CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR THESIS

Pavel Linder

Image Processing Methods for Long-Term Teach-and-Repeat Navigation of Mobile Robots

Department of Cybernetics

Thesis supervisor: **George Broughton, MSc.**

May, 2021

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Linder Pavel**	Personal ID number:	**478055**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Image Processing Methods for Long-Term T&R Navigation of Mobile Robots

Bachelor’s thesis title in Czech:

Metody zpracování obrazu pro dlouhodobou T&R navigaci mobilních robotů

Guidelines:

1) Research methods used for teach-and-repeat mobile robot navigation [1], [2].
2) Research image processing methods for navigation in changing environments [3], [4] and [5].
3) Select a set of methods that have the potential to improve the robustness of visual teach-and-repeat navigation in long-term deployments.
4) Select a set of key performance indicators to evaluate robustness of teach-and-repeat navigation.
5) Select suitable datasets to evaluate the robustness.
6) Design and implement a tool capable to automatically evaluate the performance of the individual methods.
7) Perform the experimental evaluation and discuss the results.

Bibliography / sources:

[1] Furgale, P. and Barfoot, T. (2010). Visual Teach and Repeat for Long-Range Rover Autonomy. Journal of Field Robotics.
[2] Krajnik, T. et al. (2010). Simple yet stable bearing-only navigation. Journal of Field Robotics.
[3] Porav, H. et al. (2018). Adversarial Training for Adverse Conditions: Robust Metric Localisation using Appearance Transfer. In ICRA.
[4] Shen, Y. and Wang, Q. (2012). Sky Region Detection in a Single Image for Autonomous Ground Robot Navigation. International Journal of Advanced Robotic Systems.
[5] Lowry, S. (2019). Similarity criteria: evaluating perceptual change for visual localization. In ECMR.

Name and workplace of bachelor’s thesis supervisor:

George Broughton, MSc., Department of Computer Science, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **08.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

George Broughton, MSc.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration

Acknowledgements

I would like to express my gratitude towards my supervisor George Broughton, MSc. and to the head of Chronorobotics Lab, doc. Tomáš Krajník. Without their support, this work could not be created. Further, I would like to thank them for the chance to explore the research environment. I am exceptionally grateful for that.

Abstract

The thesis follows the work on long-term teach-and-repeat navigation used for path following of mobile robots. Teach-and-repeat represents an alternative to SLAM (Simateneously Mapping And Localization) method. BearNav, being an instance of a teach-and-repeat system, includes two phases. The first (teaching) phase involves driving a robot along a path to take images and record control commands used to guide the robot along the path. In the second (repeating) phase, the robot repeats the control commands and corrects the deviation from the path by aligning images. This navigation is sensitive to appearance changes which consequently decreases the robustness of the navigation. Thus, we concentrate on improving the robustness and stability of the pixel-wise image alignment process. By making the image alignment more robust, we make the navigation more robust. There are two methods used to improve the robustness of this process. Firstly, we use low-key image processing techniques to select features that are more stable for the image alignment process. Secondly, we use a Siamese Neural Network to perform the image alignment directly. Neural Networks have been shown to be more robust to appearance changes that happen over time. Hence, the network aims to recognize image features that are more robust and stable. It can be concluded that using a Siamese Neural Network for image alignment increases the navigation's stability.

Keywords: mobile robotics, visual navigation, teach-and-repeat, long-term navigation, neural networks

Abstrakt

Tato práce navazuje na výzkum dlouhodobých 'teach-and-repeat' navigací, které jsou využívány u mobilních robotů s úkolem následování cesty. Tento druh navigací představuje alternativu ke SLAM ('Simateneously Mapping And Localization') systémům. BearNav, který je zástupcem 'teach-and-repeat' navigací, se skládá ze dvou fází. První fáze spočívá v řízení robota podél cesty, kde kontrolní příkazy řízení a snímky pořízené kamerou jsou robotem nahrávány. V druhé fázi robot opakuje nahrané přikazy a koriguje odchylku od naučené cesty pomocí zarovnání snímků ('image alignment'). Tato navigace je citlivá na změny prostředí, které dělají navigaci méně robustní. Práce se proto soustředí na zlepšení robustnosti a stability procesu zarovnání snímků. V této práci jsou předvedeny dvě metody použité k zvýšení robustnosti tohoto procesu. Zaprvé použijeme techniky zpracování obrazu k výběru rysů snímků ('feature selection'), abychom zvýšili robustnost procesu zarovnání snímků. Zadruhé použijeme Siamské Neuronové Sítě k samotnému zarovnání snímků. Bylo prokázáno, že Neuronové sítě jsou odolné dlouhodobějším změnám prostředí. Proto se síť snaží rozeznat ty rysy, které jsou více robustní a stabilní dlouhodobě. Můžeme shrnout, že použitím Siamských Neuronových Sítí pro zarovnání snímků zvyšujeme stabilitu navigace.

Klíčová slova: mobilní robotika, vizuální navigace, teach-and-repeat, dlouhodobá navigace, neuronové sítě

Contents

List of Figures

1 Introduction

1.1 Motivation

Autonomous navigation of mobile robots is currently a very research field impacting both the academic environment and industry. Some of the demands for these systems include working in an unstructured 3D environment of any size. This is quite challenging since the environment also changes over time. Taking an example of intelligent mobile robots that assist people with various everyday tasks operating in indoor environments, such as offices and hospitals. These robots operate in real environments for a long time, needing to be robust to the appearance changes that occur over time. This is discussed in the STRANDS (Spatiotemporal Representations and Activities for Cognitive Control in Long-Term Scenarios) [1] project. For the example of a mobile robot guiding visitors and providing them with information about the residents, navigation must be stable long-term because it adapts to the facility's routines. Therefore it cannot rely on the GPS signal as the signal is not stable enough indoors. Further, it has to be computationally un-expensive and consequently energy-efficient.

The challenges with long-term navigation are even more pronounced in large outdoor environments, especially those including natural elements. Apart from significant illumination and other appearance changes that happen during a day, seasonal changes and weather changes also occur. For outdoor applications, we can mention patrolling robots.

A common requirement for these applications is that they only need to follow a specific path and not finding a path on their own. In this thesis, we concentrate on bearing-only navigation systems for applications similar to those which were mentioned. We show the robot a path which it has to reliably and autonomously follow under different environmental conditions. The path following happens in real-time and long-term.

To reach this task efficiently, we need to estimate the robot's position relative to the path. Some original localization systems use odometry for this task. Odometry measures the traveled distance and rotation of a robot depending on the movement of the wheels. While this works in general conditions in the short-term, it does not work for long-term navigation. Odometry is not suitable for long-term navigation due to its cumulative error caused by wheel slippage.

There are other navigation systems based on sensors like laser or LiDAR, but we concentrate on camera-only localization methods for our purpose. These methods generally build a map which they later use to estimate the robot's position and navigate the robot along the path (direction and speed). There are two main approaches: (a) the mapping is done a priori; (b) SLAM (Simultaneous Localization And Mapping). SLAM uses a map to navigate along the path if it exists. If not, it needs to create a map as it is moving around the path. Simultaneously mapping and localization is a complicated problem as the accuracy of one depends on the other. For instance, we mention ORB-SLAM2 [2], a commonly used SLAM system working with monocular, stereo, and RGB-D cameras. In real-time, it computes the trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments. One of the main drawbacks of SLAM methods is the computational complexity. The methods perform rather complex probabilistic calculations to estimate the pose in real-time. The robots can do such operations, but it takes the computational space that could be used to perform other calculations such as object detection.

