Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Temporal Models for Mobile Robot Visual Navigation

Eliška Dvořáková

Supervisor: Ing. Tomáš Krajník, Ph.D.
Field of study: Open informatics
Subfield: Software systems
May 2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dvořáková** Jméno: **Eliška** Osobní číslo: **456989**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Temporální Modely pro Vizuální Navigaci Mobilních Robotů

Název bakalářské práce anglicky:

Temporal Models for Mobile Robot Visual Navigation

Pokyny pro vypracování:

The thesis aim is research of probabilistic models capable of representing naturally- occurring environment changes in the mobile robotics domain. These models will be used to predict visibility of salient environment elements (landmarks) used for teach-and-repeat visual navigation of mobile robots. The proposed model should be able to eciently represent long-term observations of multiple landmarks and predict their visibility for a particular time in the future. These predictions will be used to generate time-dependent maps used by a mobile robot for navigation. The impact of the models' predictive capabilities will be experimentally evaluated on a real robotic platform.

1. Get to know the principles of teach-and-repeat visual navigation used in mobile robotics [1, 2, 3]
2. Get to know the models of environment changes used in mobile robotics [1,4, 5, 6, 7, 8].
3. Get to know available visual navigation frameworks used in mobile robotics.
4. Select a set of perspective models and methods, combine them and perform their comparison on publicly available datasets.
5. Integrate the best performing methods into the Robot Operating Systém and test them on a real robot.

Seznam doporučené literatury:

[1] 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, vol. 32, no. 6, p. 1309{1332, 2016.
[2] T. Krajník, J. Faigl, V. Vonásek et al., \Simple, yet Stable Bearing-only Navigation,"" Journal of Field Robotics, 2010.
[3] P. Furgale and T. D. Barfoot, \Visual teach and repeat for long-range rover autonomy,"" Journal of Field Robotics, 2010.
[4] T. Krajník, P. Cristoforis, K. Kusumam, P. Neubert, and T. Duckett,\Image features for visual teach-and-repeat navigation in changing environments,""Robotics and Autonomous Systems, pp. 127{141, 2016.
[5] 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.
[6] D. Austin, L. Fletcher, and A. Zelinsky, \Mobile robotics in the long term- exploring the fourth dimension,"" in Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, vol. 2. IEEE, 2001, pp. 613{618.
[7] S. Lowry, N. Sunderhauf, 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.
[8] T. Krajník, J. P. Fentanes, J. Santos, and T. Duckett, \Fremen: Frequencymap enhancement for long-term mobile robot autonomy in changing environments,"" IEEE Trans. on Robotics, 2017.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Krajník, Ph.D., centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **19.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Tomáš Krajník, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání

Podpis studentky

Acknowledgements

I would foremost like to thank my supervisor Tomáš Krajník for his patient guidance through such a challenging project as bachelor thesis can be. I would also like to expres my thanks to my relatives and friends for their support and understanding why they have not seen me for some time. The work has been supported by the Czech Science Foundation project 17-27006Y and MŠMT project FR-8J18FR018.

Declaration

I hereby declare that I have completed this thesis independently and that I have used only the sources (literature, software, etc.) listed in the enclosed bibliography.

In Prague, 25th May 2018

Abstract

This thesis focuses on modelling environmental changes depending on the time for long-term mobile robot visual navigation and integrating these models into an existing functional navigation system. The goal of this thesis is to create a temporal environment model, which would allow to capture and predict operational environment changes providing long-term autonomous operating of a mobile robot in a changing environment. This thesis divides the problem of a temporal model creation into two sub problems: "How recorded changes should be interpreted?" and "How to predict current environment model usable for navigation?". This thesis extends a system that uses image features for visual navigation, but the abstraction of the solution allows using different methods instead. The system is implemented in the *Robotic Operating System* in C++ programming language.

Keywords: environment temporal model, long-term navigation

Supervisor: Ing. Tomáš Krajník, Ph.D.

Abstrakt

Tato práce se zabývá modelováním změn prostředí v čase pro dlouhodobou vizuální navigaci mobilních robotů a integrací těchto modelů prostředí do existujícího funkčního navigačního systému. Cílem této práce je vytvořit temporální model prostředí, který by umožnil postihout a předpovídat změny operativního prostředí a ummožnit tak dlouhodobé autonomní působení mobilního robota v měnícím se prostředí. Tato práce dělí problém tvorby temporálního modelu na dva hlavní podproblémy, které je možné řešit odděleně: „Jak interpretovat zaznamenané změny?“ a „Jak predikovat aktuální model prostředí použitelný pro dlouhodobou navigaci?“. Tato práce rozšiřuje systém, který používá pro navigaci vyznačné body v obraze, ale abstrakce řešení dovoluje použití jiných method pro visualní navigaci. Celý systém je implementován v systému „*Robotic Operating System*“ v programovacím jazyce C++.

Klíčová slova: Temporální model prostředí, dlouhodobá navigace

Překlad názvu: Temporální modely pro vizuální navigaci mobilních robotů

Contents

Figures

Tables

Chapter 1

Introduction

The autonomous robot navigation is a challenging problem that has been studied for over 30 years. Tremendous progress has been made since the very beginning. The research has already brought us success in the form of some vision-based autonomous robots (e.g. domestic robots, autonomous cars or Mars rovers) or range-sensor based autonomous robots (e. g. Google car).

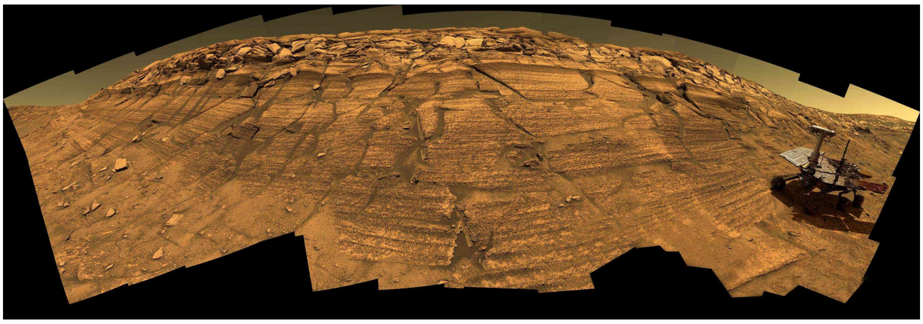


Figure 1.1: CGI rendering of Mars rover surroundings. On the right side is an example of an autonomous vehicle - Mars rover. Courtesy of [1].

There are three major problems of robot's autonomous movement: "Where am I?", "Where am I going?" and "How should I get there?" [8]. First two questions are associated with the estimation of robot's location and desired destination relatively within the environment. The third question desire the solution of motion and path planning.

The environment the robot operates in can be either structured or unstructured. The structured environment contains modification that improve navigation (e.g. roads, sidewalks, runways etc.). The unstructured environment is in its raw form without supporting modifications. The primary task of autonomous vehicles moving in structured environment is to recognize the road and stay in the right line or identify obstacles. The car follows road marks to guide itself. However lines fade away over time due to tear or disappear entirely due to changes in daylight. It brings us to the fact that there are two types of changes: in appearance and in structure. Appearance changes are caused by a variation or a lack of the light, so vehicles use lights to prevent lines disappearance. Structure changes are associated with object itself, no matter light or where the objects are placed and therefore the roads are maintained to prevent structure changes. Autonomous navigation is usually supported with additional sensors, which help determine the location (e.g. GPS, 2D scanner etc.) [9] [10].

Using a GPS localization in underground tunnels, on Mars or indoors is problematic, or the device is not equipped with a GPS tracker at all. Thus we focus on vision-based navigation only to achieve higher robustness. Rovers are able to travel tens of kilometres autonomously without a GPS tracker in an environment similar to the one on Mars or the Moon [11]. Although the surroundings of Mars rovers differ from the surroundings we can see outdoors in nature on Earth, there are changes again due to illumination and structure. Structure changes are observed during seasons. The environment looks different in summer and winter especially on Earth, where the plants bloom, the leaves fall from the trees or it snows. The seasonal changes occur on Mars too, in the form of Martian polar ice caps melting - the caps grow or shrink according to the season. For long-term navigation, day-light changes are not easy to understand and map correctly but there have been several approaches to model surroundings visibility during the whole day. It is essential for domestic robots to detect whether an object cannot be seen or was removed. Navigation of domestic robots is challenged by daylight changes, object replacement and movement. Fortunately large furniture remains consistent over time. Indoors the change in structure is usually imposed by the movement for small furniture.

The structured environment has a lower uncertainty due to the defined structure. Compared to that, the unstructed environment has a higher uncertainty due to typically higher dynamics. To answer all of the aforementioned questions, a robot should have a suitable model of the environment. If a robot has to operate perpetually, the model should encompass not only the world structure, but also how the world changes over time. In this work, we focus on modelling, understanding and predicting the environment changes that occur over time.

1.1 Motivation

The goal of our work is to create a robust temporal model of surroundings, which is meant to support the visual navigation in long-term scenarios. We intend to achieve the result by using probabilistic methods, that model uncertainty caused by environmental changes and variations.

We have to deal with operational environment changes. This requires new information captured during autonomous runs over time to answer the primary questions. There are more ways to cope with it. Robot builds a map to navigate itself. The map is updated to capture the difference as the environment alternates. This approach presumes that once the change occurs, it is permanent. In another approach, a map is created at the start and abnormalities are captured, processed and saved in a model. We aim to indicate which objects from currently processed map should be used. The decision is supposed to be based on rides that occurred in history. Once we detect s landmark it is marked as either correctly matched, not matched or incorrectly matched. We divide the problem into two parts: evaluation of saved data and submaps selection. We plan to keep those parts separated for flexibility. This work answers two primary questions:

1. How should the probability of visibility be calculated?

2. What strategy should we use for correct submap selection?

We claim that learning from the past and using this knowledge in the future is crucial for long-term visual navigation.

Chapter 2

State of the art

Reliable navigation is essential for any autonomous vehicle. The reliability problem becomes more challenging for long-term navigation systems, because of the environmental variations. It is easier for people to understand and predict gradual changes of environment appearance due to their experience gained over time. We present a solution in the form of a temporal model of the environment to help the system gain and understand this experience. For creating a temporal model to improve the visual based self-guiding robot navigation, it is essential to understand how the entire process works. As mentioned earlier, the navigation process is divided into three primary questions: "Where am I?", "Where am I going?" and "How should I get there?" [8]. In practice the deployment of a robot is typically performed as follows:

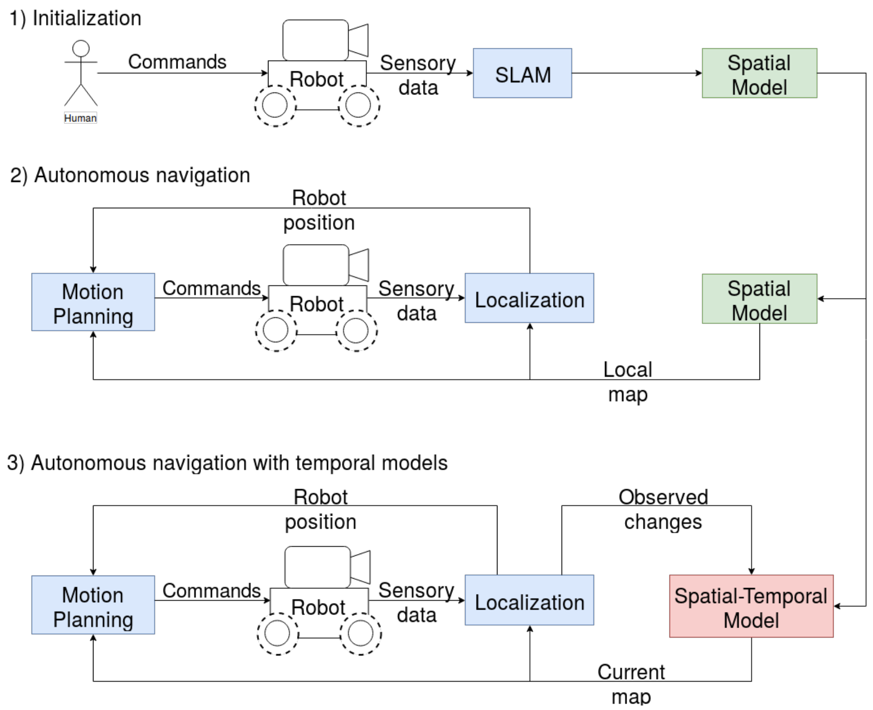


Figure 2.1: Process diagram of a visual autonomous navigation.

