	0.33	0.03	0.03	0.07

estimated by the system were considered as the measure of the localization error.

6.2.1 Three-dimensional localization precision

In this test, the system was set to perform full three-dimensional localization. In this model, the most significant cause of the localization error is the wrong distance estimation of the pattern (as noted in Section 5.2.2). The distance measurement is caused by an imperfect estimation of the pattern semiaxes lengths, see (19). The equation indicates that a camera with a higher resolution would provide a better precision.

The measured and predicted average and maximal localization errors for the individual pictures are shown in Table 3. The table also contains the predicted average localization error η_{pred} calculated by (3) for a comparison of the model and the real achieved precision.

6.2.2 Two-dimensional localization precision

In the case of indoor ground robot localization, we can assume that the robots move in a plane. The plane where the robots move and the image plane form a homography, which was previously defined by four reference patterns during the system setup. The real achievable precision of two-dimensional localization was measured within the same experimental scenario as the previous full 3D case. The average and maximal measured localization errors are depicted in Table 4. Similar to the previous case, the table contains the predicted mean error η_{pred} calculated by (17).

The results indicate that the assumption of ground plane movement increases the precision by an order of magnitude. Moreover, the results also confirm that increasing the image resolution does not necessarily increase the localization precision. Rather, the precision of localization is influenced mostly by the camera calibration imperfections. This fact confirms the assumptions presented in Section 5.2.1.

6.3 Computational requirements

The purpose of this experiment was to evaluate the estimation of the computational requirements provided by the model proposed in Section 5.3. Thus, the hypothesis is to test if the algorithm processing speed estimation (20) conforms to the proposed assumptions. Moreover, in this experiment, we also verify if the algorithm complexity depends only on the pattern size rather than on the image resolution.

6.3.1 Processing time vs. image and pattern dimensions

The model of computational requirements assumes that once the circles are reliably tracked, the system processing time is independent of the image size. In such a case, the image processing time is a linear function of the overall number of pixels belonging to all the patterns. Three synthetic datasets were created to verify this assumption. The first dataset consists of images with variable resolution and one circular pattern with a fixed size. The image resolutions of the second dataset are fixed, but the pattern diameter varies. Both pattern and image dimensions of the third dataset images are fixed; however, the number of patterns in each image ranges from one to four hundred. Each image of each dataset was processed one thousand times and the average time to track all the roundels in the image was calculated. The average processing time is shown in Figure 6.

The presented results clearly show that the image processing time is proportional to the number of pixels occupied by the tracked circular patterns and does not depend on the processed image dimensions. Moreover, the results demonstrate the scalability of the algorithm, which can track four hundred robots more than one hundred times per second. The aforementioned tests were performed on a single core of the Intel iCore5 CPU running at 2.5 GHz and accompanied with 8 GB of RAM.

6.3.2 Processing time using different platforms

From a practical point of view, processing images at a speed exceeding the camera frame rate is not necessary.

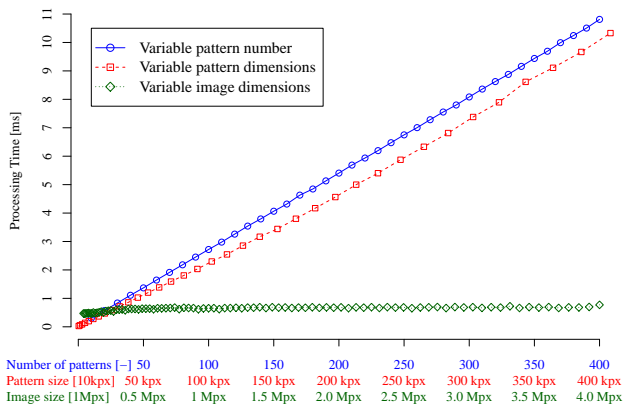


Fig. 6: Influence of the number of tracked patterns, pattern and image sizes on the method’s speed.