Given the above, the methods are complex and relatively slow. Now, we move to another approach, being the teach-and-repeat navigation, such as [3], [4]. Krajnik et al. provided mathematical proving the convergence proof [5]. It states that the robot will eventually converge to the followed path by correcting the errors for an extended period. In contrast with other popular methods, it uses a camera-only metric localization based on pixel-wise image alignment. This method also copes well in highly unstructured outdoor environments. We follow the work done in BearNav [6] which uses a map-and-replay approach to navigate. The robot firstly traverses the path while taking images and recording control commands. Then, it repeats the traversal route according to the recorded commands, and the image alignment is done using feature-matching to correct the robot's displacement from the path. In particular, the robustness of this pixel-wise image alignment is crucial for the whole navigation process. BearNav does not compute complex probabilistic calculations but aligns consecutive images to predict absolute horizontal displacements. Predicted displacements are then used to correct the displacement from the path and set the steering control accordingly. Image alignment is the central aspect as it is fast and straightforward while being sufficient to follow the path. It does not use as much computational space, which enables other operations to be executed simultaneously. To perform the feature-matching, we use commonly used detector and descriptor algorithms, namely SIFT[7], SURF[8], ORB[9], KAZE[10] and A-KAZE[11].

While teach-and-repeat navigation works in general conditions with great accuracy, they face a decrease in performance when dealing with challenging environmental conditions. Environmental changes include weather or seasonal changes, i.e., snow, rain, fog. Further, these also include during-day changes connected with illumination changes. The feature-matching used in the navigation is found to be sensible on a severe appearance change recurring routinely overtime during outdoor traversals. Image key points from a bright sunny day will be much different from a night image, which a LED torch may distort. (as you can see in Figure 1) Usually, there are fewer feature matches in these adverse conditions. Then, the robot may fail at the feature-matching step if it cannot track a minimum number of sparse feature points from the saved image to the currently captured image. [12]

Given the above, the thesis aims to address the problem with adverse conditions. We focus on improving the robustness across traversals with different conditions, including adverse conditions. Nevertheless, we also strive to maintain the same accuracy as the methods we are building upon. The emphasis of the work is not to build a new mobile robot long-term navigation system. It is also not the ultimate goal to create a more accurate version of such a system. We want to improve the robustness and stability of the pixel-wise image alignment used in the BearNav. Again, a robust image alignment is a core element

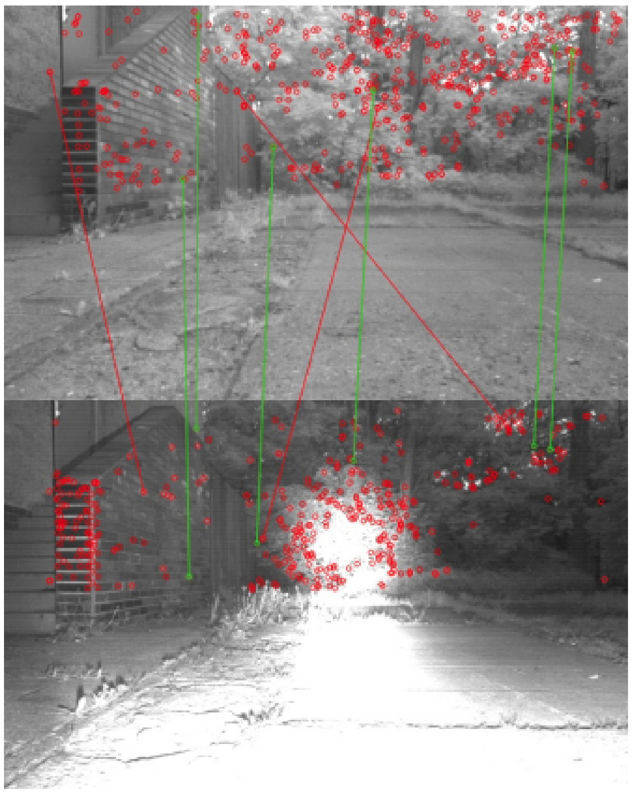
Figure 1: Feature-matching between day (top) and night (bottom) image in the 'Consolidated' dataset. It shows the features are not stable in both conditions and traditional feature-matching methods do not perform well.

for the navigation process.

The first direction we are heading is improving the existing feature-matching procedure used in BearNav. We use image-processing techniques to select more robust features and increase the stability of the predictions. We want to identify those features that should be resilient to appearance changes caused by changes in the environment. Thus, the feature-matching algorithm used in Bearnav [6] is improved by applying heat maps before the prediction itself. We use two different image processing techniques. Firstly, we establish our selection algorithm on Selective search [13]. We compute the per-pixel density of bounding boxes produced by the search. A heat-map is generated as the per-pixel density. Secondly, we use a state-of-the-art sky detector [14] to produce binary maps directly. As the sky is an area occurs in a most outdoor environment while being extremely volatile with seasonal and during-day changes, we discard the image features in the sky region.

We also cover replacing the existing feature-matching procedure from BearNav with a Siamese Neural Network (SNN for short). The Siamese Neural Network takes two (or more) images, puts both of them through Convolutional Neural Networks that share parameters and weight to produce feature vectors. These vectors are usually used to compare the images somehow, showing their similarities or differences in other Siamese networks' applications.

However, we use the Siamese Neural Network to predict the horizontal displacements of two images. By using Neural networks, we are moving to more abstract reasoning. We are free from pixel-to-pixel rationale, which could increase the robustness even more. Therefore, we train the Siamese Neural Network to align two images. We use a dataset containing traversals with adverse conditions to train the network. (more on that in Section 4)

1.2 Hypothesis

Here, we formulate the main questions we cover in the thesis and explain how we will measure their attainment. We strive to improve the robustness of the teach-and-repeat navigation system based on image alignment. They are two hypothesizes which will be covered in this thesis:

1. The robustness of teach-and-repeat navigation increases by selecting features using low-key image processing techniques. **(improved feature-matching image alignment)**
2. The robustness of teach-and-repeat navigation increases by predicting horizontal displacements with a Siamese Neural Network. **(Siamese Neural Network-based image alignment)**

We formulated the main hypothesizes we want to achieve in the thesis. Now, we discuss how we want to verify them. The primary measure we use is a displacement error. As described, we aim to improve the image alignment in the navigation process. We perform several comparisons and visualizations to evaluate the performance of the proposed methods and address the hypotheses. For that, we need to calculate the predicted horizontal displacements for each method.

Firstly, we compare these displacements with the ground truth. We calculate Mean error, standard deviation, and RMSE (Root-Mean-Square Error) for each method. Experiments are done on the 'Consolidated' dataset, which includes several adverse traversals that are very challenging for state-of-the-art navigation systems. It also includes different environments, such as urban and natural ones. This dataset contains hand-annotated ground truth as an absolute horizontal displacement.

Secondly, we compare the displacements with a reference method. As a reference method, we use a feature-matching procedure which is used in BearNav for image alignment. We used feature detectors, such as SIFT [7], SURF [8], KAZE [10], AKAZE [11], ORB [9] to get the features to be matched.

The horizontal displacement error depends mainly on the selected traversal and its conditions - whether the sub-maps were taken at day, night, with the sun, etc. We acquired the mean errors at every position for each traversal. Therefore we can use these to analyze the performance across the dataset. For that reason, we will also calculate the performance of each method for each traversal individually to see how a method is performing under different conditions. Likewise, we show individual positions in a traversal to show which positions are more challenging under different conditions. We should see at which position of the traversal the robot may have a problem to localize the position, and consequently, when the error is too large, the navigation would fail.

To conclude, we compute displacement errors for several methods, including some reference methods. At this point, we want to compare these methods using statistical tests. We could use the ANOVA test for the reason as it enables us to compare more than two methods. Formulating the null hypothesis that the methods have the same performance and the alternative hypothesis that some of the performance is better than another. The problem here is that we could not establish which one is the best method. We could only state there are some differences.

Therefore, we pick the best method for our proposed methods and the reference methods by comparing their mean errors. Then, statistical tests are executed between the best proposed and the best reference method to validate if a method performs significantly better over the other one. Typically, paired t-test would be executed. This test has two main assumptions. The data are normally distributed and homoscedastic. Firstly, we test these assumptions and prove the displacement errors do not meet these assumptions. Therefore t-test cannot be used. As a result, we select and perform a paired non-parametric test, being the Wilcoxon signed-rank test.