2.1 Initialization

Before a mobile robot is able to navigate itself autonomously, it must learn the paths it's allowed to go and what the operational environment looks like. To do so, a person operates the robot manually for the first time [11]. The robot builds an inner spatial representation using data from on-board sensors [9]. A process of creating spatial-model is called mapping. After the robot stores the map of the whole operational environment, it can

1. determine its position
2. determine a position of the desired destination
3. plan its path.

It generates motion commands itself, and a human aspect is not needed for navigation anymore. It is typically problematic to derive the inner spatial representation from the robot's position and therefore we use SLAM to build the model.

2.1.1 Simultaneous Localization And Mapping (SLAM)

The SLAM is a class of online map building methods [3] where the robot performs localization and mapping concurrently. SLAM combines position estimation and on the fly environment map building. Robot's location is usually described by a position and an orientation in space but other quantities can be used as the state description, e.g. robot velocity, sensor biases or calibration parameters [9], representing the uncertainty of the environment. Current approaches assume that the uncertainty originates mainly from the sensor noise. The map represents aspects of interest including a position of landmarks and its descriptors. The SLAM problem is formulated as a maximum a posteriori estimation problem [12]. Robot calculates the join a posterior density [11] of latent variables, which includes a sequence of positions in trajectory and the state of interesting features in the environment.

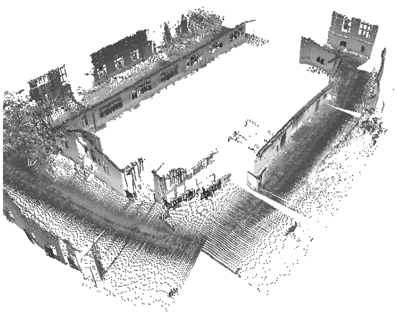
It is possible to use the SLAM to deal with environmental variations. SLAM can update environmental model and take significant changes into account, the map remains static after the update [12].

2.1.2 Mapping

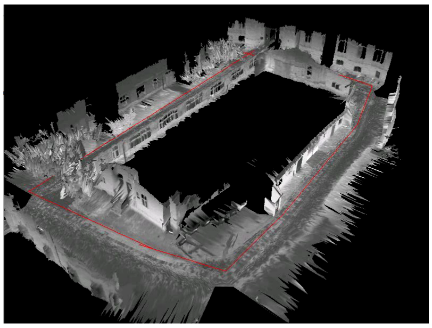
To answer the questions "Where am I?" and "Where am I going?", we need to be able to remember and recognize the visited places in the future. We create maps to provide a system with an ability to recognize the environment that has been already visited. The map is an integrated representation of an environment the vehicle operates in. It is supposed to be compared to the incoming sensory data [3]. The map can be built online with on-board sensors or be known a priori. The online building method is SLAM, described in the previous section. It is possible to create simple environmental models manually, e.g. beacons positions. We can divide maps into four types according to the data representation [13] and the abstraction level.

Sensory Maps

Sensory maps consist of pure sensory data representation. An example of a sensory map is a 3D point cloud that is created by a laser scanner. Although this technique is simple for data storing, it's not practical, because of the large memory demands. Sensory maps are useful for further processing. If the purpose of the environment model is a place visualization, the 3D point cloud is used for creating a 3D mesh.



(a) : 3D point cloud. Courtesy of [14]



(b) : Occupancy grid. Courtesy of [15]

Figure 2.2: Different models representing the same place

Occupancy Grid

Occupancy grid represents the environment models as a grid consisting of uniform cells where each cell is either occupied or free [14]. Each cell stores the probability of its occupancy [16]. We calculate the probability of occupancy separately for each cell because the probabilities are modeled as independent. The occupancy grids are suitable for both motion and path planning and localization, which is the purpose of occupancy grid environment representation introduced in [16].

Geometric Maps

Geometric Maps represent the world as a set of geometrical primitives [14]. A two-dimensional representation is created with lines and polygons and a three-dimensional with planes or bodies. The abstraction associated with the geometric primitives usage brings memory efficiency; therefore geometric maps are more suitable for a mobile navigation than sensory maps [13]. Since the real world consists of more complex objects than used primitives, creation process becomes difficult.

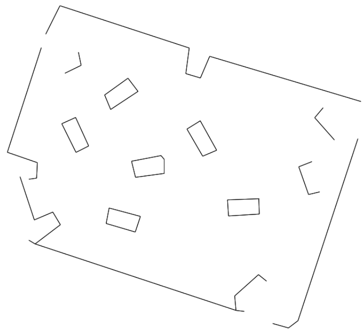


Figure 2.3: Two-dimensional geometric map. Courtesy of [2]

Landmark Maps

Landmark maps include information about significant points or objects in the view and are frequently used in the vision-based navigation domain. The landmarks are primarily used for localization [14]. Typically once they are detected, robot compares landmarks stored in the map and the landmarks from the current view and then calculates the horizontal landmarks shift and uses this information to estimate its position. A visual navigation system called ORB-SLAM presented in [17] uses the landmark map for the environment representation. The landmarks are represented in the 3D world coordination system. The robot positions is then estimated by the triangulation method using the matched landmarks from the view to the landmarks in a map.

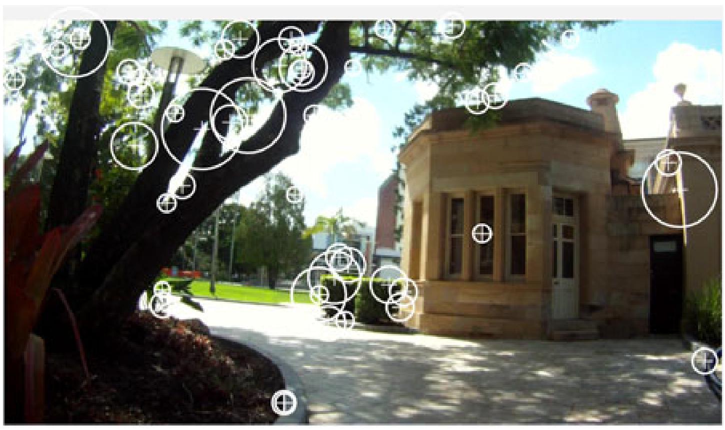


Figure 2.4: An example of a landmark map. Courtesy of [3]

Topological Maps

The advantage of the topological maps lies in the abstraction level. The data stored in topological maps represent a graph where nodes constitute of distinguishable places and paths between them. Topological maps are an abstraction, memory efficient and suitable for a mobile navigation and a high level path planning. An example of a topological map is a public transport map (in figure 2.5) where stations represent nodes and edges are paths between them. The system introduced in [18] uses only a visual topological map for the navigation. The system distinguishes roads and places of roads crossings. The environment is represented by a topological map where nodes represent roads crossings and edges represent existing roads that connect two crossings.

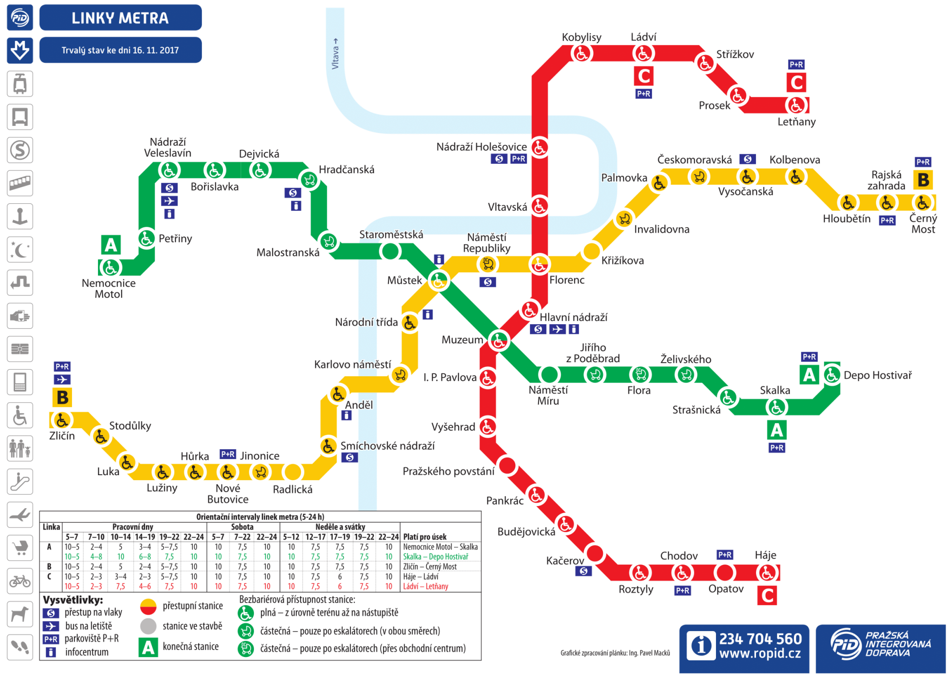


Figure 2.5: Trams and Metro in Prague map. Courtesy of [4]

Hybrid maps

The hybrid map combines two or more map types and therefore a usage of more maps allows to reduce problems associated with a certain map type. The disadvantage of a landmark map being used separately can cause for example the Perceptual aliasing problem [19]. The problem occurs when the two places are mismatched because they look alike (e.g. hallways in different floors of the same building or hedges on the same countryside). The problem can be resolved by adding a topological map which refines localization. The Large Maps Framework [20] represents the world as the topological map with additional information. The vertices of the map represent places of interest and edges of the topological map represent possible roads that connect places of interest. Vertices and edges are described with additional information, which is used for planning, interfacing and reasoning about the environment. The Atlas framework [21] also uses hybrid map, which consists of topological and metrical map. It represents the world as a graph of multiple local maps, where vertices represent local frames and edges represent the transformation between frames. It creates a map per every frame describing the local environment and robot pose including uncertainties of each. The Atlas framework enables a robot to map and localize within an unknown environment. In [22], the operational domain is represented as 3D Occupancy Grid, where the cell occupancy probability is described by time depending function. The Fourier Transform is used for retrieving the probability function. This model is able to predict the periodic environment variations.

2.1.3 Image processing

One type of landmarks that are stored in landmark maps is so-called *image features*. The image features identify significant points in images, usually points with the high localized contrast in a small space. The set of image features can identify an object or a scene or a location. The features are extracted from images in two phases feature detection and feature description [23]. Detectors search the input image to identify significant points. The enable the feature reidentification we stored its descriptors, which goal is to identify the image features which are invariant to the illumination in the future even though the viewpoint is shifted or the illumination has changed.

2.1.4 Localization

The knowledge of own location relatively to the goal destination is essential for a successful navigation. We determine the robot's location in the relative or the absolute frame [13]. We present three types of localization: Dead Reckoning, Beacons based localization and map-based localization [14].

Dead Reckoning

Dead reckoning localization method is a process that determines the position in the relative coordinate frame. Location in the relative frame is calculated relatively to the starting or the previous point. In dead reckoning the position is not measured directly. Instead, the robot measures its velocity or acceleration, which is integrated over time [14]. The odometry is measured by motion sensors and then we calculate the difference from the previous point. Although precise sensors and careful calibration decrease measurement difference from reality, the error accumulates with time [14]. Thus dead reckoning is not suitable for a long-term navigation.

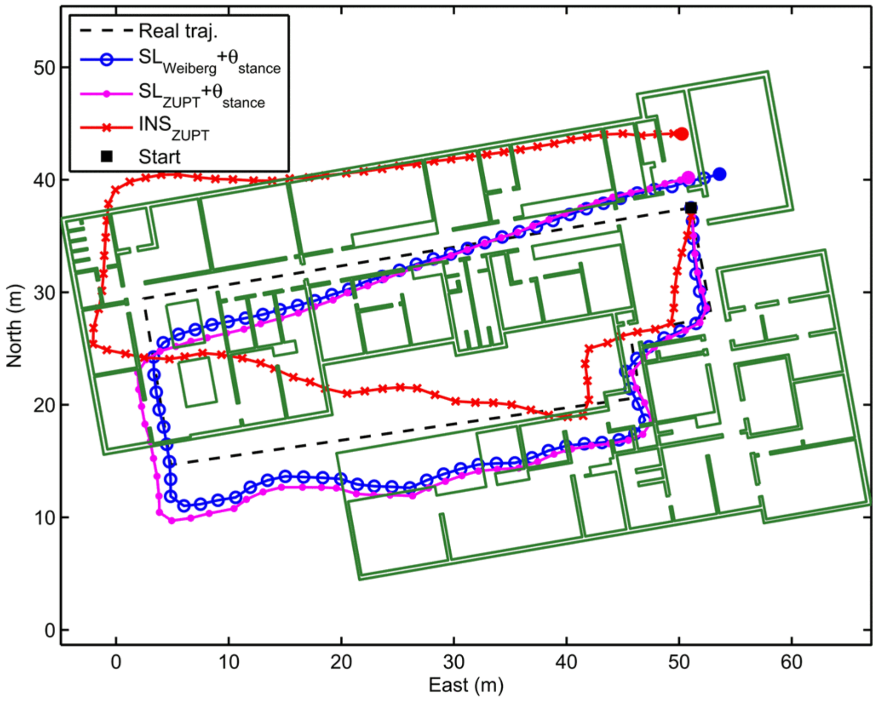


Figure 2.6: Demonstration of cumulated error using Dead Reckoning localization. Courtesy of [5].