Rather, the algorithm might be deployed on systems with slower processing units. Thus, one should be able to establish what kind of computational hardware is needed for a particular setup. This can be roughly estimated using the time to process one image by means of (20). Three real world datasets and five different platforms, including two credit-card sized computers, were used to verify the model in a realistic setup.

- The “*small*” dataset consists of one thousand images of a static pattern, which occupies approximately seven hundred pixels, i.e., 0.1 % of the image’s total area.
- The “*large*” dataset is similar, but with a larger, sixty-pixel diameter pattern, occupying approximately 0.3 % of image pixels.

The algorithm performance with these two datasets (“*small*” and “*large*”) is relevant in scenarios where the tracked objects are moving slowly and the camera is in a static position.

- The “*fast*” dataset contains 130 images of a fast moving pattern with a variable size. The dataset was tailored to cause failure of the tracking mechanism in one case. Thus, the performance of the algorithm with this dataset is similar to cases when the camera is not stationary or the tracked objects are moving quickly.

The average processing time per image for each dataset was measured and calculated by (20). The results summarized in Table 5 indicate the correctness of the model described in Section 5.3.

6.4 Comparison with a precise localization system

The real achievable precision of the localization system has been reported in Section 6.2; however, only for experiments with static targets, where the patterns were placed at the predefined positions. Such a setup provides verification of the precision for scenarios where the system tracks slowly moving robots. On the other hand,

Table 5: Required image processing time

CPU	Processing time [ms]					
	Measured			Predicted		
Dataset:	small	large	fast	small	large	fast
i-5 2450M	0.04	0.10	0.37	0.04	0.12	0.35
Atom N270	0.30	0.72	3.25	0.33	0.89	2.68
Pentium M	0.20	0.45	1.44	0.17	0.48	1.45
Odroid U2	0.27	0.89	2.76	0.29	0.79	2.86
Raspb. Pi	1.10	4.00	15.8	1.34	3.66	11.0

Table 6: Localization accuracy of a moving target

Mode	Abs. [cm]		Rel. [%]	
	ϵ_{avg}	ϵ_{max}	η_{avg}	η_{max}
2D	1.2	4.2	0.4	1.5
3D	3.1	11.2	1.2	4.4

rapid movement of the tracked targets introduces additional effects, which might have a considerable impact on the system precision. First, the captured images can be affected by motion blur and deformation caused by the camera’s rolling shutter. Besides, there might be a delay in position estimation because standard USB cameras deliver the images with a delay caused by the interface’s limited bandwidth. Therefore, we consider an additional experiment to evaluate the impact of these factors on the real performance of the presented global localization system. We consider a precise reference system and set up our localization system in an area where a high-precision motion capture system is installed and which is able to track multiple targets². The motion capture system provides positions of the tracked targets 250 times per second with a precision up to 0.1 mm; so, it can be considered as a ground truth for our position measurements.

Four reference targets were placed in the area and a common coordinate system was calculated for both systems. After that, four sequences of targets moving at speeds up to 1.2 m/s were recorded by a Logitech Quick-CamPro and the commercial motion capture system. Euclidean distances of target positions provided by both systems were taken as a measure of our system accuracy. The mean precisions of two- and three-dimensional localization were established as 1.2 cm and 3.1 cm, respectively, see Table 6. Although the system’s relative accuracy is lower than in the static tests presented in Section 6.2, centimeter precision is still satisfactory for many scenarios. The error is caused mostly by the image blur because of a long exposure rate set by the camera

² Human Performance Centre at the University of Lincoln

Table 7: Localization precision comparison

mode	view	Relative error [%]					
		<i>WhyCon</i>		<i>ArUco</i>		<i>OpenCV</i>	
		<i>avg</i>	<i>max</i>	<i>avg</i>	<i>max</i>	<i>avg</i>	<i>max</i>
2D	side	0.12	0.19	0.20	0.41	0.52	1.03
	top	0.12	0.32	0.22	0.37	0.77	1.62
3D	side	0.31	1.10	0.63	2.52	-	-
	top	0.33	1.04	1.08	2.90	-	-

internal control. Careful setting of the camera exposure and gain parameters might suppress this effect. In fact, such a tuning has been made for localization of flying quadrotors, see Section 7.1.