The motivation and evaluation of the proposed methods were covered. At last, we explain how we perform the experiments. We use the 'Consolidated' dataset for the experiments. (see Section 4) For the first method (improved feature-matching), we generate heat maps by selecting important features which may be robust to appearance changes. A state-of-the-art sky detector and a modification of Selective Search were used to get these heat maps. Consequently, we apply these heat maps to the original images (those the maps were built upon) and perform a feature-matching algorithm to align the images. After matching the features, the procedure builds a sliding histogram of predicted displacements with the individual matched features. We use the peak of such histogram to select the predicted horizontal displacement.

The second method does not use the feature-matching procedure at all. Instead, it takes a Siamese Neural Network trained to align the images and predict the correct displacement correctly. As a result, we get similarity scores for different offsets. We perform some transformations on the similarity scores data to get rid of local maxima and imprecise measurements. In the end, we take the peak of this transformed data to obtain the resulting predicted offset.

1.3 Organisation

The thesis is organized into six sections, the first being the Introduction. The Second Section 'Related Work' gives an overview of mobile robot localization, including SLAM and teach-and-repeat navigation. Research made on methods coping with the issues raising with adverse conditions was also concluded in this section.

The Third Section ‘System description’ gives a detailed description of our proposed methods. Implementation details, network architecture, and other procedures are explained here.

The Fourth Section ‘Datasets’ describes the ‘Consolidated’ dataset in detail.

The Fifth Section ‘Experiments’ describes the conducted experiments. The testing algorithm is presented, then the quantitative results are shown, and their discussion is concluded.

The thesis is concluded with the Fifth Section ‘Conclusion’.

2 Related Work

Mobile robots and their navigation is a problem that is here for some time. In 1986, Smith et al. [15] address the problem of uncertainty in an environment and measurements. They set the mathematical and probabilistic foundations to allow the robots to move in an unknown environment. Leonard and Durrant-Whyte [16] then introduce simultaneous map building and localization as a new essential problem for mobile robots. It was stated as a long-term globally referenced position estimation without a priori information. Also, they called this the 'Chicken and egg problem' paradox since for a robot to move accurately, it needs an accurate map, but, at the same time, the environment mapping needs a precise localization. Given the above, it is necessary to perform both of the tasks simultaneously. The authors then made a proof of concept formalized as Simultaneous Localisation and Mapping (SLAM). SLAM was then formulated as a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. In SLAM, both the platform's trajectory and the location of all landmarks are estimated online without the need for a priori location's knowledge. It is worth mentioning that the solution theoretically existed at the time, but the applications were time exhaustive for real-life applications.

Statistical independence is the mandatory requirement to cope with metric bias and with noise in measurements. Laser SLAM systems or LiDAR SLAM systems were developed, including GMapping [17]. This laser-based system builds a map, and as the robot moves, a Kalman filter is used to estimate the robot's position and then correct the map.

Later, in 2005, VSLAM (visual SLAM) systems started being developed using primarily camera sensors because of the increasing ubiquity of cameras such as those in mobile devices. Karlsson et al. [18] developed a novel algorithm, which is vision and odometry-based and enables low-cost navigation in cluttered and populated environments. Other example using camera as a primar sensor include [19], [20]. In 2017, Mur-Artal et al. [2] developed ORB-SLAM. ORB-SLAM uses monocular, stereo, or RGB-D cameras to estimate the real-time trajectory and build a map of the environment.

SLAM systems were developed for 30 years, and they are capable of accurate and robust localization. SLAM is used to build a precise map of the robot's surroundings and then use the map to find the position. Cadena et al. [21] point out the problem with the robustness and scalability of SLAM. An experiment [22] was performed for ORB-SLAM2, which once again concludes with the finding that SLAM systems are not robust to adverse conditions. Another problem of SLAM is its computational complexity but also the fact that the system is rather complicated. SLAM maps the whole environment, but there are many applications where we only need to follow a particular path. These applications include patrolling robots. For this reason, teach-and-repeat navigation systems were created. Furgale et al. [4] developed a system using a stereo camera which they tested in a highly unstructured 3D environment without a GPS signal. This system is based on repeating a learned path by building maps and using visual odometry combined with RANSAC. They showed the possibility to repeat long traversals without the need for an accurate global reconstruction. This was essential for developing similar systems. In 2010, Krajnik et al. [5] provided mathematical proof proving the convergence proof. This formulates that the robot will eventually converge to the following path by correcting the errors for an extended period. Bearing-only systems are therefore entirely sufficient, and what is more, they require much less computational complexity as they are based on a simple image alignment. BearNav navigation [6] is fully described in Section 3.

Nonetheless, as the BearNav experiments tell, all of these methods are not robust to adverse conditions. The errors differed throughout seasons, and there were fewer correspondences in adverse conditions. The BearNav uses a feature-matching algorithm to align two images. To do so, firstly, we need to extract and describe the features. As the features change drastically in an outdoor environment (especially in nature) when operating long-term, this step naturally brings a problem. Austin et al. [12] then discuss the reliability of mobile robots in real life in the long-term in a paper. Lowry et al. [23] evaluate the perceptual change for a visual localization by defining similarity criteria that reflect the ability of the image descriptor to perform visual localization successfully. Other papers, such as [24] described the technical issues under adverse weather conditions such as sun glare, rain, fog, and snow. They also stated why these conditions could be a problem with the application. Further, there are also during-day changes that affect the features significantly. The main issue is the illumination change as some features which occur under direct light do not display at night.

To detect and describe the features, an algorithm such as Scale Invariant Feature Transform (SIFT) [7] or Speeded-up Robust Features (SURF) [8] is used. As these are not resilient to seasonal and illumination changes, other feature descriptors were researched to increase the robustness. In 2017, Krajnik et al. [25] developed an algorithm that was created to get more robust features by training them. A trainable feature descriptor, called GRIEF, is proposed and tested against other commonly used descriptors with promising results.

Adaptive methods are another approach to increase robustness. Works, such as [26], [27], [28], [29] use map adaptation and advanced map management techniques. The process of such adaptation is to select features useful for the navigation task, remove obsolete features, and add the features currently taken by a camera. The maps are therefore updated as the robot is moving through the environment. The main drawback is that the robot needs to acquire many training data with different conditions. Another negative aspect is that adaptive mapping does not work when the environment's conditions change quickly, which is very common in real-life applications.

In 2014, Lowry et al. [30] address the stability issues occurring under substantial appearance change by using Neural Networks to transform the domain. Simply put, the network takes the image and transforms it into another time of the day. It estimates the current time and given the appearance of the position in the past with knowledge of appearance change over time. In 2018, Porav et al. [31] introduced a method using Generative Adversarial Networks with SURF descriptor and SURF detector to improve metric localization under adverse conditions. While working well, it is computationally exhaustive, and it is needed to specify the conditions for the appearance transformation explicitly.

For some applications, it may be challenging to identify the current conditions. Therefore the methods mentioned above do not work. In 2017, DeTone et al. introduced SuperPoint [32], another method introducing a more robust feature detector and descriptor. Their fully-convolutional model operates on full-sized images and computes pixel-level interest point locations and associated descriptors in one forward pass. In 2018, Sarli et al. [33] used a monolithic Convolutional Neural Network that simultaneously predicts local features and global descriptors for localization. First, they perform a global retrieval to obtain location estimation and only later match local features within those candidate places.

Given the above, in this thesis, we use the abilities of Neural Networks to cope with adverse conditions in another way. We will straightforwardly train the Neural Network to perform an alignment of two images. Melekhov et al. [34] developed a Siamese Neural Network for image matching. They find the matching and non-matching pairs of images by representing them with Neural Network-based feature vectors, whose similarity is measured by Euclidean distance. One proposed method of the thesis lies in this very idea. We propose a similar Siamese Network, which produces a similarity score on a pair of images to predict an absolute horizontal displacement. The robot consequently uses the aforementioned horizontal displacement to correct its movement as defined in BearNav. By the nature of Neural Networks, they are meant to build more abstract image features. Thus, it should be a more stable method that should be robust to any appearance changes.