Map based localization

Map based localization uses previously constructed maps to compare incoming sensory data with the additional data stored in the environment model. The map based localization is usually combined with dead reckoning localization method because this approach is able to reduce the accumulated error when the position is calculated relatively to the previous location [10]. It is a process when we build a map to be compared with the current view to adjust the location derived from the odometry and reduce the error.

Beacons based localization

Beacons based localization determines the position in the absolute frame defined by beacons. The system is either global or local. The beacons are artificial objects placed in the environment with known position. The robot is equipped with the a priori known map which contains only beacon positions [3] and a detection system to find beacons and determine robot's position in the system. The vehicle finds beacons in robot's surroundings and estimates its position using visible beacons [3] by a triangulation method [13]. An example of the beacon localization method is the Global Positioning System (GPS).

2.1.5 Motion planning

To answer the "How do I get there?" question, we generate a series of control signals to reach the goal, e.g. in [10]. We can divide motion planning into two parts: path planning and the motion control [14]. In [10], the system uses a hybrid map for a world representation, consisting of a topological map which represents the path as line segments and places of interest as end of segment, and landmark map, which models the robot view in places of interest. Robot drives along the straight line segment and rotates to face the end of a segment at every segment's starting point. To find the right direction, the robot compares coordinates of the image features stored in the landmark map. The histogram voting method estimates a modus of each horizontal feature difference. The angle of the rotation is calculated from the histogram bin with the greatest number of features. The robot plans its path to decrease the angle and head toward the next end of the segment. Then the appropriate motion commands are sent to the robot and it moves towards the next mapped location.

2.2 Autonomous navigation

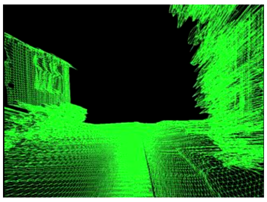
Autonomous navigation requires the knowledge of the trajectory and surroundings. We use the teach-and-repeat method to provide the robot with necessary information. The robot first learns the environment model and trajectory during the manual navigation. Once the robot gains knowledge about its surroundings, it navigates itself autonomously. The whole process starts with localization. Once the vehicle is aware of its position, it plans where it should go to get closer to the next goal position. The last part of the process is motion planning and the robot generates motion commands to reach the goal. Once the robot can autonomously operate within the mapped environment, it must decide what to do with environmental changes. The easiest approach is to ignore all changes, the model becomes obsolete, which makes the uncertainty grow over time. Another approach is to remap the environment when a change is detected. A failure can occur when remapping which can completely destroy the map. The last approach to deal with environment variations is to learn the behaviour of environment dynamics within changes and predict the feature environment alternations in future.

2.3 Long-term autonomous navigation

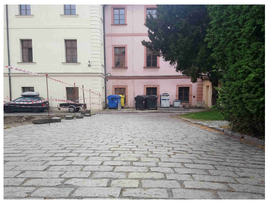
Typical autonomous navigation assumes the inner world model created by SLAM or teach-and-repeat techniques to be static [12]. But the world we live in is affected by changes. We face daylight or natural season changes but there are changes induced by a human activity (e.g. an object movement or removal) [24]. In the short-term perspective, we can neglect the changes and keep the map static. The long-term navigation is influenced by environmental changes and therefore we create a tool for the system to deal with and understand the environment evolution. To represent the changes, we develop a temporal model and use it to improve the spatial model by adding time dependence, which results in the temporal-spatial model building and map updating.

Figures 2.7 and 2.8 demonstrate the need for temporal models. The figures 2.7a and 2.7b show how the environment appeared to a robot when it was building a map in 2006 and figure 2.7c shows what the robot's view would be today in 2018.

(a) : Robot's view in 2006.
Courtesy of [15].S

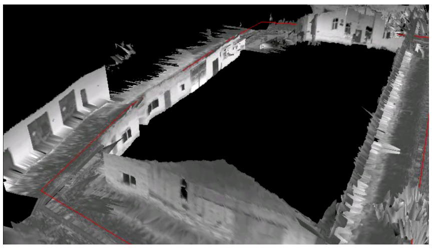


(b) : A part of created map
in 2006. Courtesy of [15]



(c) : View from the robot
position in 2018.

Figure 2.7: The views of CTU Charles square campus from robot position



(a) : Final environment map in 2006. Courtesy of [15]

(b) : Today's appearance of the mapped environment in 2018.

Figure 2.8: The CTU Charles square campus appearance

2.3.1 Environment changes modelling

In long-term navigation, we are forced to face a dynamic environment with moving objects in it, however the model is often assumed to be static [12]. Although many maps and real-world differences can be corrected by map updates, we create a temporal model to learn and predict the environment evolution. The purpose of the model is to represent changes throughout the time according to the observed variations. To keep the problem simple, we built the model for one viewpoint feature, landmark or object depending on what the map data represents. We learn the state of the monitored object while the robot performs localization and compares its view with an appropriate map. The object is matched correctly, mismatched or not seen at all. This information can be explained in multiple ways and therefore we present multiple types of dealing with environmental changes in the following section.

Map updating

It is possible to deal with changes by updating the spatial model of the operational environment. An overview to map update approaches is presented in [9]. The Simultaneous Localization And Mapping (SLAM) consists of the environment model and the robot position estimation. The SLAM is defined as a maximum a posterior estimation problem [9] where the robot's position is estimated according to on-board sensory incoming data. Each location is associated with the map, the sensory data are processed and compared to maps, and the robot location is estimated as the place described by the sensory data. The problem of the map updating is that the SLAM system has to recognize if the map update is needed. When the map is updated the environment model is usually permanent until the next update is made. This approach does not support the understanding the changes and their subsequent prediction.

Environment mapping

In [3], the author poses a fundamental question: "What is a place for a robot?" and represents several types of environment models as point or line image feature representation, a model using a 3D scanner for creation and recognition to avoid the light changes, but this approach does not model changes of image features or whole objects within a map describing the view. It creates multiple types of one place representation and selects the one to be used according to incoming sensory data or gained knowledge of cyclic changes of world representations. Once the place is considered as problematic, new "experience" of a place is created and ready to be used in future. Trajectory consisting of *experiences* is shown in figure 2.9a.

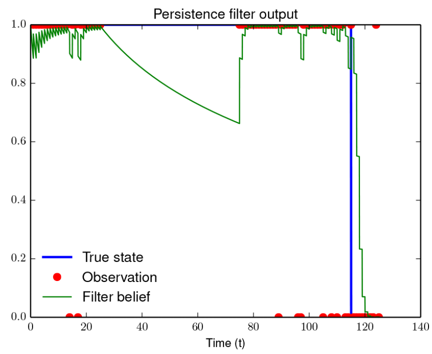
Moving objects detection

The problem of creating environmental models from on-board incoming sensory data is that the final environment map consists of both static and dynamic object. The dynamic object is such which changes its position frequently in a small amount of time and the static object usually occupies one place for a long time or doesn't move at all. We consider the static objects to be, e.g. trees, houses, roads, furniture etc. The examples of dynamic objects are people, cars, animals etc. The change modelling in [24] is meant to detect a dynamic object and predict its trajectory using an average of flow vectors associated with the object.

Generated BRIEF

The visual-based navigation system using a video camera as the main sensor for the world detection is prone to the light changes, which was the motivation in [23] article. It creates a new feature descriptor based on a combination of Binary Robust Independent Elementary Features (BRIEF). BRIEF uses binary strings as features, which make the feature processing more efficient, and evolutionary algorithms. The final descriptor is called Generated BRIEF (GRIEF). The principle of GRIEF is to learn the feature descriptors from occurring changes, which makes the feature that uss this descriptor to be robust and resistant to environment changes. GRIEF uses the BRIEF descriptor which consists of interest points. The interest points are divided into two sets: correct and false, by, e.g. histogram voting scheme [23]. The sets serve as positive and negative training samples. The GRIEF descriptor improves the feature description with every iteration and therefore the environment changes are included in the descriptor training sets, which makes it more robust for the changeable environment.

(a) : Trajectory consisting of multiple experiences. Courtesy of [25]



(b) : Feature persistence filter. Courtesy of [12]

Figure 2.9: Method of environmental changes capturing.

Feature persistence filter

In [12], the world is considered as continuously changing which is represented by a persistence filter that is supposed to filter the image features that probably disappeared and are likely not to be seen anymore. The world is represented by a collection of a maps each of which one corresponds to a one place. The place is repeatedly visited and the persistence filter calculates an explicit Bayesian belief of feature appearance. The belief is in [0,1] interval and consequently the removal threshold is selected. The threshold is represented by a function that depends on the feature appearance observation. Once the feature crosses the belief threshold, it is removed from the environment model. The feature visibility prediction using the Persistence filter is displayed in the figure 2.9b.

Frequency Map Enhancement(FreMEn)

FreMEn temporal model describes the feature visibility by a harmonic function. The function shows the probability of environment states over a period. The model learns from a set of observed variables of time and state where the state describes if the feature was visible or not. The model creates a harmonic function of visibility over the period. The period length changes according to the feature behaviour. The model predicts if the feature will be visible in given time. This allows predicting which particular features are the most likely to be visible at a point in time. Figure 2.10 shows the learning of one feature visibility and its prediction.

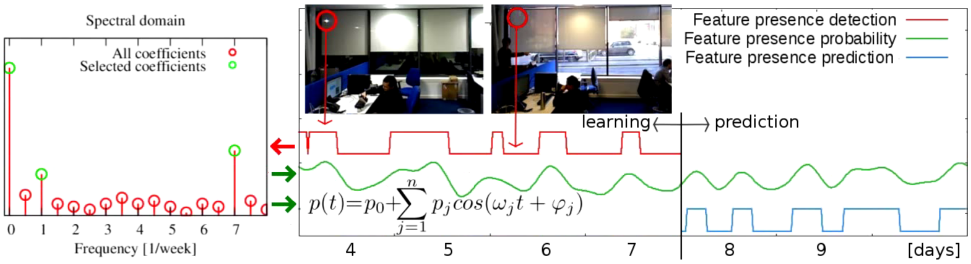
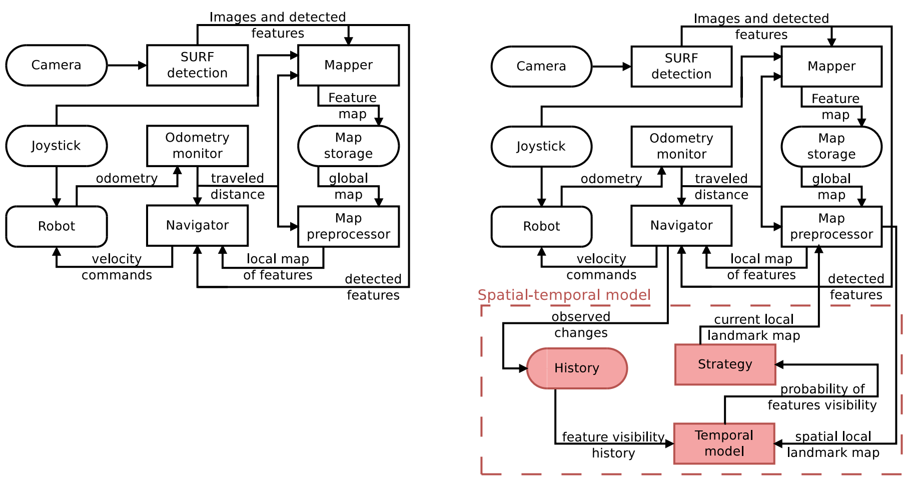


Figure 2.10: Feature visibility prediction using FreMEn. Courtesy of [6]

Chapter 3

Navigation System Description

Our navigation system is based on an existing system called BearNav [26], we chose this system for the simplicity of use, its ability of long-term navigation using an elegant spatial model of the environment, teach-and-repeat method and open source availability at `www.github.com/gestom/stroll_bearnav`. The system structure based on the stationary model is displayed in figure 3.1a and further in this chapter we will describe each module in details.