It is also worth to mention that even though the commercial system is able to localize rapidly moving targets with a higher precision, its setup took more than thirty minutes while the presented system is prepared in a couple of minutes (just placing four patterns to establish the coordinate system).

6.5 Comparison with other visual localization systems

The advantages and drawbacks of the presented localization system are demonstrated by a comparison of its performance with the well-established localization approaches based on AR markers and OpenCV. The performance of AR-based markers has been measured using the ArUco [14] library for detection and localization of multiple AR markers (similar to the ones used in ARTag and ARToolKit systems). A comparison with the OpenCV circular pattern detection is based on the OpenCV’s “SimpleBlobDetector” class. The precision, speed, and coverage of all three systems was established in a similar way as described in the previous sections. For the sake of simplicity, we will refer to the presented system as *WhyCon*.

6.5.1 Precision comparison

The localization precision of the ArUco-, OpenCV-, and WhyCon-based localization methods was obtained experimentally by the method described in Section 6.2. The comparison was performed on 4608×3456 pixel pictures taken by an Olympus VR-340 Camera from two different (*side* and *top*) viewpoints.

The achieved results are presented in Table 7. The WhyCon position estimation error is significantly lower than the error of ArUco and OpenCV in both the two- and three-dimensional localization scenarios. Moreover, we found that the OpenCV’s blob radius calculation was too imprecise to reliably estimate the pattern distance and could not be used for the full 3D localization.

6.5.2 Performance comparison on different platforms

The computational performance of the three evaluated systems was compared for three different platforms. The methods’ performance was compared using two datasets similarly to the evaluation scenario described in Section 6.3.2. The *slow* dataset contains an easy-to-track pattern while for the *fast* datasets about 1 % of the images are tailored to cause a tracking failure.

Table 8: Image processing time comparison

CPU	Processing time [ms]						
	Dataset:	<i>ArUco</i>		<i>OpenCV</i>		<i>WhyCon</i>	
		fast	slow	fast	slow	fast	slow
i5 2450M	19	19	63	62	0.35	0.04	
Pentium M	121	119	329	329	1.00	0.18	
Odroid U2	148	149	371	366	0.93	0.28	
Raspb. Pi	875	875	1795	1759	6.59	1.21	

The results presented in Table 8 indicate that the proposed algorithm is capable of finding the patterns approximately one thousand times faster than the traditional methods. Even in the unfavorable case where the patterns cannot be reliably tracked, the method outperforms ArUco and OpenCV hundred times. The performance ratio is even better for small embedded platforms with limited computational power. This property is favorable for deployment in the intended applications, especially under real-time requirements.

6.5.3 Range and coverage comparison

The AR fiducial markers are primarily intended for augmented reality applications and in a typical scenario, the localized marker is situated close to the camera. Therefore, the AR marker-based systems are not tuned for a reliable detection of distant patterns with small image dimensions. Thus, the range and coverage of the AR marker-based systems would be lower compared to WhyCon. On the other hand, OpenCV’s circular blob detector can detect small circular patterns.

To estimate the ArUco and OpenCV detectors maximal range, we have established the minimal size (in pixels) that the tags need to have in order to be detected reliably. The sizes that correspond to the minimal pattern diameter D in the Equation (15) were established in a similar way as described in Section 6.1.1. While the OpenCV detector can find blobs larger than 12 pixels, the ArUco detector requires the AR marker side to be longer than 25 pixels. Therefore, ArUco’s maximal detection range is less than a half of WhyCon’s or OpenCV’s range.