3 System description

In this section, we will firstly introduce the overall teach-and-repeat navigation process. We will concentrate on the camera-only feature-matching-based navigation used in teach-and-repeat navigation systems such as BearNav [6]. The main contribution of this thesis lies in improving the existing image alignment process used to predict horizontal displacements. The first method takes the feature-matching algorithm used in BearNav and combines it with two different feature-selection methods. The feature selection is made by applying heat maps generated from the original images. The way how these heat maps are generated will be explained in detail. The second method uses a different approach by building a Siamese Neural Network. We will show and explain how the implementation detail of the Siamese Neural Network model and how the architecture was built. We will also discuss individual functions and modules.

Teach-and-repeat navigation The effort of teach-and-repeat systems is to reliably navigate a mobile robot along a path in a 2D or 3D unstructured environment of any size. Additionally, it should satisfy real-time constraints and operate long-term, meaning it should be robust to different, often adverse, conditions. This section explains the overall architecture of such systems and indicates where we extend the system. Further, we discuss where the main contribution of the thesis is.

3.1 BearNav Navigation System

One of the implementations of teach-and-repeat navigation systems is called BearNav. The objective of this thesis is built upon this navigation system. There are several versions of this system. such as [5], [35]. [6] BearNav does not require a calibrated camera and does not rely on the environment structure. The complexity of this method is also scalable, meaning it does not depend on the environment size. An essential aspect of this navigation algorithm is that it does not need to explicitly localize the robot or create a **three-dimensional** map of detected landmarks which lowers the computational complexity. By aligning two images, the robot estimates the error of the robot's heading from the followed path. This estimation is then used to control the steering motion and adjust the robot's heading. It should also be noted that the proposed method works in real-time, long-term, and even outdoors. [29]

3.1.1 Map and Replay Navigation

The proposed navigation procedure is based on the map-and-replay technique. The idea is simple. Firstly, the robot records its control commands and captures images from a camera along the path. Secondly, the robot is put at the start of the learned path. The images and the recorded control commands are used along with the robot's camera to follow the learned path. The algorithm also uses odometry. Odometry measures the traveled distance by measuring the wheel movement. This may be accurate in the short run, but it is not reliable long-term because of the cumulative odometric error caused by wheel slippage. The navigation process is described below.

Navigation strategy (for a robot driven by a joystick)

1. Mapping (Learning) phase
   (a) The robot is driven along a path and records its movement (joystick commands)
   (b) The camera captures images at fixed distances (i.e., 30cm or 50cm) using odometry
2. Replay phase
   (a) The robot repeats the joystick commands from the learning phase
   (b) It loads an image from the learning phase at a position determined by odometry
   (c) It performs image alignment with the current camera's image and the loaded image to predict a displacement error
   (d) It combines the error with the recorded joystick commands to adjust the steering motion and follow the learned path

3.1.2 Feature-matching in the Image Alignment

In both the Mapping and the Replay phase, BearNav performs a feature-matching algorithm. SURF [8] descriptor was used due to its low computational complexity and good overall performance at BearNav. Other standard detection and descriptor algorithms can be used, such as BRIEF, SIFT, BRISK, KAZE, and AKAZE. For completeness and to increase the performance, we decided to perform experiments with all of the mentioned algorithms.

Increasing the robustness of image features BearNav and other navigation systems do not operate well under adverse conditions. With a different illumination caused by seasonal, weather, or during-day changes, the performance of these methods decreases significantly. This suggests that by implementing a different feature-matching algorithm that is more robust against these adverse conditions, the whole performance of BearNav could be improved.

3.2 Extending BearNav by Generating Heat Maps

Producing different maps The heat maps and binary maps are produced to improve the performance of the feature-matching. In the standard feature-matching algorithm, all features have the same weight, meaning they are given the same importance. We want to create a heat map to highlight certain parts of an image that might be more important and robust to appearance changes. There are two methods used for the purpose. Each method uses a different approach to create the heat maps.

1. The Selective Search produces a continuous heat map which indicates how important a pixel is (where 0 indicates a pixel which is completely insignificant and 1 is very important).

2. The sky detector produces a binary map that detects a sky (where 0 indicates a sky and 1 indicates the rest of the image).

3.2.1 Selective Search

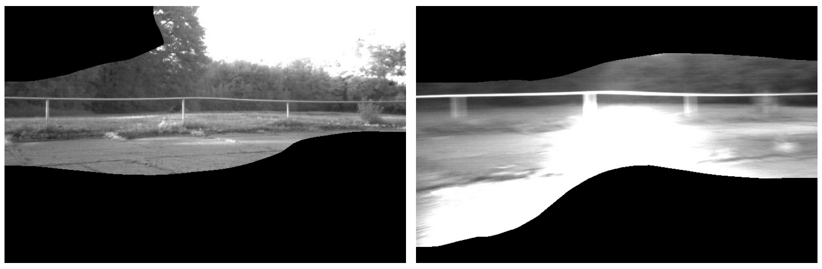

Figure 2: Heat maps produces by Selective Search at day (left) and at night (right).

This method proposes the use of the Selective Search [13] to find areas of interest of an image. We do that to give more precedence to the image features from these regions. This stems from the assumption that these regions are generally candidate ‘objects of interest’ for neural networks or other methods used with Selective Search. Therefore, these regions are more likely to be navigable objects rather than ‘background clutter.’
OpenCV implementation of Selective Search was used. Similarity criterias can be set to color similarity, size similarity, fill similarity, or texture similarity. A combination of all of these similarities results in the following function S, which takes two regions and return the similarity score between them. We used the function in our system.

$$S_{r_i,r_j} = s_{color}(r_i, r_j) + s_{texture}(r_i, r_j) + s_{color}(r_i, r_j) + s_{size}(r_i, r_j) + s_{fill}(r_i, r_j) \tag{1}$$

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min({c_i}^k, {c_j}^k) \tag{2}$$

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min({t_i}^k, {t_j}^k) \tag{3}$$

$$s_{size}(r_i, r_j) = 1 - (size(r_i) + size(r_j)) \div size(img) \tag{4}$$

$$s_{fill}(r_i, r_j) = 1 - (size(BB_{ij} - size(r_i) - size(r_j)) \div size(img)) \tag{5}$$

where:

${c_i}^k, {c_j}^k$ is k^{th} value of histogram bin of region r_i and r_j

${t_i}^k, {t_j}^k$ is k^{th} value of texture histogram bin of region r_i and r_j

$size(r_i)$, $size(r_j)$ and $size(img)$ are the sizes or regions r_i, r_j and image in pixels

$size(BB)_{ij}$ is the size of the bounding box around i and j

Greedy algorithm for Selective Search:

1. From set of regions, set two which have the biggest similarity S.
2. Combine them into one region.
3. Repeat step 1 and 2 for a set amount of iterations.

The algorithm returns a set of bounding boxes that may be used for further object detections. At this step, instead of running a complex model, the density of the binding boxes is computed to produce the heat map.

Algorithm using bounding boxes to produce heat maps:

1. Run Selective Search to produce bounding boxes.
2. Compute the number of bounding boxes for each pixel to get the density.
3. Use Gaussian blur on the heat maps.
4. Normalise the values between 0 and 1.

The values of the heat maps are not binary but continuous and range between 0 and 1. (inclusive) These heat maps were produced with different levels of Gaussian blur, including no blur. Later, we tuned the level of the blur to get the best performance.

3.2.2 Using Sky detector

State-of-the-art implementation of a sky detector by [14] was used. This detector uses gradient information from a single image, then the optimal segmentation threshold is obtained. Further, it also includes some post-processing methods to refine the regions.