(a) : The original system **(b) :** Extended system

Figure 3.1: Navigation system structure

The navigation system is divided into initialisation and repeat phases. The initialization phase was explained theoretically in the section 2.1. In the first phase initialisation, the robot is guided manually along the path that is supposed to be travelled through autonomously in the future. While the robot is navigated by a human operator, the robot extracts significant image features 2.1.3 from incoming data from an onboard camera. It stores the extracted features as well as its velocity and the travelled distance along the taught trajectory. In the repeats phase, the robot uses directional localization and repeat velocity commands taught by a human operator while correcting its heading based on the previously remembered and currently visible image features.

3.1 Sensors

To keep the focus of this work on the vision-based navigation, we select the video camera to be used for map building and heading estimation. The system can be enriched by additional sensors, e.g. a compass to determine the orientation and sonar sensors to avoid collisions.

3.2 Image processing

Image processing is essential for a visual navigation. The Speeded Up Robust Features (SURF) method is used for features identification in this system. The extractor has two functionalities, and it is features detection and description. SURF provides features coordinates within the image.

3.3 Map

The map used in the system is a hybrid map (2.1.2) which consists of a global topological and a local landmark map. Unlike in [26], where only the map from a human-guided tour is used, we update our map from autonomous runs. The hybrid map is used to solve the *Perceptual aliasing* mentioned in Section 2.1.2. In the initialization process the robot measures and saves the travelled distance and changes in forward or angular acceleration which constitutes the path profile [27] which represents the topological map. The topological map is the only one for the path. Every 1 m travelled distance saves the processed image from the on-broad camera which represents landmark local maps consisting of image features. The used image processing is described in 3.2. Each feature is represented as a set of variables which are the image, the first and the last the visible distance and its descriptor.

3.4 Autonomous navigation

Autonomous navigation uses the simple repeat principle. When the robot is supposed to navigate itself, it relays the commands. To prevent the error accumulation, the robot compares the features extracted from the incoming camera data to the features from the relevant local map chosen according to the travelled distance. The robot computes the horizontal difference between the mapped and the currently visible corresponding features by the *histogram voting* method (represented in section 2.1.5) and saves it into a histogram to the appropriate bin. The bin with the largest amount of features represents the angle the robot turns wheels to face towards the feature which corrects the error and provides a reliable navigation.

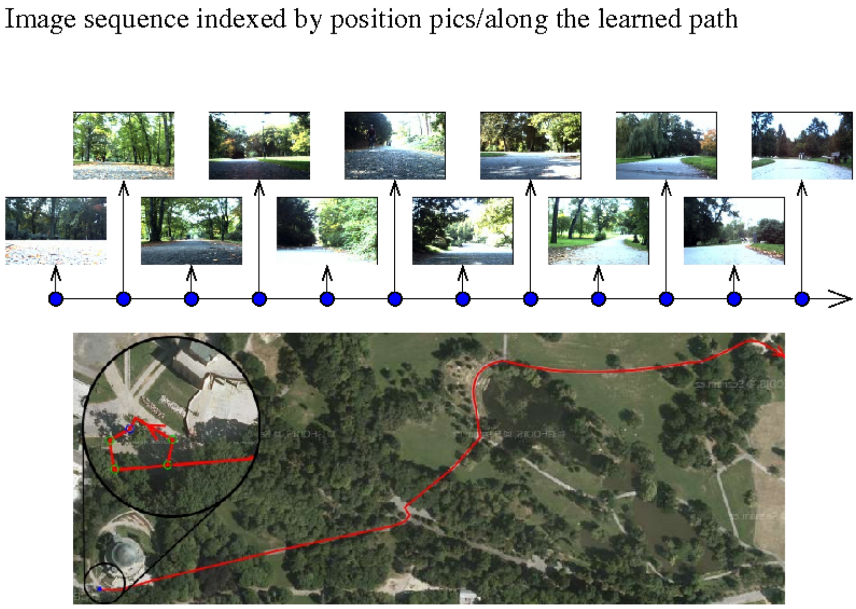


Figure 3.2: The ilustration of global topological map consisting of local landmark maps.

3.5 Long-term autonomous navigation

We modified the navigation system to enable its deployment of the aforementioned system for long-term use. We changed its spatial representation in a way which allows it to reflect the environment changes. In particular, we used a spatio-temporal model to generate a temporally local spatial model that is used by the system for the navigation. We presented two primary question necessary for reliable long-term navigation:

1. How does the probability of visibility depend on time?

2. How should we build the temporally local map?

We added two separate modules that allow us to study these questions. The module related to the first question is the Temporal model module and the module related to the second question is the Strategy module. Developement and integration of these modules are the goal of this thesis. The extended system with integrated Temporal and Strategy modules is shonw in the figure 3.1b.

The temporal model module learns from past observations of the same locations performed by the robot during a routine operation and assigns each feature to a visibility probability, and therefore the module relates to the first primary question. The strategy module filters image features according to the probability calculated by the previous module which relates to the second primary question. The system structure of the modified navigation system is shown in the figure 3.1b. The figure 3.3 shows sequence diagram representing the process of predicting the probability of feature visibility and filtering.

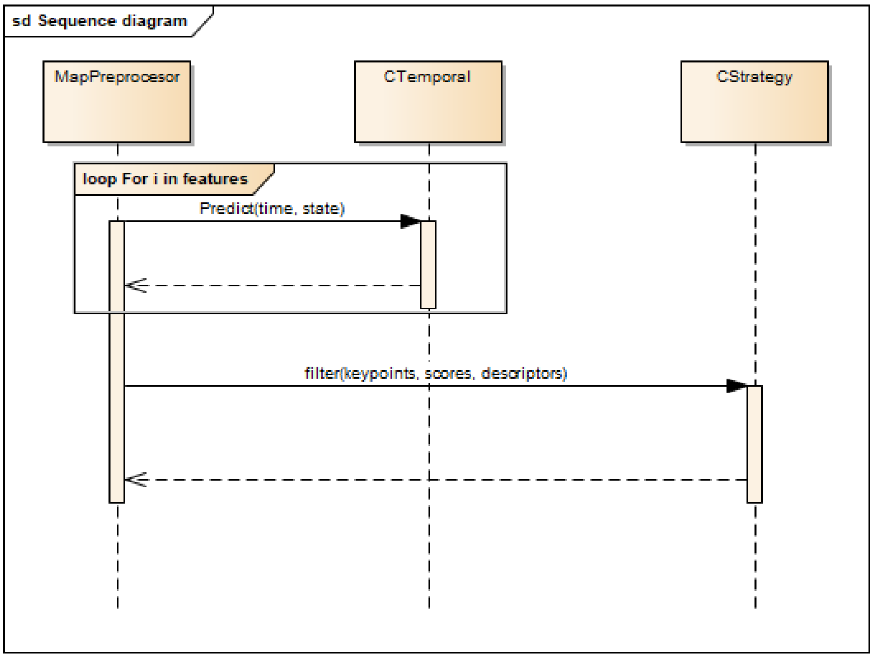


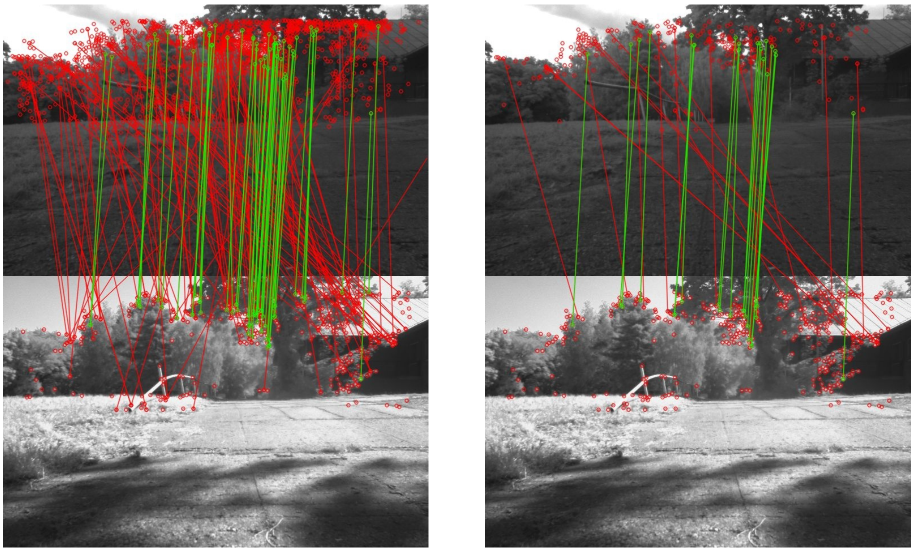
Figure 3.3: Sequence diagram of feature evaluating and filterings

3.6 Spatial-temporal model

The spatial-temporal model provides knowledge of operational environment evolution. We learn from history of observations; the observation contains feature identification and its state and time when the observation occurred. When matching currently visible features to the ones from the map, a given feature can be either correctly matched, not matched or incorrectly matched. Our goal is to filter the features that are likely to be matched incorrectly. This reduces the chance that the robot will turn in incorrect direction and cause navigation failure. The filtration is demonstrated in the figure 3.4. In other words we want to predict features that are more likely to be correctly matched to estimate the direction estimation of the navigating robot correctly.

3.7 Temporal model

The Temporal model is supposed to solve the first aforementioned primary question for reliable long-term autonomous navigation: "How does the probability of visibility depend on time?" The goal is to calculate a visibility score of each feature that is supposed to model the feature visibility probability.



(a) : Feature matching without a filter. **(b) :** Feature matching with a filter.

Figure 3.4: Feature matching

The model creates an inner representation of features and predicts what the visibility likelihood will be.

We use a statistics based approach to calculate the score of visibility, which is directly proportional to the probability of visibilty.

Sum

This temporal model calculates a sum of past states over time. The purpose of this temporal model is to separate stable features from the unstable, which are usually not seen or mismatched. The result score of the feature is given by the following equation:

$$\sigma = \sum_{i=1}^{n} s_i, \tag{3.1}$$

where s_i is the i-th state detected in history, n is the total number of measured states.

Weighted Sum

This temporal model calculates a sum of given states over time as the previous model, but the negative state is weighted to separate the features that are incorrectly matched and features that are not detected. The purpose of this temporal model is to separate stable features from the unstable ones which are the usually not seen or mismatched. In contrast with the previous model the weight of negative state is assigned to the model, which is supposed to filter the incorrectly matched features faster than the ones that were only not seen. This temporal model is given a weight, how many times is it worse for the feature to be mismatched than match correctly. The result score of the feature is given by the following equation:

$$\sigma = \sum_{i=1}^{n} s_i + \sum_{j=1}^{m} w s_j, \tag{3.2}$$

where s_i is the i-th correctly matched state detected in history, n is the total number of correctly matched feature states, s_j is the j-th incorrectly matched state, w is given the weight of s_j, and m is the total number of incorrectly matched features.

If the final score of visibility is positive, we can say that the feature was at least w times more correctly matched that mismatched.

Sliding average

The Sliding average temporal model evaluates the observed feature state with respect to the observation time. The older the observation the smaller value it gains. The purpose of this temporal model is to assign the states that occurred in older history lower weight than of the latest one. It assumes that the environment has gradually changed over time and the latest visit describes the environment the best but every state affects the model. This temporal model works a lot alike the presistance filter (presented in the section 2.3.1) except this approach does not use Bayesian statistics. The result score of the feature is given by following equation:

$$w_i = e^{-\frac{t-t_i}{\tau}}, \tag{3.3}$$

$$\sigma = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} w_i s_i}, \tag{3.4}$$

where t is current time, t_i is time, when data was collected, τ is predefined time interval, in our case $\tau = 12$ hours, s_i is the right matched label, (It is 1, if feature was matched correctly and it is -1, if it was matched incorrectly).