7 Practical deployment

In this section, we present an overview of several research projects where the proposed circle detection algorithm has been successfully employed. This practical deployment demonstrates the versatility of the presented localization system. A short description of each project and comment about the localization performance is presented in the following sub-sections.

7.1 UAV formation stabilization

In this setup, the circle detection algorithm was considered for a relative localization and stabilization of UAV formations operating in both indoor and outdoor environments. A group of quadrotors are supposed to maintain a predefined formation by means of their relative localization. Each quadrotor UAV carries a circular pattern and an embedded module [24] running the localization method, see Figure 7.

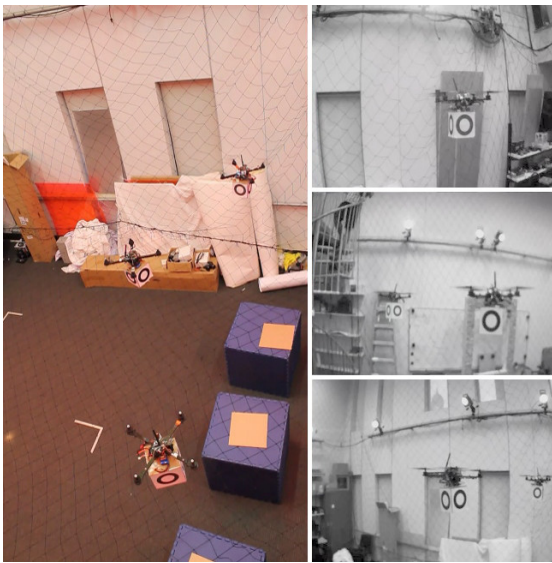


Fig. 7: Decentralized localization of quadrotor formation performed by the presented method. Courtesy of the GRASP laboratory, PENN.

Thus, each UAV is able to detect other quadrotors in its vicinity and maintain a predefined relative position. Although the UAV's movements are relatively fast, we did not observe significant problems caused by image blur and the system detected the patterns reliably. This scenario demonstrates the ability to reliably detect circular patterns despite their rapid movements and variable lighting conditions. Moreover, it proved its ability to satisfy real-time constraints when running on computationally constrained hardware. The precision of the relative localization was in the order of centimeters [24].

7.2 Birds-eye UAV-based localization system

The algorithm has also been used for relative localization of ground robots, which were supposed to maintain a predefined formation shape even if they lack direct visibility among each other. In this setup, one robot of the formation carried a heliport with the Parrot AR.Drone [25] quadrotor, which can take off and observe the formation from above using a downward-pointing camera. Each ground robot had a roundel pattern, which is elliptical rather than circular to provide also an estimate of the robot orientation. Using the roundel detection algorithm, the position and heading of the ground robots are provided by the flying quadrotor while it maintains its position above the formation.



Fig. 8: Mixed UAV-UGV robot formation.

Moreover, the heliport was designated by a circular pattern, which makes it possible to autonomously land the quadrotor after the mission end, see Figure 8. Despite the relatively low resolution (168×144) of the UAV's downward-looking camera and its rapid movements, the overall localization precision was approximately 5 cm.

7.3 Autonomous docking of modular robots

The Symbion and Replicator projects [26] investigate and develop novel principles of adaptation and evolution of symbiotic multi-robot organisms based on bio-inspired approaches and modern computing paradigms. The robot organisms consist of large-scale swarms of robots, which can dock with each other and symbiotically share energy and computational resources within a single artificial life form. When it is advantageous to do so, these swarm robots can dynamically aggregate into one or many symbiotic organisms and collectively interact with the physical world via a variety of sensors and actuators. The bio-inspired evolutionary paradigms combined with robot embodiment and swarm-emergent phenomena enable the organisms to autonomously manage their own hardware and software organization.