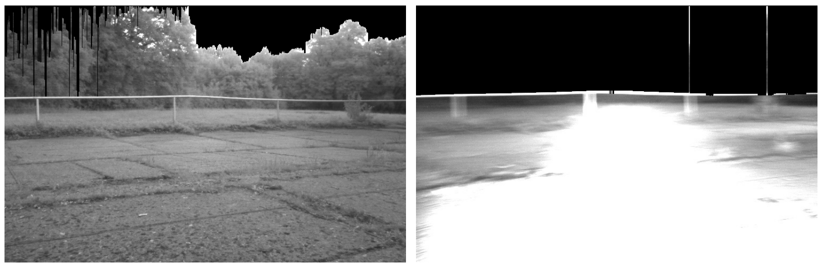

Figure 3: Images where the sky is segmented by binary maps (from the sky detectory) in day (left) and night (right).

Sky detector algorithm to produce binary masks:

1. Converts to grayscale. (if it is not already)
2. Computes Sobel operator to approximate the (absolute gradient) magnitude.
3. Computes energy based on image values.
4. Optimizes the thresholds for gradients.
5. Makes a borderline where energy function is a fit function.
6. Discard sky detection when there is no sky in the image. (too small or zig-zag pattern)

Using this detector on a series of images produces binary masks, which indicate where the sky is. The idea here is that we want to discard the sky as a region that is sensitive to time and seasonal domain. The sky usually does not produce many features, but it does under adverse conditions, such as a cloudy sky.

3.3 Extending BearNav by Building a Siamese Neural Network

Siamese Neural Network A Siamese Neural Network [34] includes two or more Convolutional Neural Networks (CNN) of the same architecture, parameters, and weights. Any parameter updates are mirrored across both subnetworks. Usually, a Siamese Neural Network (SNN) is used to detect either similarities or changes of a pair of images. For our purposes, the Siamese network architecture contains two parallel streams to estimate the similarity between two inputs and learn their discriminative features. In addition, it uses a pre-trained CNN as a feature extractor to get a dense higher-order feature vector for both images. It is important to note that both images go through the same network in the feature extraction stage. This radically speeds up the learning process, and the whole SNN is trained ‘only’ to detect similarities of some features.
We utilize labeled training image pairs to learn an image-level feature representation so that similar images are mapped close to each other in the feature space, and dissimilar image pairs are mapped far from each other.

3.3.1 Network Architecture

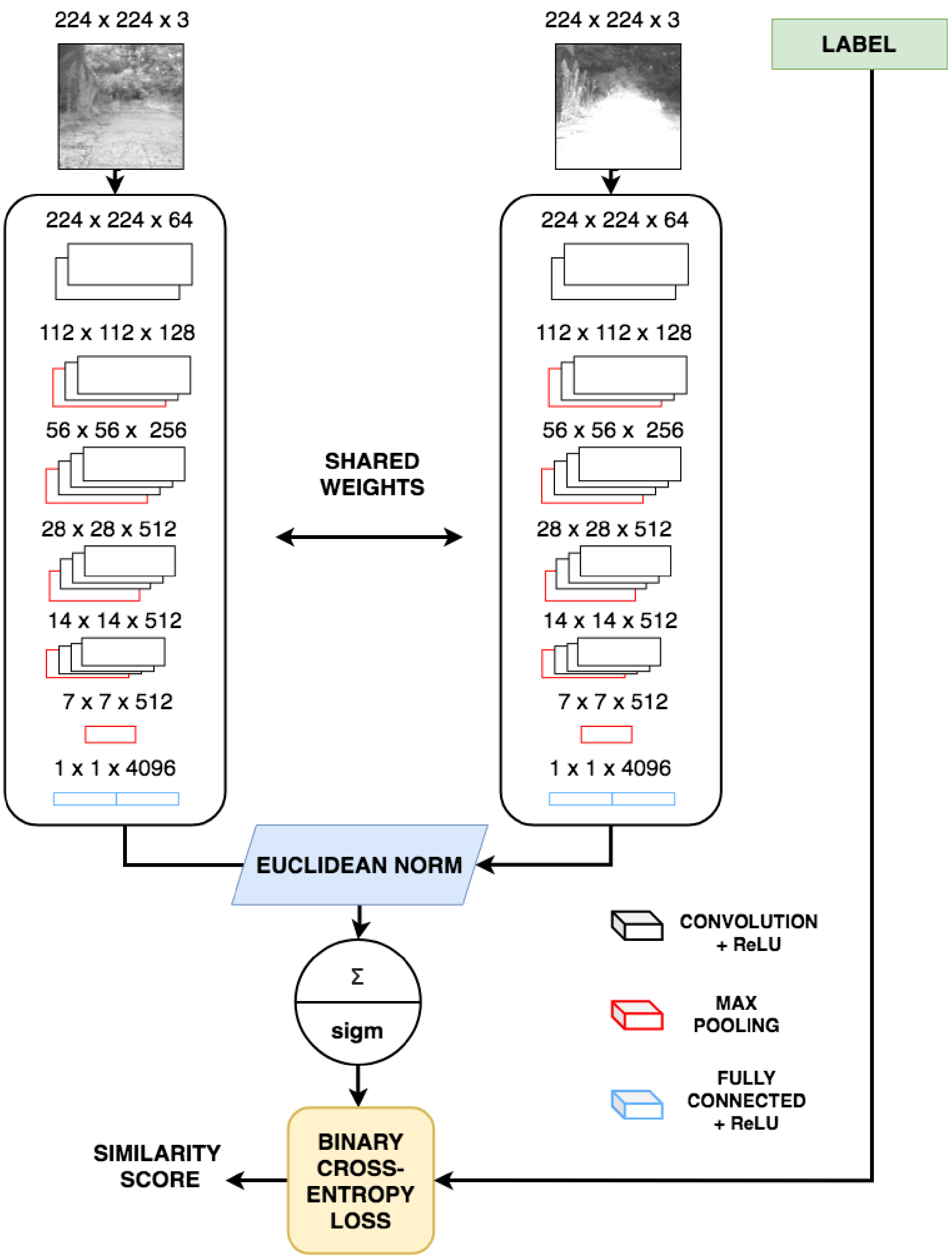

Figure 4: The Siamese Network architecture, including VGG16 architecture.

The network's architecture is visualized in Figure 4. The input for the network is an image pair from the robot's traversal. These images are preprocess using Keras VGG16 function to be compatible with the pre-trained VGG16 network. It follows that both images are fed into a VGG16 subnetwork which shares the same weight and parameters. This property goes straight from the Siamese Neural Network definition. The VGG16 network architecture is described more thoroughly in the next paragraph. The output of VGG16 is a dense feature vector. Euclidean norm (distance) is used with these two feature vectors to measure the distance of the vectors. Consequently, the sigmoid activation function is used to estimate the similarity score for the pair of images. Ultimately, the Binary Cross-Entropy loss compares each of the predicted probabilities to the correct alignment label. It classifies a pair of images either correctly aligned 1, or not correctly aligned 0. The label is given along with the input pair. Then, it back-propagates the error and sets the network's weights accordingly.

Using pre-trained CNN To distinguish the images from each other, we need to extract the features at first. We do that by using a deep Convolutional Neural Network (CNN). We use one of the most used implementations by K. Simonyan, and A. Zisserman from the University of Oxford [36]. VGG16 uses very small 3 x 3 convolution filters. It achieves very good accuracy with 16 weight layers.

VGG16 Architecture The image is passed through a stack of 3 x 3 convolutional layers, with 1-pixel striding and 1-pixel padding. Spatial pooling is done by five max-pooling layers over a 2 x 2 pixel window, with 2-pixel striding. Three Fully-Connected layers follow the stack of convolutional and max-pooling layers. The first two of them have 4096 channels, and the third contains 1000 channels. We do not use the last Fully-Connected layer for our model, and we feed the 4096 sized vectors directly to the Euclidean norm function. All convolutional layers contain ReLU to achieve the non-linearity of the layers.