Frequency Map Enhancement(FreMEn)

The FreMEn temporal assumes that the feature visibility is possible to be represented as a harmonic function. The most significant changes of feature appearence are caused by light changes. The most significant light changes are observed periodically as day turns to night and opposite. This temporal model uses the Furrier transformation to estimate the harmonic function describing the feature visibility. The model uses cosinus harmonic funtion for feature visibility probability estimation. The temporal model description in detailes is provided in [6]

3.8 Strategy

The strategy relates to the aforementioned second primary question: :How should we build the temporally local map given the likelyhood of feature visibility?". The purpose of the strategy is to select enough features for a reliable navigation while filtering out the ones that are more likely to be incorrectly matched. When we filter features, then the filtered ones are hidden to the robot completely so it doesn't try to find them in its view. It means we don't get the information about filtered features state, and therefore we face another problem and it is known as the *exploration vs. exploitation* problem [28]. Our goal is to select features, which are the most likely to be recognized correctly, that present the exploitation. If we always select the same features, we don't get the information about the rest of unselected features and they stay hidden because the visibility score doesn't update.

We created three selection strategies that are described in the following section. Each strategy method is associated with an illustrative figure where the strategy is given by the visibility scores, which are displayed in red color. The width and saturation of color illustrate the size of visibility score. The white array represents the final features selection.

N best

The N Best strategy assumes that for a reliable navigation we need n features, which are most likely to be visible. The figure 3.5 shows two examples of the N Best filtration strategy.

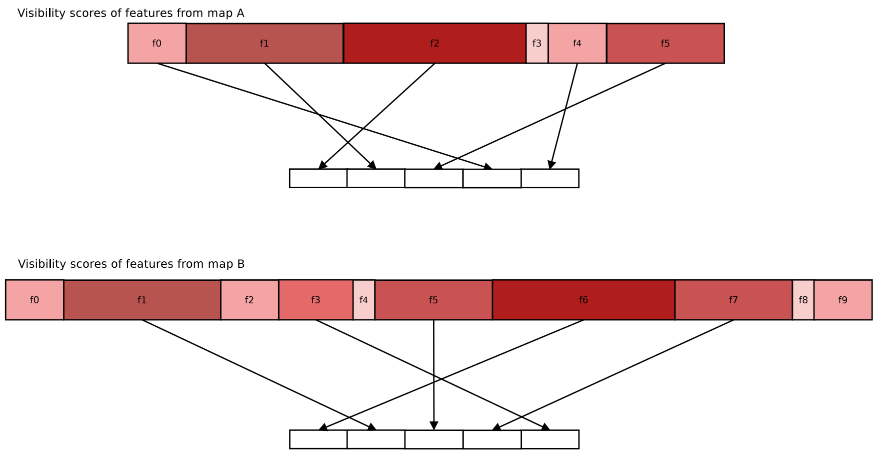


Figure 3.5: Example of N Best strategy selection.

Quantile

The Quantile strategy selects the fraction of the total number of image features. This strategy represents the idea that we need a predefined percentage of image features for reliable navigation. This strategy selects the fraction of total amount of features. Unfortunately features with the lowest probability of visibility won't be selected as long as the other features will have a higher probability. The figure 3.6 demonstrates two examples of the Quantile feature selection strategy.

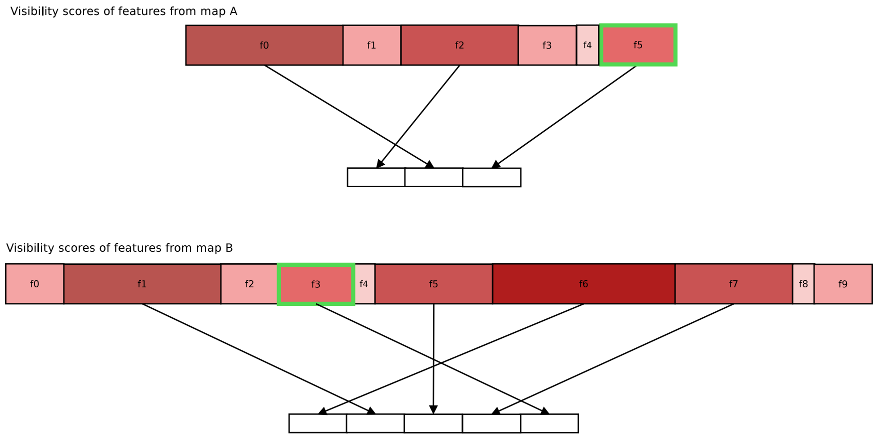


Figure 3.6: Example of Quantile strategy selection.

Monte-Carlo

The Monte-Carlo strategy is based on a roulette wheel selection [29]. The probability of being selected is directly proportional to the calculated score of visibility by temporal model. This approach solves the exploration vs. exploitation by adding a random factor into selection. Features with a higher probability of visibility are more likely to be selected but the set of selected features is not determined. The figure 3.7 represents two examples of the Quantile feature selection strategy.

3.8.1 Software architecture and implementation

This section describes, how the actual software is implemented. The whole platform is implemented in Robotic Operating System (ROS) which is described in the next section. The system is written in C++. We created three temporal models and three strategies for the feature selection.

3.8.2 Robotic Operating System

The system is implemented in an open source Robotic Operating system [30] available at `http://www.ros.org/`, which is a "middle ware" system for robots. It provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes and package management [30]. The system works on a subscribe - publish

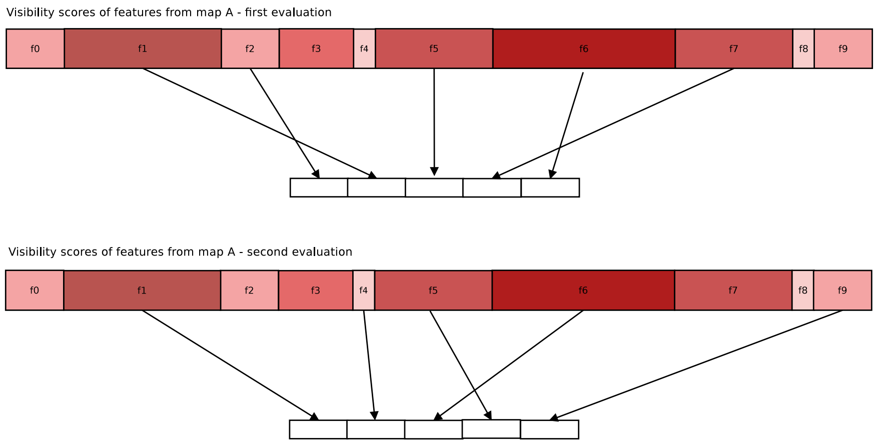


Figure 3.7: Example of Monte-Carlo strategy selection.

principle. One module publishes a topic to which the other model is subscribed to receive all topic messages which is realized as an assynchronous communication. ROS provides multiple types of communication i.e. synchronous over services, asynchronous over topics. ROS also implements an infrastructure for a data storage on Parameter server which provides the ability to a change parameter while the system is running. Another advantage of ROS is that it is a language-independent system. The supported languages are Python, Lisp and C++. The ROS system is suitable for practical and theoretical testing of algorithms because of the feature ability of low level and high level abstration. The ROS inner representation of extended system is shown in the figure 3.9.

3.8.3 Temporal models

We created three temporal models Sum, $Weighted\ Sum$ and $Sliding\ Average$. Each temporal model implements the CTemporal virtual class. This virtual class was used in project FreMEn [10] available at `https://github.com/strands-project/fremen`. We use the main functions *add*, *predict* and *update* where the prediction has two parameters: time in seconds and state of the feature in a float number. The state value is either 1, 0 or -1 where 1 represents a correctly matched feature, 0 is assigned when the feature was not seen at all and the value -1 corresponds to mismatched feature. In our system "The virtual class" was modified to accept string ID and a new function setParameter was added to allow changes in runtime. One temporal model is assigned to one particular feature described by the string ID.

3.8.4 Strategy

We created three temporal models $N\ best$, $Quantile$ and $Monte\ Carlo$. Each temporal model implements the CStrategy virtual class that we created. The

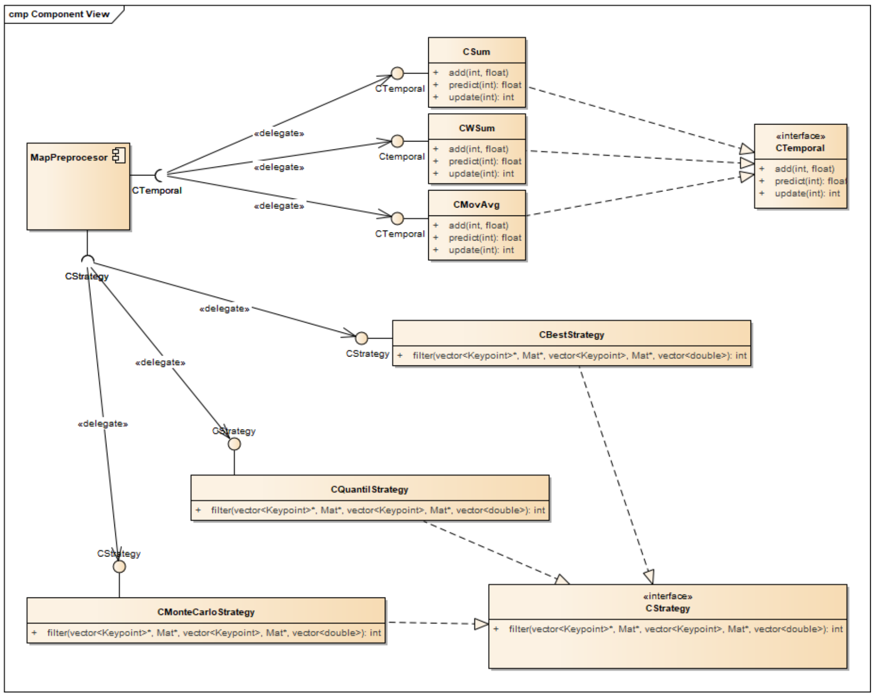


Figure 3.8: Component diagram

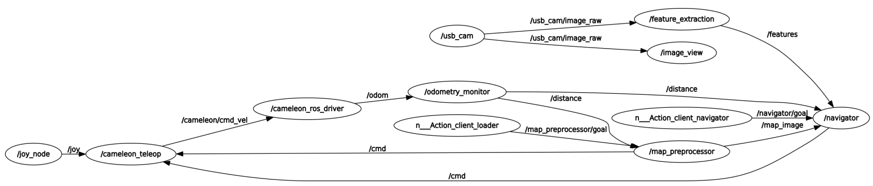


Figure 3.9: Diagram representation of system using ROS. Vertices represent modules and edges represest publishing and subscribed rostopics.

strategy is initialized only once for all features.

N best

The N Best strategy assumes that for a reliable navigation we need N features which are most likely to be visible. The Best strategy has one argument and it is n, the number of features that are supposed to be selected. If n is greater than the total number of features, the vector of features is returned unchanged. The best n features are selected and the rest of it is removed from the vector. The robot will no longer try to match the removed features which doesn't solve the exploration vs. exploitation problem.

Quantile

The Quantile strategy selects the fraction of the total number of image features. This strategy represents the idea that we need a predefined percentage of image features for a reliable navigation. The Quantile strategy has one argument and it is p, the minimum probability a feature has to gain to be selected. The scores are sorted by ascending. Then the threshold index is calculated. We calculate the index i:

$$i = pn, \tag{3.5}$$

where n is the total number of features, and p is the given probability. If the result i we calculate the threshold value as mean of i-th and following score: $\frac{\sigma_i+\sigma_{i+1}}{2}$. Then we filter features with a score σ lower than the threshold value. Because this approach uses the threshold value to select features, not the total number of selected features, the strategy is more abstract than the previous one. This strategy selects the fraction of the total amount of features. Unfortunately features with the lowest probability of visibility won't be selected as long as the other features will have a higher probability.