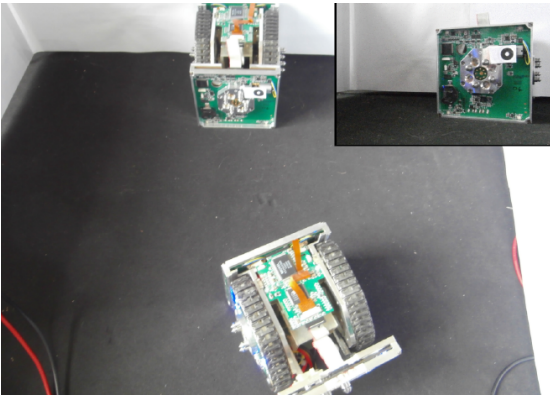


Fig. 9: Symbrion/Replicator robots during docking.

In these projects, the proposed localization algorithm has been used as one of the methods for detecting power sources and other robots, see Figure 9. The method demonstrated its ability to position the robot with a sub-millimeter precision, which is essential for a successful docking. The method's deployment in this scenario demonstrated not only its precision, but also its ability to run on computationally constrained hardware.

7.4 Educational robotics

SyRoTek [19] is a remotely accessible robotic laboratory, where users can perform experiments with robots using their Internet connectivity. The robots operate within a flat arena with reconfigurable obstacles and the system provides an overview of the arena from an overhead camera. The project has been used for education and research by several institutions in Europe and Americas. An important component of SyRoTek is the localization system providing estimation of the real robots' positions.

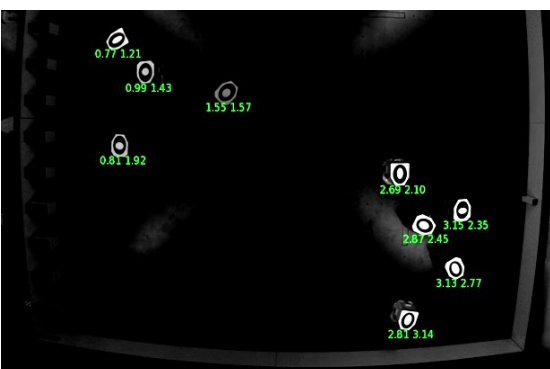


Fig. 10: A top-down view to the SyRoTek arena.

Originally the localization was based on a convolution algorithm. Even though it is computationally demanding

and rather imprecise, it demonstrated suitability for 24/7 operation. After replacement of this original localization system by the presented roundel-based system, the precision of the localization was improved. Moreover, the computational requirements were decreased as well [27]. In this deployment, the roundel pattern is formed from ellipses where the inner ellipse has slightly different dimensions, see Figure 10, which allows to distinguish between individual robots. This use case demonstrates the ability of the system to operate in 24/7 mode. In addition, using different dimensions of the inner ellipse allows to distinguish between 14 SyRoTek robots.

7.5 Ground truth assessment in mobile robot navigation

BearNav (originally SURFNav) is a visual based navigation system for both ground [28] and aerial mobile [29] robots. The method is based on convergence theorem [30], which states that map-based monocular navigation does not need full localization, because if the robot heading is continuously adjusted to turn the robot towards the desired path, its position error does not grow above certain limits even if the position estimation is based only on proprioceptive sensing affected by drift. The aforementioned principle allows to design reliable and computationally inexpensive camera-based navigation methods.

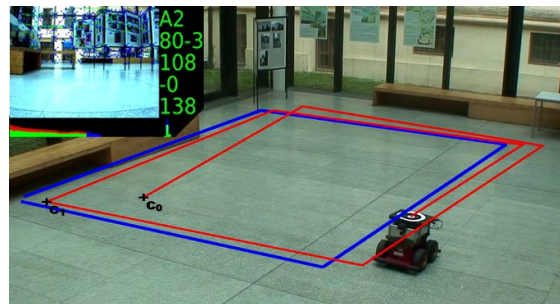


Fig. 11: Reconstructed trajectory of a mobile robot.