Pre-trained VGG16 The model was trained on the ImageNet dataset but according to [36], VGG16 is expected to work on other datasets. We use Keras's implementation of the VGG16 trained on ImageNet. This should provide a good feature description when using it for the Siamese Neural Network.

Euclidean distance $d\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2}$ where p,d are points given by Cartesian coordinates in n-dimensional Euclidean space.

3.3.2 Processing Training Data

Network Input Pipeline To summarise, we feed two images into the network, and the network returns a similarity score. What we need is to predict a displacement in between two images. This is used in the navigation process to correct the heading of the robot. For this reason, we take need to generate the pairs from the robot's traversal. The dataset is explained in detail in Section 4.

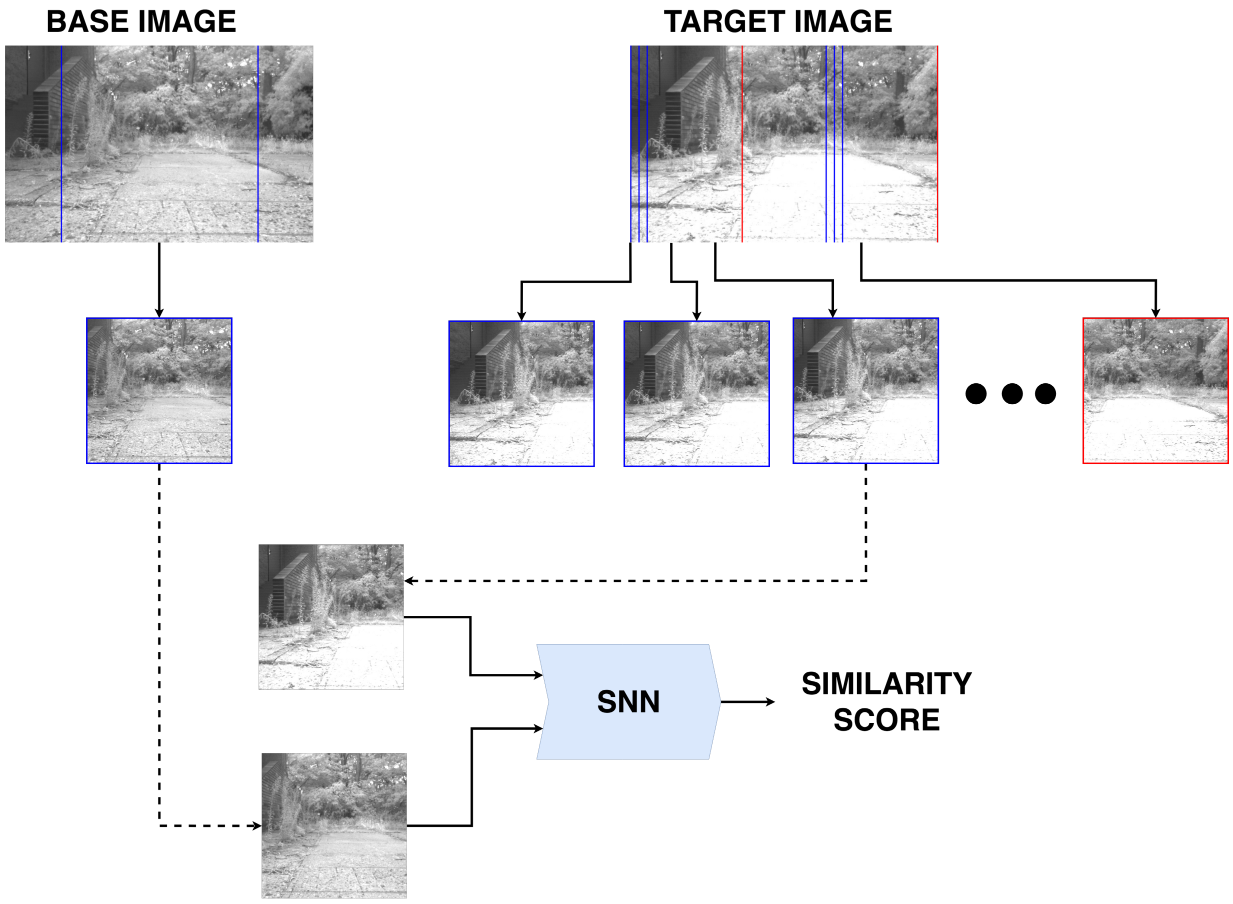

Figure 5: Here you can see how the image pairs are generated in order to predict a correct displacement. There are four image slices displayed with different displacements: 0px, 20px, 40px and 272px.

Generating pairs The input for a Siamese Neural Network is a pair of images connected with a label. The label is a binary value telling whether a pair is correctly aligned (1) or not (0).

In the navigation system, the pair would consist of a saved landmark (from the Learning phase) and the current frame taken by the robot's camera. Consequently, we need to train the network to be able to establish a correct alignment. There are two terms used in the following text: *base image* and *target image*. The base image is always the image obtained in the Learning phase. The target image is a current frame that is then sliced to predict the displacement. *Height* is then the height of an image, and *Width* is the image's width. We only take the center part of the base image. We crop a square that maximizes the height of the image, meaning the size of the resulting cropped image is *Height* x *Height*. At this moment, we have the base image with *Height* x *Height* dimensions and the target image with dimensions *Width* x *Height*.

Slicing the target image Further, we slice the target image to label the correct alignment. Similarly, as with the base image, the left-most *Height* x *Height* square is cropped from the original target image. The same is done repeatedly while moving the slicing window with a step equal to the offset size. See Figure 5 for better visualization. For each position in the traversal, we take the center part of the base image and generate labeled pairs describing whether the pair is aligned correctly or not. There are *(Width-Height) ÷ offsetSize* slices generated for every target image.

Labeling the correct aligment We acquired a ground truth for the dataset. The ground truth contains an absolute horizontal displacement between an image in a traversal and the base image. To identify the correct alignment, we get the ground truth for the target image - $targetGT$. The current slice has offset *currentOffset*, and we compute Manhattan distance to identify if the pair is correctly aligned or not:

$|\mathbf{targetGT} - \mathbf{currentOffset} * \mathbf{offsetSize}| < \mathbf{threshold}$ where *offsetSize* is the model's parameter and *threshold* is a value which tells how precise the offset detection can be.

3.3.3 Training Models

Siamese Network Implementation and Parameters TensorFlow and Keras implementation are used. All the parts are implemented in Python. The implementation also uses OpenCV and NumPy packages. There is a list of the network's parameters:

1. **Parameters which are constant (same for all models):**
    - Batch size: 64
    - Number of epochs: 80
    - Loss Function: Binary Cross-Entropy
    - Optimizer: Adam
    - Used metrics: accuracy
    - The use of validation dataset: Validation split 0.2 was used (meaning 20% of training data was used for validation)
2. **Model's specific constants and parameters:**
    - Input for the SNN: Selection of dataset traversals with their ground truth

- Offset size: 2px and 5px
- Choice of base traversal: A000 and A002 from the 'Consolidated' dataset
- Processing function for the resulting similarities: none, spline, gauss and top-hat transform

Different models were built based on the parameter choice. A000 and A002 traversals were used as the base traversals. These were selected as they contain the standard conditions, being taken during the day, without sun or other seasonal or weather change. There are also other traversals with the same conditions in the dataset, but they contain images where the human operators were present. Accuracy was used as the main metric due to its straightforward interpretability. It tells us the proportion of the image pairs that were aligned correctly among all pairs. We used a validation split to address possible overfitting. In Figure 6, the behavior of two of the models during the training is shown. Both the training and validation accuracy stays high (around 96%). Most of the image pairs are negative, meaning they are not aligned correctly. Since we want to identify the positive pairs, meaning they are aligned correctly, we did not balance the training data anyhow. For this reason, the accuracy stays almost constant during the training. Conversely, the training loss and the validation loss are slowly lowering (< 0.1). However, for both of the displayed models, the validation loss increased at a point, and no significant improvement is reached.