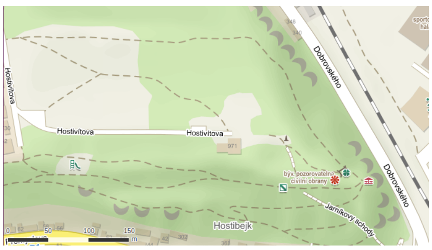
Monte-Carlo

The Monte-Carlo strategy is based on a roulette wheel selection [29]. The probability of being selected is directly proportional to the calculated score of visibility by the temporal model. The strategy is provided with n, the number of features to be selected. If n is greater than the total number of features, no filtering occurs. The Monte-Carlo strategy assigns each feature an interval according to its score. It finds the lowest score and if it is negative, it adds an opposite value to every feature score. The interval starts at zero then each feature is given the end of its interval. We store the end of the last feature interval as m. Then the n numbers are generated by random; the feature is selected if the generated number is in its interval. If one feature is selected more than once, we generate a new number. All unselected features are removed. This approach solves the exploration vs. exploitation by adding a random factor into the selection. Features with a higher probability of visibility are more likely to be selected but the set of selected features is not determined.

Chapter 4

Datasets and Evaluation

We chose to test our long-term navigation system outdoors. The place where the dataset was collected on was the Hostibejk hill in the Kralupy nad Vltavou town, the Czech Republic in the middle of August 2017, shown in figure 4.1a. Hostibejk was declared a natural heritage in 2002 [31]. The hill is covered by the forest, and therefore the natural changes in appearance and structure are significant.



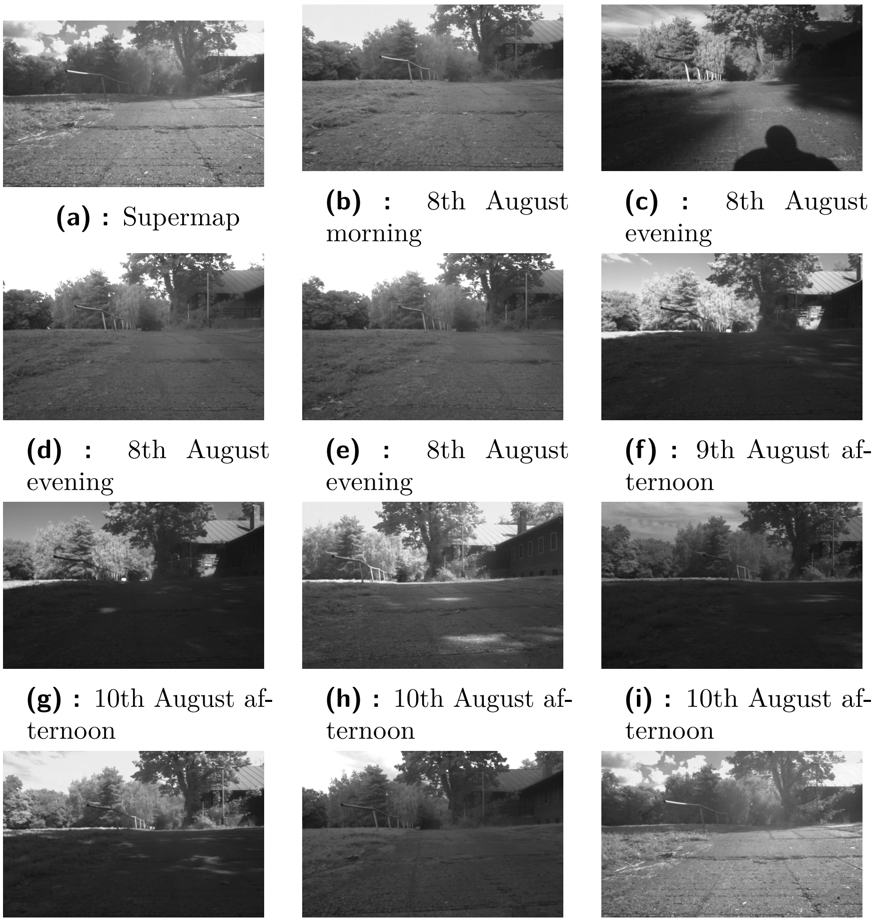
(a) : Map of Hostibejk hill. [32]

(b) : Robot used for data collection. Courtesy of [33]

Figure 4.1: Dataset lacation and used robot for its collection

We created datasets from the experiments that occurred from 8th August 2017 to 17th August 2017 [34]. These experiments of autonomous navigation were saved into rosbag files, which is the ROS primary storage format for sensory data. The rosbag file stores sensory data from robot autonomous drives. The rosbag files represent an example of a sensory map (section 2.1.2). The data stored in rosbag files contain data unnecessary for our purposes and therefore we created a subset called a local map by extracting the relevant data from rosbag files. The final local maps consist of images from the on-board camera and list of image features and its descriptors saved every 1 m of travelled distance. The small amount of data guarantees the repeatability of experiments. The robot used for rosbag collection is shown in figure 4.1b and 4.3 and described in details in the following section.

We divide the datasets into two sets: the training set and the testing set. The training set is used for history collection and spatial model creation which is used for navigation. The training data were collected within five days in the different day phases such as in the morning, afternoon and evening to capture the daylight changes. The testing set consists of 6 drives occurred in three days from 15th to 17th August in different day phases. The figure 4.2 shows the initial robot view and demonstrates the changes.



(a) : Supermap

(b) : 8th August morning

(c) : 8th August evening

(d) : 8th August evening

(e) : 8th August evening

(f) : 9th August afternoon

(g) : 10th August afternoon

(h) : 10th August afternoon

(i) : 10th August afternoon

(j) : 11th August evening

(k) : 11th August evening

(l) : 12th August morning

Figure 4.2: Initial robot views of training drives

4.1 Rosbag

The sensory map created by recording the incoming robot on-board sensory data (i.e. images, odometry etc.) using ROS system is called the "rosbag" file. The saved sensory data provides a possibility to replay the sensory data stream, which would allow us to test the created temporal models and strategies. Unfortunately replaying the rosbag file is processed asynchronously and results of two experiments with the same input could differ and therefore we use rosbag to create local maps to make our experiments repeatable. The order of local maps is guaranteed by its topological linearization.

4.2 Robot description

The robot used for data collection and evaluation is shown in figure 4.3. It is a military robot equipped with an on-board stereo video camera. The robot is able to travel even over a challenging type of terrain due to its continuous tracks. On the other hand, the continuous tracks measure the odometry less precisely than wheels. The dataset for our experiment was collected with the robot which uses BearNav system for navigation, described in chapter 3.

Figure 4.3: Mobile robot used for data collection and experiments.

4.2.1 Collected data

number of used rosbags	17
travelled distance	1020 m
number of lacal maps	5100
weather	varying

Table 4.1: Collected data

4.3 Evaluation method description

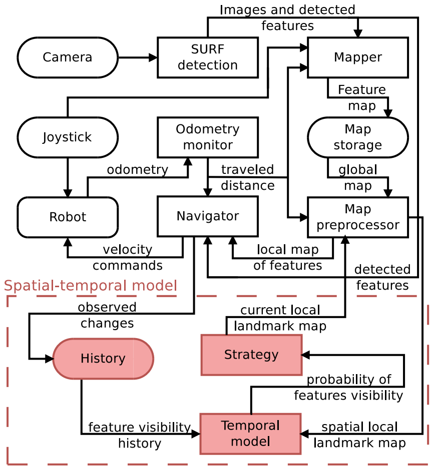
We split the evaluation process into two phases: the training and the evaluation phase. We collect features observation history in the training phase by simulating the robot's movement. Since we focus on the robot directional correction aspect of autonomous navigation, three question arise:

1. How many map features are associated with the current view and what is the ratio of the correct association?
2. How does the calculated direction correction, using the predicted map, differ from the real horizontal shift?
3. How does the created long-term navigation system work on the real mobile robot?

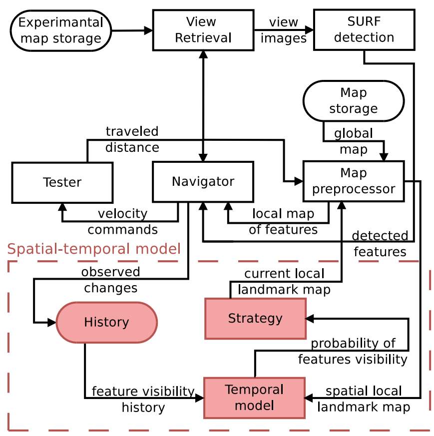
The first two questions can be assessed through statistical methods applied to gathered datasets contrary to the third question which has to be evaluated through the field trial.

4.3.1 Evaluation system implementation

The extended system (figure 4.4a) is used for the experiment evaluation where the robot movement is simulated by a tester node to make the evaluation more time efficient and repeatable. The system structure is shown in figure 4.4b.



(a) : Extended system for mobile robot navigation



(b) : Extended system for data evaluation

Figure 4.4: Navigation system structure

The collected datasets consist of local maps, one for every 1 m of travelled distance storing processed image features and the current view image from on-board video camera. These local maps store image features and current view image from onboard video camera. The evaluation principle is simple because the chosen system for modification uses the directional correction localization. We create a supermap, that stores all features and history of their detection. The supermap can be used for both environment representation and camera simulation. This ensures that the images, which are processed by the navigation pipeline, are relevant to the maps provided by the map preprocesor module that ensures a repetition ability.

4.3.2 Training phase

To create the temporal models, we processed all observations to teach the model the feature behaviour. In the initial phase, the feature filtration is not used to collect states of all possible features. In this phase, we simulate the robot movement and create the history file which is supposed to be used by temporal models to learn the feature probability of visibility. We save the feature ID for identification and its position, size, angle, response and an octave. Whenever the feature state is observed the time of observation occurrence and the state are added to feature values. Each history file line describes one feature history observation. The history file structure is represented by the following table:

feature id	x	y	size	angle	response	octave	time	state	...	time	state
id_1	x_1	y_1	s_1	α_1	r_1	o_1	$t_{1,1}$	$st_{1,1}$	...	$t_{1,j}$	$st_{1,j}$
id_2	x_2	y_2	s_2	α_2	r_2	o_2	$t_{2,1}$	$st_{2,1}$	...	$t_{2,k}$	$st_{2,k}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
id_i	x_i	y_i	s_i	α_i	r_i	o_i	$t_{i,1}$	$st_{i,1}$	...	$t_{2,m}$	$st_{2,m}$

Table 4.2: History file

The history file contains i features. The n-th feature is described by a set of variables: identification id_n and its position x_n and y_n, size s_n, angle α_n, response r_n and an octave o_n. When the feature is served, we save the time of observation $t_{n,m}$ and its state $st_{n,m}$.

Once we have the history of observations, we simulate the runs of testing datasets. The tested datasets are evaluated with the use of multiple spatial-temporal models. We remember the number of matched, correctly matched and incorrectly matched features and the directional correction for each local map and use spatial-temporal model.

4.3.3 Testing phase

We create spatial-temporal models from the created history file according to the model type and parameter and strategy type and parameter. The system then compares the created spatial-temporal models and view images from testing data collection.

In the testing phase we use the previously created map with the feature visibility history. Using a given temporal model and a selection strategy we chose a subset of features and use the subset for robot navigation. Using the aforementioned processing pipeline (figure 4.4b) we can simulate the robot movement as if it was given the using temporal model and selecting strategy. The quality of navigation is evaluated according to aforementioned criteria (section 4.3.1).

4.3.4 Matching features criteria evaluation

The first criterion question that follows from the principle of robot navigation system is *"How many map features are associated with the current view and what is the ratio of the correct feature association?"*. Our goal is to predict the features that are more likely to be associated correctly. This could be evaluated by the *Precision and Recall* metrics [35], however we don't have the information about the false negative labeled features, that were considered not matched but were actually visible. And therefore we are able to calculate only the *precision* which answers the question *"How many selected features are relevant?"* [35], as:

$$r_{c/a} = \frac{n_{correct}}{n_{all_matched}} \tag{4.1}$$

where the $n_{correct}$ is a number of correctly matched features and $n_{all_matched}$ is a total number of matched features.

We add another ratio to see how much greater the number of correctly matched features was the incorrectly matched features number, calculated as:

$$r_{c/i} = \frac{n_{correct}}{n_{incorrect} + 1} \tag{4.2}$$

where the $n_{correct}$ is a number of correctly matched features and $n_{incorrect}$ is a number of incorrectly matched features. To prevent dividing by zero we add +1 to the denominator.

We are well aware of that the these ratios are related but we intent to use them for better results visualization.

4.3.5 Direction correction criteria evaluation

The criterion related to the calculated horizontal shift is associated with the second criteria question and it is the actual accuracy of the robot navigation. We simulate the robot movement and capture the calculated directional correction d_c by the *"histogram voting"* method (section 2.1.5) which is compared to the manually established ground truth. Then we calculate the deviation d of the calculated shift from the real one with the equation:

$$d = |d_r - d_c| \tag{4.3}$$

where d_r is the real horizontal difference, d_c is the calculated one and d is the final deviation.