The presented roundel based localization system was used to provide a continuous and independent measurement of the robot position error, which allowed to verify the convergence theorem and benchmark the individual navigation algorithms in terms of their precision, see Figure 11. The system proved to be useful especially for aerial robots [29], which, unlike the ground robots, cannot be simply stopped for a manual position measurement.

7.6 Autonomous charging in long-term scenarios

The STRANDS project [31] aims to achieve intelligent robot behaviour in human environments through adaptation to, and the exploitation of, long-term experience.

The project approach is based on a deeper understanding of ongoing processes affecting the appearance and structure of the robot's environment. This will be achieved by extracting qualitative spatio-temporal knowledge from sensor data gathered during months of autonomous operation. Control mechanisms that will exploit these structures to yield adaptive behaviour in highly demanding scenarios will be developed.



Fig. 12: SCITOS-5 platform near its charging station. Notice the three o's of the label.

The circle detection method is used in the project as an initial solution of localization-related problems before more sophisticated implementations take its place. One of such deployments is localization of the robot during its approach to a charging station, which has been solved by placing three patterns in the charging area, see Figure 12.

8 Conclusion

We present a fast and precise vision-based system intended for multiple robot localization. The system's core component is based on a novel principle of circular roundel detection with computational complexity independent of the processed image size. The resulting system allows to localize swarms composed of several hundreds of robots with millimeter (2D) or centimeter (3D) precision, while keeping up with standard camera frame rates. In addition, we provide a model to calculate the sufficient camera and computer parameters to achieve the desired localization precision, coverage and update rate, which

support potential users to decide which kind of equipment is needed for their particular setup.

The most notable features of the system are its low computational requirements, ease of use, and the fact that it works with cheap, off-the-shelf equipment. The system has been deployed already in a number of international mobile robotic projects concerning distributed quad rotor localization [24], visual based autonomous navigation [30], decentralized formation control [25], long-term scenarios [31], evolutionary swarm [26], and educational [19] robotics. Since the system has already proved to be useful in a variety of applications, we publish its source code [23]; so, other roboteers can use it for their projects. The experiments indicate that the presented system is three orders of magnitude faster than traditional methods based on OpenCV or AR markers while being more precise and capable of detecting the markers at a greater distance.

In the future, we plan to increase the precision and coverage of the system by using multiple cameras. We will plan to improve the tracking success rate by predicting the position of the target by considering the dynamics of the tracked object.

References

1. Thrun, S., Burgard, W., Fox, D., et al.: Probabilistic robotics, vol. 1. MIT press Cambridge (2005)
2. Breitenmoser, A., Kneip, L., Siegwart, R.: A monocular vision-based system for 6D relative robot localization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 79–85 (2011)
3. Yamamoto, Y. et al.: Optical sensing for robot perception and localization. In: IEEE Workshop on Advanced Robotics and its Social Impacts, pp. 14–17. IEEE (2005)
4. Vicon: Vicon MX Systems. URL <http://www.vicon.com>. [cited 8 Jan 2014]
5. Mellinger, D., Michael, N., Kumar, V.: Trajectory generation and control for precise aggressive maneuvers with quadrotors. *International Journal of Robotics Research* **31**(5), 664–674 (2012)
6. Fiala, M.: 'ARTag', an improved marker system based on artoolkit (2004)
7. Wagner, D., Schmalstieg, D.: ARToolKitPlus for pose tracking on mobile devices. In: Proceedings of 12th Computer Vision Winter Workshop, pp. 139–146 (2007)
8. Kato, D.H.: ARToolKit. URL <http://www.hitl.washington.edu/artoolkit/>. [cited 8 Jan 2014]
9. Fiala, M.: Vision guided control of multiple robots. In: First Canadian Conference on Computer and Robot Vision, pp. 241–246 (2004)
10. Rekleitis, I., Meger, D., Dudek, G.: Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems* **54**(11) (2006)
11. Stump, E., Kumar, V., Grocholsky, B., Shiroma, P.M.: Control for Localization of Targets using Rangeonly Sensors. *International Journal of Robotics Research* (2009)
12. Fiala, M.: Comparing ARTag and ARtoolkit plus fiducial marker systems. In: Haptic Audio Visual Environments and their Applications, 2005, pp. 6–pp. IEEE (2005)
13. Bošnjak, M., Matko, D., Blažič, S.: Quadcopter hovering using position-estimation information from inertial sensors and a high-delay video system. *Journal of Intelligent & Robotic Systems* **67**(1), 43–60 (2012)