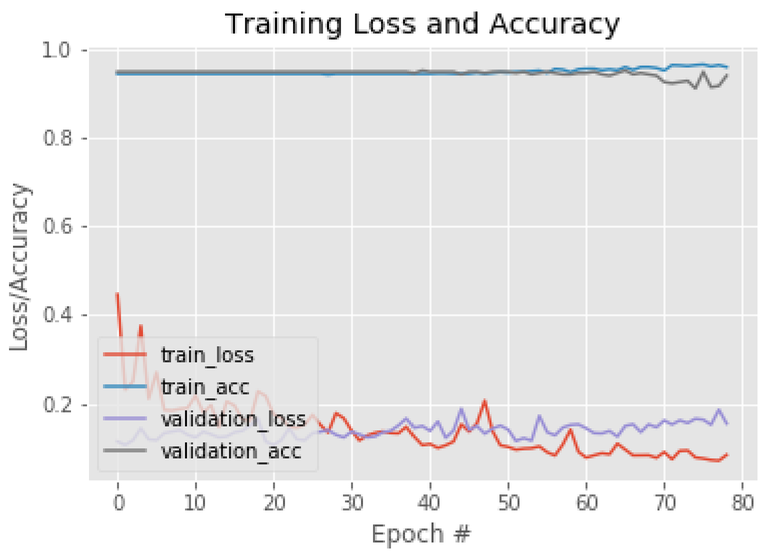

Figure 6: Siamese Neural Network training plot for the model with 5px offset size trained only on day traversal.

Hardware - RCI Cluster The RCI Cluster was used to train the model. Therefore, we acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

4 Datasets

For the experiments, the ‘Consolidated’ dataset from the paper [29] was used. Here, we explain how the dataset was gathered, its contents, and how it is organized. We also discuss the choice as the dataset at the end of this section.

4.1 Selection of the dataset

Nordland Railway dataset [37] is commonly used to test the performance of long-term navigation. The Nordland dataset was taken from a train, so it does not contain translation or rotation. Oxford RobotCar dataset [38] is also commonly used for the purpose. This dataset was taken by a car, so it sticks to a single lane. Therefore, the rotational and translational differences are small. Further, the Oxford dataset contains mainly images of a city. Navigations for the urban environment, such as [39] were successfully implemented. However, other outdoor environments, including nature, were found more challenging for the navigations. In the thesis, we want to test the robustness of the navigation. For this purpose, we want to examine different kinds of outdoor environments, including nature and urban environments. Further, we want to test long-term navigation, so the dataset should include some seasonal or during-day changes. Finally, we also want to address possible issues of the navigation with rotations or translations. Given the above, we selected the ‘Consolidated’ dataset, which meets all these requirements.

4.2 Getting the dataset

This dataset was acquired by authors of [29]. A robot gathered the dataset following a path in a small forest park with one building. The data gathering happened at Hostibejk Hill in Kralupy nad Vltavou, Czechia. One traversal has around 30 meters in length, and the robot autonomously completed 179 of those traversals.

Robot equipment and configuration The dataset was acquired by CAMELEON ECA tracked robot. Images were taken by the left camera of the e-Con TARA stereo camera. The robot was equipped with a 4000 Lumen LED torch which was turned on during the night passes. This is very important for the image features since they are very distorted by artificial light.

4.3 Dataset description and properties

A big part of the pathway goes along a building. Therefore, almost one-third of the traversals contain the building in them. Along the traversal, there are also other subjects, i.e., trees, bushes, railing, and other visible structures. The robot captured the traversal

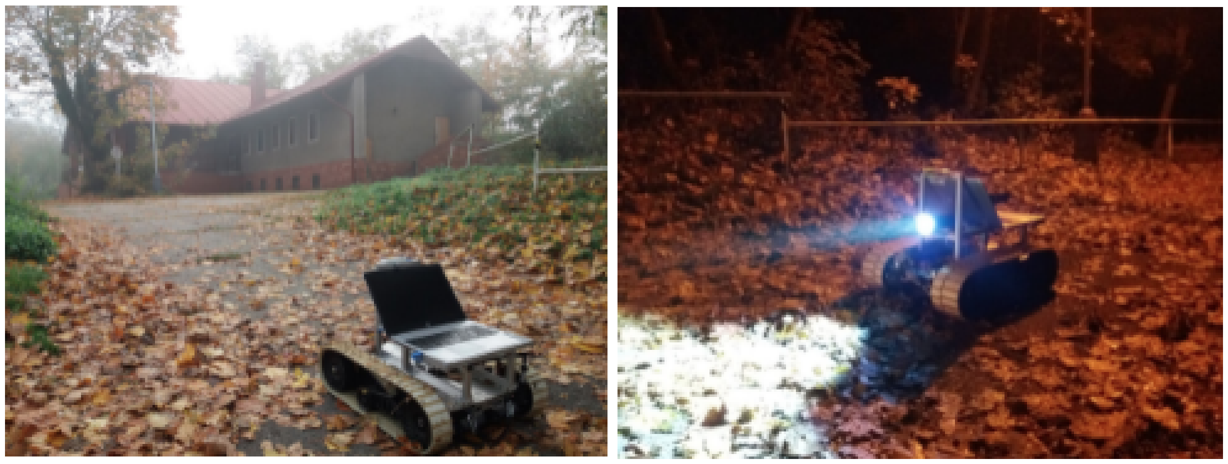

Figure 7: CAMELEON ECA robot aquiring the **'Consolidated' dataset** during the day (left) and during the night (right).

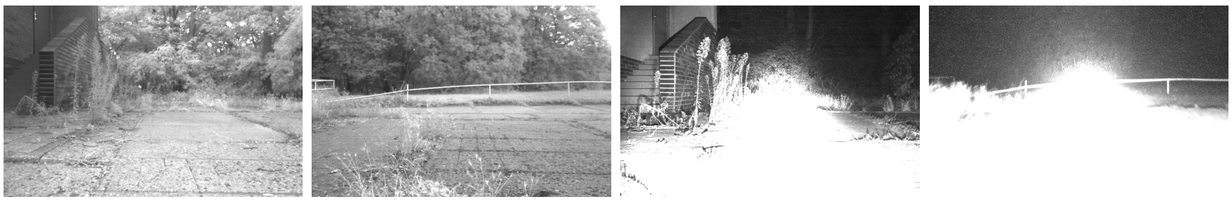

Figure 8: The **'Consolidated' dataset** showing two images from each of the traversals with these during-day conditions: (Day, Without Light), (Night, With Light).

pathway over one month. It follows that there are some adverse conditions across the dataset. There are challenging conditions such as cloudiness, bright sun, light rain, sunset, and night (with LED torch). In addition, the robot following the path turns at some point. Therefore there is an apparent blur present on this position at each traversal. These were included too to demonstrate real-life conditions.

Format and organization There are 32 positions for a traversal and 179 of those traversals. Consequently, 5796 consecutive images were extracted and used for the evaluation. Each of these is a grayscale (monochrome) image with dimensions 768x480px. They were saved to .jpg files and converted into NumPy array type.

Testing robustness As stated above, the 'Consolidated' dataset contains many adverse conditions. The traversals vary in illumination **(during-day changes)**, but also various weather conditions **(weather changes)** and finally may contain different elements depending on the seasonal conditions. **(seasonal changes)**. For this reason, the dataset was used in the experiments since the main objective of this thesis is to increase the robustness across the various environment. At last, the traversal contains positions with brick buildings and a paved yard, which could represent an urban environment. Other positions include trees and other outdoor scenery. With that, we cover a natural environment too.