We display the deviations in graphs that show what is the probability that the error will be lower than the defined deviation value d using certain spatial-temporal model. We also test if the deviations using temporal models are growing or decreasing using statistical T *test*.

Ground truth

The ground truth represents the real directional correction and is created by shifting the local maps images horizontally and comparing them to the appropriate local view. The method that proposes the shift difference is presented presented in [36]. In particular the two images are displayed on the screen with the proposed shift and the human operator is allowed to change the shift if he assumes it was incorrect. The final result is saved into the Ground truth statistics when the human operator confirms the shift.

Paired Samples T test

To test if the calculation of the directional correction was improved by using spatial-temporal models, we use the statistical *t test* of *the Student's distribution*. We have two random variables X and Y, where both random variables consist of the calculated deviations. The X and Y random variables represent deviations calculated using different spatial-temporal models. The size of X and Y is big enough to assume normal distribution due to the Central limit theorem . To calculate the Paired Samples T test, we create a new random variable Z:

$$Z_i = X_i - Y_i \tag{4.4}$$

We set the zero hypothesis so that the mean value of the random variable Z is zero and therefore we test the mean μ of the random variable Z on being equal, lower or greater than zero. Because the variance of the Z distribution is unknown we use the One-sample t test. Because we assume the μ to be zero, we calculate the t statistic with the following equation:

$$t = \frac{\bar{z}}{s_z}\sqrt{n} \tag{4.5}$$

where $\bar{z}$ is realization of the sample mean, n is the sample size and s_z is the sample standard deviation realization of sample calculated as:

$$s_z = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z_n})^2} \tag{4.6}$$

The realization of the sample mean is computed as:

$$s_z = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4.7}$$

The t value is then compared with quantile of Student's distribution with $n-1$ degrees of freedom $q_{t(n-1)}$. The statistical significance is calculated using the distribution function of Student's distribution with $n-1$ degrees of freedom $F_{t(n-1)}$. Because we test if the spatial-temporal models make a change in calculating the directional correction and if so, if the change is positive or negative to the heading correction, we test three hypotheses H_0. The tested hypothesis, rejection conditions and the final significance value is shown in the following table 4.3.

H_0	**rejection condition**	**statistical significance**
$\mu = 0$	$\lvert t\rvert > q_{t(n-1)}(1 - \frac{\alpha}{2})$	$2(1 - F_{t(n-1)}(\lvert t\rvert))$
$\mu \leq 0$	$t > q_{t(n-1)}(1 - \alpha)$	$1 - F_{t(n-1)}(t)$
$\mu \geq 0$	$t < -q_{t(n-1)}(1 - \alpha)$	$F_{t(n-1)}(t)$

Table 4.3: The hypothesis rejection conditions and statistical significance. Countesy of [7]

4.3.6 Field trial evaluation

To prove that the extended navigation system using spatial-temporal models is suitable for mobile robot navigation we provide the practical experiment using a real mobile robot described in section 4.2. We chose several temporal models and several strategies to be used for a practical experiment. Then the robot is navigated autonomously using the modified system (chapter 3) for a long-term autonomous navigation. The trajectory length is 60 m and we evaluate the practical experiment by comparing the travelled distance.

Chapter 5

Experimental Results

The navigation system used for spatial-temporal models evaluation is based on the teach-and-repeat method and the directional correction. The robot simply repeats the motion commands taught by a human operator and corrects its own direction by matching image features stored in local maps to image features extracted from the current view and therefore we use three criteria to test the created spatial-temporal models, mentioned previously in section 4.3.1.

1. How many map features are associated with the current view and what is the ratio of the correct association?
2. How does the calculated direction correction, using the predicted map, differ from the real horizontal shift?
3. How does the created long-term navigation system work on the real mobile robot?

Unlike the third question which has to be evaluated through the field trial, the first two questions can be assessed through statistical methods applied to gathered datasets. To evaluate the first two criteria, we simulate the robot's movement using the real data retrieved almost a year ago and observe the feature matching results. The third method is practical, where the robot uses our extended system for an autonomous navigation. All methods are described in detail in the previous section.

We took the created FreMEn temporal model from the opensource gitHub project FreMEn [10] provided at `https://github.com/strands-project/FreMEn` to demonstrate how easy it is to integrate the new temporal model into the system and to compare our temporal models to the more complex one.

5.1 Feature matching

As we mentioned in the previous section one of the temporal model evaluation methods is to count n_c the number of correctly matched features. The count n_c itself is not a strong identification of the temporal model reliability because of the aforementioned Precision and recall problem. The greater value of features can be infected by the greater value of false positive labels and therefore we compare the result n_c to the rest of matched features. To compare used models we calculate the ratio of correctly matched to incorrectly matched features $r_{c/i}$ and correctly matched to the total number of features $r_{c/a}$. The table 5.1 presents the results of matching feature criteria sorted descending according to $r_{c/i}$ ratio.

5.1.1 Results

The results clearly show that it is better to use a lower number of precise features than a higher number of features. Monte-Carlo does not always provide the consistent result, however it is more likely to provide the correct feature selection than not. We can also see that image feature persistence trained on observation only day old selects submodel that represent the current surroundings better than periodicity. The main outcome is that using temporal models does not impair the ability to match features, it more likely improves it.

5.2 Directional correction

To decide if the spatial-temporal models held the directional correction estimation, we calculated the deviation d of the real and calculated horizontal shift as the square of its difference. We visualize the deviation change by creating a graph, where the x-axis represents the magnitude of deviation and the y-axis represents the probability that the deviation of the calculated horizontal shift will be lower or equal to the deviation value on the x-axis. The resulting graphs are shown in figure 5.1.

T-test results

To verify the deviation change empirically, we calculate the Paired Samples Student's t-test per each pair of tested spatial-temporal model. The principle of this test is that if two models predict heading with similar deviation, the mean μ of the sample of the deviation difference would be equal to zero. If the use of certain temporal model improves the calculation of horizontal shift, the mean of the sample will be either $\mu > 0$ or $\mu < 0$ according to the difference calculation. We use the significance level $\alpha = 0.05$ for our calculations.

The following table 5.2 shows the t-test results and number n_d of models that tested model dominates to and number n_s of models that dominates to the tested one and domination difference δ.

$$\delta = n_d - n_s \tag{5.1}$$

The spatial-temporal models are sorted descending according by δ.

Spatal-temporal model	**correctly matched**	$r_{c/i}$	$r_{c/a}$
Sliding Average 43200 Monte-Carlo 1000	134987	17.594	0.946
Sliding Average 43200 Monte-Carlo 500	63701	15.210	0.938
Sliding Average 43200 Monte-Carlo 250	28188	11.984	0.922
Sum Best 250	8636	2.587	0.721
Weighted Sum 2 Best 250	7984	2.502	0.714
FreMEn Monte-Carlo 1000	36231	2.415	0.707
Sliding Average 43200 Best 250	7761	2.060	0.673
Sum Best 500	12176	1.829	0.646
Weighted Sum 2 Best 500	11231	1.749	0.636
Sliding Average 43200 Best 500	11932	1.698	0.629
Sliding Average 43200 Quantile 0.75	12517	1.575	0.611
FreMEn Best 250	6038	1.295	0.564
Sum Best 1000	16077	1.245	0.554
Sliding Average 43200 Best 1000	15903	1.198	0.545
Weighted Sum 2 Best 1000	14556	1.162	0.537
Sliding Average 43200 Quantile 0.5	16193	1.107	0.525
FreMEn Monte-Carlo 500	8926	1.096	0.523
FreMEn Monte-Carlo 250	4561	1.090	0.521
Sum Quantile 0.5	19319	0.999	0.499
Sum Quantile 0.75	17875	0.980	0.495
FreMEn Best 500	8622	0.977	0.494
Sum Monte-Carlo 250	3517	0.965	0.491
Weighted Sum 2 Quantile 0.5	17342	0.965	0.491
Weighted Sum 2 Quantile 0.75	16459	0.953	0.488
Sliding Average 43200 Quantile 0.25	19665	0.899	0.473
Sum Monte-Carlo 500	6767	0.898	0.473
Sum Monte-Carlo 1000	13054	0.884	0.469
FreMEn Best 1000	14757	0.876	0.467
Weighted Sum 2 Monte-Carlo 250	3198	0.875	0.466
Weighted Sum 2 Monte-Carlo 500	6282	0.842	0.457
Weighted Sum 2 Monte-Carlo 1000	12315	0.832	0.454
FreMEn Quantile 0.5	19673	0.829	0.453
Sum Quantile 0.25	21939	0.815	0.449
Weighted Sum 2 Quantile 0.25	21044	0.806	0.446
FreMEn Quantile 0.75	11398	0.798	0.443
without	24744	0.722	0.419
FreMEn Quantile 0.25	24329	0.714	0.416

Table 5.1: Matching features criteria for all datasets using spatial-temporal models

5.2.1 Results

The graph results support the outcome of the previous experiments that using fewer features improves the directional correction. We can also say that the

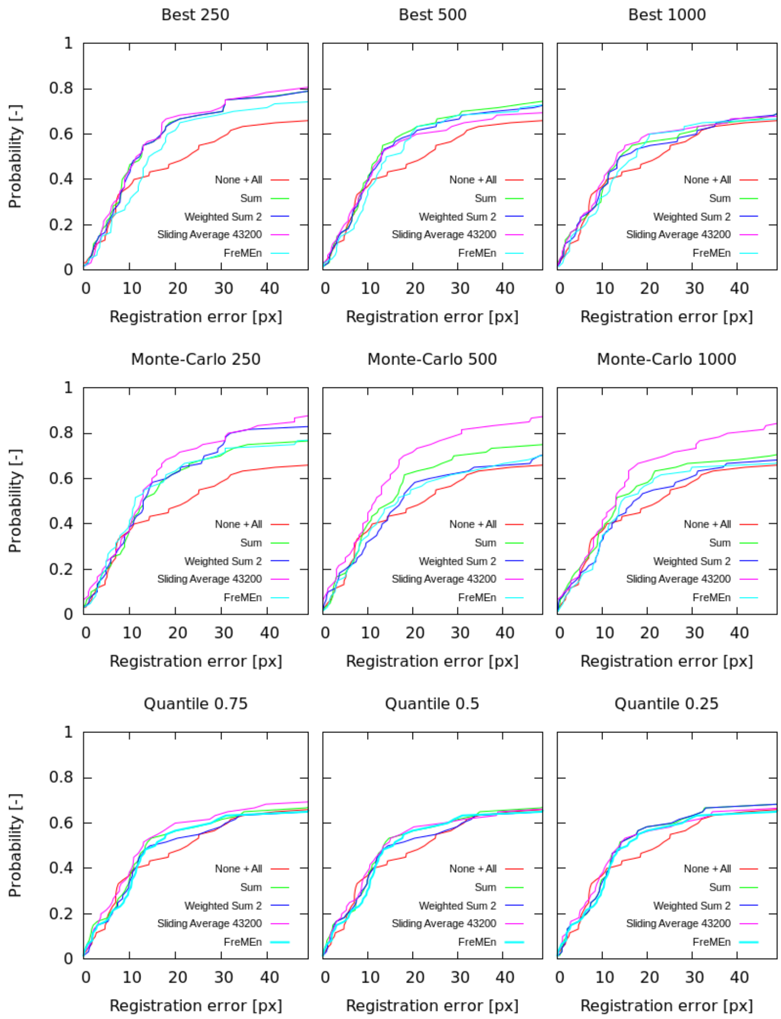


Figure 5.1: The probabilities of deviation according to used spatial-temporal model.

use of appropriate temporal models affects the horizontal shift in a positive manner which is also the outcome of the statistical t-test experiments.