14. ArUco: a minimal library for augmented reality applications based on opencv. URL <http://www.uco.es/investiga/grupos/ava/node/26>. [cited 8 Jan 2014]
15. Ahn, S.J., Rauh, W., Recknagel, M.: Circular coded landmark for optical 3d-measurement and robot vision. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1128–1133. IEEE (1999)
16. Yang, S., Scherer, S., Zell, A.: An Onboard Monocular Vision System for Autonomous Takeoff, Hovering and Landing of a Micro Aerial Vehicle. *Journal of Intelligent & Robotic Systems* **69**(1-4), 499–515 (2012)
17. Lo, D., Mendonça, P.R., Hopper, A., et al.: TRIP: A low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing* **6**(3) (2002)
18. Pedre, S., Krajník, T., Todorovich, E., Borensztein, P.: Hardware/software co-design for real time embedded image processing: A case study. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 599–606. Springer (2012)
19. Kulich, M. et al: Syrotek - distance teaching of mobile robotics. *IEEE Trans. Education* **56**(1), 18–23 (2013)
20. Pedre, S., Krajník, T., Todorovich, E., Borensztein, P.: Accelerating embedded image processing for real time: a case study. *Jour. of Real-Time Image Processing* (2013)
21. Heikkila, J., Silven, O.: A four-step camera calibration procedure with implicit image correction. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1106–1112 (1997)
22. Yang, S., Scherer, S.A., Zell, A.: An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems* **69**(1-4), 499–515 (2013)
23. Krajník, T., Nitsche, M., Faigl, J.: The WhyCon system. URL <http://purl.org/robotics/whycon>. [cited 8 Jan 2014]
24. Faigl, J., Krajník, T., Chudoba, J., Přeučil, L., Saska, M.: Low-Cost Embedded System for Relative Localization in Robotic Swarms. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 985–990. IEEE, Piscataway (2013)
25. Saska, M., Krajník, T., Přeučil, L.: Cooperative Micro UAV-UGV Autonomous Indoor Surveillance. In: *International Multi-Conference on Systems, Signals and Devices*, p. 36. IEEE, Piscataway (2012)
26. Kernbach, S. et al.: Symbiotic robot organisms: Replicator and symbion projects. In: *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pp. 62–69. ACM (2008)
27. Cajtler, V.: Syrotek localization system. Bachelor thesis, Dept. of Cybernetics, CTU (2013). In Czech
28. Krajník, T., Přeučil, L.: A Simple Visual Navigation System with Convergence Property. In: *Proceedings European Robotics Symposium (EUROS)*, pp. 283–292 (2008)
29. Krajník, T., Nitsche, M., Pedre, S., Přeučil, L., Mejail, M.: A Simple Visual Navigation System for an UAV. In: *International Multi-Conference on Systems, Signals and Devices*, p. 34. IEEE, Piscataway (2012)
30. Krajník, T., Faigl, J., Vonásek, V., Košnar, K., Kulich, M., Přeučil, L.: Simple yet stable bearing-only navigation. *Journal of Field Robotics* **27**(5), 511–533 (2010). URL <http://dx.doi.org/10.1002/rob.20354>
31. Hawes, N.: STRANDS - Spatial-Temporal Representations and Activities for Cognitive Control in Long-Term Scenarios. URL <http://www.strands-project.eu>. [cited 8 Jan 2014]