5.3 Field trial

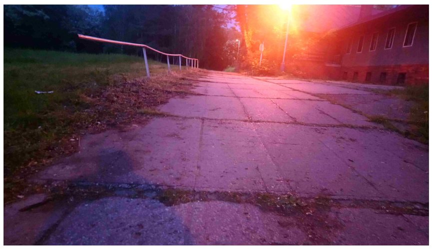
Although we can calculate which spatial-temporal model is the best to use, the real system aspects are not considered, e.g. realtime issues or wrong direction estimation. We integrated the spatial-temporal model usage into functional visual autonomous system BearNav [?]. To test if the integration was successful and the modified system is still functional in field trial, we deployed the modified system for a long-term navigation on the robot. We did experiments with different temporal models. We also let the robot use

Spatal-temporal model	n_s	n_d	δ
Sliding Average 43200 Monte-Carlo 500	0	35	35
Sliding Average 43200 Monte-Carlo 250	0	34	34
Sliding Average 43200 Monte-Carlo 1000	0	33	33
Weighted 2 Best 250	1	30	29
Weighted 2 Monte-Carlo 250	2	28	26
Sum Best 250	3	29	26
Sliding Average 43200 Best 250	3	27	24
Weighted 2 Best 500	4	25	21
Sum Monte-Carlo 250	3	21	18
Sum Best 500	5	21	16
Fremen 0 Best 250	6	16	10
Sliding Average 43200 Quantile 0.75	7	16	9
Sliding Average 43200 Best 500	7	16	9
Sliding Average 43200 Quantile 0.25	9	16	7
Sum Monte-Carlo 1000	8	12	4
Sum Monte-Carlo 500	8	10	2
Sliding Average 43200 Best 1000	8	9	1
Weighted 2 Best 1000	10	9	-1
Sum Best 1000	10	9	-1
Fremen 0 Monte-Carlo 250	9	8	-1
Weighted 2 Monte-Carlo 500	10	8	-2
Sliding Average 43200 Quantile 0.5	10	7	-3
Weighted 2 Quantile 0.75	14	6	-8
Weighted 2 Quantile 0.5	14	6	-8
Sum Quantile 0.75	14	6	-8
Sum Quantile 0.5	14	6	-8
Weighted 2 Quantile 0.25	15	6	-9
Sum Quantile 0.25	15	6	-9
Weighted 2 Monte-Carlo 1000	16	2	-14
Fremen 0 Quantile 0.25	19	1	-18
Fremen 0 Quantile 0.5	21	2	-19
Fremen 0 Best 1000	22	1	-21
without.txt	28	0	-28
Sum 2 Best 1000	28	0	-28
Fremen 0 Monte-Carlo 500	28	0	-28
Fremen 0 Best 500	28	0	-28
Fremen 0 Monte-Carlo 1000	30	0	-30
Fremen 0 Quantile 0.75	32	0	-32

Table 5.2: Results of statistical Paired Sample T test

the original system to compare the difference in travelled distance. The experiments occurred in two different day phases: afternoon and evening. We chose these day phases to capture the appearance changes due to daylight variations. View images from the robot initial positions are shown in the figure 5.2. The figure 5.3 shows the robot and its surroundings during experiments.

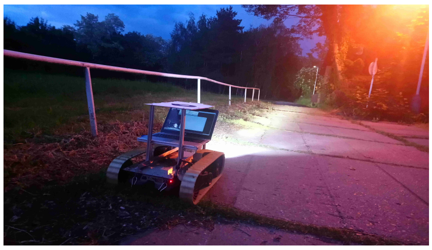
(a) : Initial view in the afternoon



(b) : Initial view in the evening

Figure 5.2: Initial robot views

(a) : The robot at the time of afternoon experiments.



(b) : The robot at the time of evening experiments.

Figure 5.3: The robot during experiments.

The trained trajectory length was 60 m. The robot used local maps from every 1 m, describing the current view of the relevant path section. The experiments occurred on the 15th April 2018 and the weather was ranging from direct sunlight to light rain. The used spatial-temporal models are shown in tables 5.3 and 5.4. The tables show how long the robot travelled autonomously before it got lost. If the travelled distance is 60 m, the robot completed the whole path autonomously.

Temporal model	Selection strategy	Travelled distance
FreMEn	100 Best	60 m
FreMEn	500 Best	35 m
Sliding Average	500 Best	10 m
Sum	500 Best	11 m
Static	All	45 m

Table 5.3: Afternoon runs

Temporal model	Selection strategy	Travelled distance
FreMEn	500 Best	60 m
Sliding Average	Quantile 0.5	9 m
Sum	Quantile 0.5	9 m
Static	All	10 m

Table 5.4: Evening runs

5.3.1 Results

The temporal models were using sparsal maps (1 m apart from each other) while the original static map consisted of local maps every 0.2 m. This might have affected the results in favor of the static model. However the FreMEn model surpassed spatial model and therefore we say that the modified system is possible to be used in practice and an appropriate spatial-temporal model is likely to improve the autonomous navigation. We observe that the Sliding Average temporal model environment representation became obsolete when the history observations had occured almost a year ago. In contrast to the empirical results, the periodic representation selected more reliable environment submap than persistence model Sliding Average.

5.4 Summary

We tested the extended system which uses spatial-temporal models to represent the operational domain, empirically and in the field trial. The empirical experiments assessed the number and ratio of correct feature matching and the impact of the used spatial-temporal models on the directional correction calculation. The field trial experiment was took place almost a year after the training and testing data collection. The empirical experiments clearly verifies that the hypothesis that the use of appropriate spatial-temporal model improves the correct feature matching and directional navigation. It also proved that it is better to use a lower number of precise features then a higher number of features. In other words the great number of unfiltered features are occupied with the noise causing the robot to misleading. The results clearly demonstrate that persistence Sliding Average temporal model represents a solid environmental model when training data are old within days. The FreMEn spatial-temporal model that captures the features periodicity represents the environment reliably, although the training datasets occured long ago. Thus we can say that it is possible to choose the temporal model according to the training data time accuracy.

Chapter 6
Conclusion

This thesis presented an approach to spatial-temporal environment representation for long-term visual navigation and divided the problem of its creation into the two primary subproblems that were introduced at the beginning of this thesis: 1) How should the probability of visibility be calculated? 2) What strategy should we use for correct submap selection? The goal of this thesis was to study the environment evolution to build an authentical interpretation capable of learning and predicting the appearance of surroundings and therefore we suggested multiple types of temporal environment models.

The developed modular flexible system provides a possibility to easily add new spatial-temporal representation and model creation strategies because of the separate abstract modules. Each module solves one primary subproblem and offers an independent study and development of the subproblem. We integrated our spatial-temporal model into an existing functional visual based navigation system BearNav [**?**], that is implemented in Robotic Operating System in C++ programming language.

The created system was tested empirically and practically. The empirical experiments focused on the spatial-temporal model usage. The empirical experiments confirmed the hypothesis that the appropriate environment model with the ability to learn and predict visual changes improves the image feature recognition and thus the visual navigation. The practical experiment verified that the modified system improved the ability to navigate outdoors using maps created almost year ago. The results also bring conclusion on the feature filtration. It was confirmed that the less number of precise features, the better.

We extended the existing functional teach-and-repeat system for a visual mobile robot navigation to use time- and space- dependent operational environment model. We integrated the spatial-temporal model in a way to be easily modified, upgraded or extended by additional spatial-temporal models. We also tested the created model empirically and practically with the use of one verified complex temporal model created within the FreMEn project [10]. We, therefore, are convinced that this thesis accomplished its goal.

Considering the future work on this project first I intend to use a database rather than a file to store the maps and history to make the data collection more efficient. I would like to create new more complex spatial-temporal models and determine the optimal number for selecting features and perform more practical experiments to verify the spatial-temporal model in field trial. The results of these experiments will be presented in [37].

Appendix A
CD Content

In table A.1 are listed names of all root directories on CD

Directory name	Description
bp	bachelor thesis in pdf format.
sources	source codes
dataset	local maps from 17 runs in August 2017

Table A.1: CD Content

Appendix B

Bibliography

[1] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[2] Miroslav Kulich. *Lokalizace a tvorba modelu prostředí v inteligentní robotice*. PhD thesis, CTU, Faculty of Electrical Engineering, 2003.

[3] S. Lowry, N. Sunderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics,*, PP(99):1–19, 2015.

[4] Pavel Macků. Linky metra. `https://pid.cz/wp-content/uploads/mapy/schemata-trvala/a4_metro.png`, November 2017. [Online, Accessed: 30-3-2018].

[5] Antonio R Jimenez, Fernando Seco, Carlos Prieto, and Jorge Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 37–42. IEEE, 2009.

[6] T. Krajník, J. P. Fentanes, J. Santos, and T. Duckett. Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. *IEEE Trans. on Robotics*, 2017.

[7] Mirko Navara. Pravděpodobnost a matematická statistika. `http://cmp.felk.cvut.cz/~navara/stat/PMS_print.pdf`. [Online, Accessed: 21-05-2018].

[8] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, Jun 1991.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[10] T. Krajník, J. Faigl, and V. Vonásek at al. Simple, yet Stable Bearing-only Navigation. *Journal of Field Robotics*, 2010.

[11] P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 2010.

[12] David M Rosen, Julian Mason, and John J Leonard. Towards lifelong feature-based mapping in semi-static environments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1063–1070. IEEE, 2016.

[13] Hana Szücsová. Computer Vision-based Mobile Robot Navigation. Master's thesis, CTU, Faculty of Electrical Engineering, 2011.

[14] Tomáš Krajník. *Large-scale mobile robot navigation and map building*. PhD thesis, CTU, Faculty of Electrical Engineering, 2011.

[15] IMR FEE. 3d outdoor mapping. `https://www.youtube.com/watch?v=9rTkUZ7HV_o`. [Online, Accessed: 1-4-2018].

[16] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[17] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[18] Karel Košnar, Tomáš Krajník, and Libor Přeučil. Visual topological mapping. In *European Robotics Symposium 2008*, pages 333–342. Springer, 2008.

[19] Lisa Zaval and Todd M Gureckis. The impact of perceptual aliasing on exploration and learning in a dynamic decision making task. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

[20] Karel Košnar, Tomáš Krajnık, Vojtech Vonásek, and Libor Pšreušcil. Lama-large maps framework. In *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial*, pages 9–16, 2009.

[21] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An atlas framework for scalable mapping. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1899–1906. IEEE, 2003.

[22] 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 *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4537–4542. IEEE, 2014.

[23] Tomáš Krajník, Pablo Cristóforis, Keerthy Kusumam, Peer Neubert, and Tom Duckett. Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, 88:127–141, 2017.

[24] D. Austin, L. Fletcher, and A. Zelinsky. Mobile robotics in the long term exploring the fourth dimension,. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2:613–618, 2001.

[25] Winston Churchill and Paul Newman. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4525–4532. IEEE, 2012.

[26] Filip Majer, Lucie Halodová, Tomáš Vinter, and Tomáš Krajník. STRoLL BearNav, Simple and Robust Visual Teach-and-replay Navigation System. `https://github.com/gestom/stroll_bearnav`. [Online, Accessed: 23-2-2018].

[27] F. Majer, L. Halodová, and T. Krajník. A precise teach and repeat visual navigation system based on the convergence theorem. In *Student Conference on Planning in Artificial Intelligence and Robotics (PAIR)*, 2017.

[28] James G March. Exploration and exploitation in organizational learning. *Organization science*, 2(1):71–87, 1991.

[29] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, 2012.

[30] Various authors. Robotic operation system. `http://wiki.ros.org/ROS/Introduction`. [Online, Accessed: 13-5-2018].

[31] Agentura ochrany přírody a krajiny ČR. Registr objektů ÚSOP. `http://drusop.nature.cz/ost/chrobjekty/zchru/index.php?frame&SHOW_ONE=1&ID=2212`. [Online, Accessed: 26-4-2018].

[32] Seznam.cz a.s. Seznam maps. `https://en.mapy.cz/`. [Online, Accessed: 16-04-2018].

[33] Tomáš Krajník. Dataset collection on Hostibejk hill. `http://labe.felk.cvut.cz/~tkrajnik/hostibejk_lh_5x.mkv`. [Online, Accessed: 26-4-2018].

[34] Various authors. Rosbags for datasets collection. Hostibejk. `https://drive.google.com/drive/u/1/folders/0B1X-nt7UejUzcWdZUi11bWxNTlE`. [Online, Accessed: 18-10-2017].

[35] Inc. Wikimedia Foundation. Precision and Recall. https://en.wikipedia.org/wiki/Precision_and_recall. [Online, Accessed: 15-05-2018].

[36] Tomáš Krajník, Pablo Cristóforis, Matias Nitsche, Keerthy Kusumam, and Tom Duckett. Image features and seasons revisited. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–7. IEEE, 2015.

[37] Lucie Halodová, Eliška Dvořáková, Filip Majer, Jiří Ulrich, Tomáš Vintr, and Tomáš Krajník. Adaptive image processing methods for outdoor autonomous vehicles.