4.4 Ground truth

The dataset's images were hand-annotated by a human operator. The pixel's absolute horizontal displacement was put in between every image with a base image. A day traversal at the initial position was selected as the base image. The ground truth is therefore always relevant to the first position and traversal. In addition to the displacement, the information, whether it was taken during the day, night, or transition between them. Since there is sometimes a large blur in the images or a significant rotation, the ground truth also contains information stating whether the annotator was not confident in the annotation. We do not use these images for our purpose.

5 Experiments

In this section, we are testing two hypotheses which tell whether the performance of teach-and-repeat navigation can be improved by: (a) selecting features and using feature-matching; (b) using Siamese Neural Network. We get the predicted displacements for each of the methods and compare them with the teach-and-repeat navigation using a feature-matching algorithm (without selecting any features). We compare the prediction errors of proposed and referenced methods, but we also perform statistical tests to confirm the validity of a hypothesis. Ultimately, we discuss the results at the end.

Horizontal displacement error from the ground truth The selected medium to evaluate a performance of a method is an absolute horizontal displacement error. This number tells us the difference between the predicted displacement and the actual displacement when comparing two images. The actual displacement comes from the dataset's ground truth. Admittedly, we ignore the vertical error of the images, but given the teach-and-repeat navigation's properties, it is not invalid. Calculating the displacement error accurately and robustly is a crucial part of the BearNav navigation. Having an accurate and robust error indicates accurate and robust navigation.

Datasets and data pre-processing As it is written in detail in the Dataset section, the 'Consolidated' dataset was used to perform all experiments. The dataset consists of .yaml files from where grayscale images were converted into .jpg files. Further pre-processing was done for the Siamese Neural Network. We used a Keras pre-processing function to pre-process the input to the VGG16 network's needs. The images are converted from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset, without scaling.

5.1 Testing Procedure using Feature Matching

5.1.1 Calculating displacements using feature-matching (reference algorithm)

In the system section, it is described how different heat maps are produced. However, we have to use a feature-matching algorithm along with the produced heat maps. The following text describes the basic feature-matching algorithm, which is used at first as a reference which we will compare our methods. But, at second, we extend this algorithm by the produced heat maps as a new method.
OpenCV package provides feature detectors and descriptors such as SIFT [7], SURF [8], KAZE [10] and AKAZE [11]. We use these to extract the features of an image and use the features to predict a displacement. The process of how we execute the feature-matching can be seen in Algorithm 1.

Selecting features using pre-computed thresholds To prepare a fair comparison, we select only the top 500 features. Yet, there is no implementation of this in OpenCV. Hence, we implement our algorithm. The main obstacle is that the descriptors produce vastly different amounts of features depending on the selected descriptor's thresholds. Where we get 5000 image features for an image taken during a day, we get only 50 image features for the same position of the image but during night time. Naturally, the first thought that comes to mind is setting the descriptor's threshold very low. It follows that even at very adverse and most challenging conditions, it would still predict 500 image features. Yet, it can easily lead up to 100000 image features for the day time image and consequently increasing the computational complexity enormously.
We implement a solution using a binary search to approximate the sub-optimal descriptor's thresholds beforehand. Accordingly, we pre-compute a descriptor's thresholds for each image beforehand to get the top 500 features.

5.1.2 Extended testing procedure with heat maps

The entire testing procedure using feature-matching is described below. (Algorithm 1) Images across the whole dataset are used as the input. Then, features are detected and described for base and target images. The sliding histogram is built and processed. Then, it applies a binary map from the sky detector or a heat map from Selective search to discard some matches (at step 14). This is the only step that differs from the general procedure described in the last section. We put the histogram's peak as the predicted displacement. Finally, we compute displacement errors which tell how far the predictions run away from the ground truth.

Algorithm 1 Feature-matching testing procedure

```
    Input: images[]
    Output: mean, rmse, stdDev - predicted displacements errors
 1: displacements ← []
 2: initialiseDetectors()          ▷ initialize the detector with the baseImage's threshold
 3: baseImage ← images[0]
 4: baseKp, baseDesc ← detectAndCompute(baseImage)
 5:                     ▷ run the detector on baseImage to get keypoints and descriptors
 6: baseKp, baseDesc ← selectBest500Feats(baseKp, baseDesc)
 7: for all targetImage in images do
 8:     initialiseDetectors()      ▷ initialize the detector with the current image's threshold
 9:     targetKp, targetDesc ← detectAndCompute(targetImage)
10:     targetKp, targetDesc ← selectBest500Feats(baseKp, baseDesc)
11:     matches ← match(baseDesc, targetDesc)
12:     hist ← buildHistogram(baseKp, targetKp, matches)
13:     hist ← applyMask(hist)  ▷ OMMITED FOR REFERENCE SOLUTION
14:     histPeak ← getPeak(hist)
15:     displacements.append[histPeak]
16: end for
17: mean ← Mean of displacements[]
18: rmse ← Root-Mean-Square Error of displacements[]
19: stdDev ← Standard Deviation of displacements[]
20: return mean, rmse, stdDev
```

5.2 Testing Procedure for Siamese Neural Network

5.2.1 Predicting displacement using trained SNN

We trained the SNN model for several epochs in order to predict the displacement of two images. For two images, we generate many pairs of image slices. We run the model on each of those pairs to get the similarity scores.

Using similarity score to get displacements Similarity score tells how the slices are likely to be aligned. To predict the actual displacement, we need to process the similarities somehow. We compare a particular target image slice to the center part of the base image. By selecting the center part of the base image, we firstly limited the possible amount of image pairs. But secondly, when the robot moves from a path (it has some displacement), it is more likely having the target image features at the center part of the base image. If we choose a left-most part, we would miss some features encountering with positive displacements. Similarly for the right-most part, we would miss the features when having negative displacements. The displacements for the left-most target slices start with negative

values as they are shifted to the left from the center part (of the base image). Then, the slice's displacement moves to zero (when the slice is aligned with the base slice), and finally, the displacements are positive, stating they are shifted to the right from the base slice.
In Figure 9 there is a plot showing the similarity scores. The final step we need to take is to select a value from the plot as the predicted displacement. We do that by taking the peak of these similarity score curve.

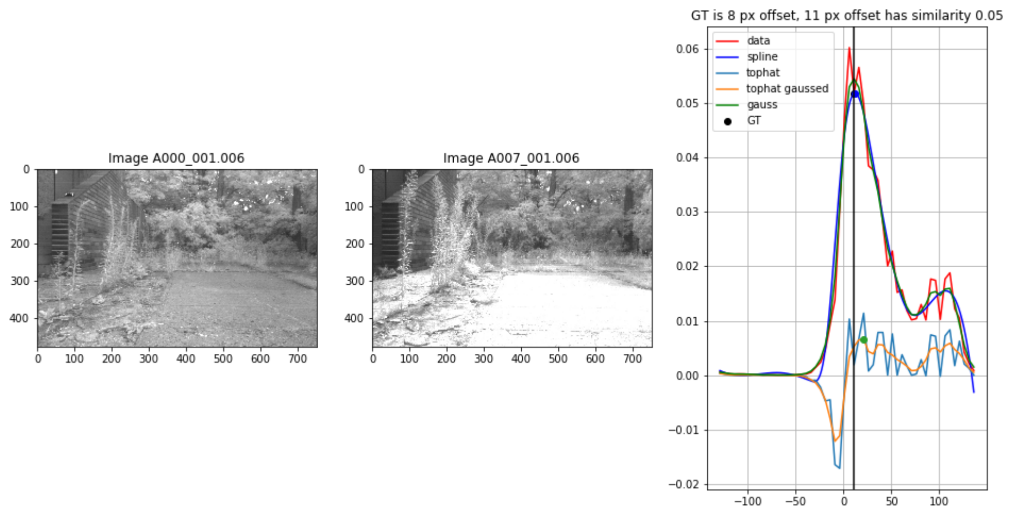

Figure 9: Plot showing the similarities of a certain offset for two images. The spline, Gaussian and Top-Hat transforms are also shown.

Transformation of the similarity score curve Some predictions could be imprecise due to local maxima. To address that, we perform several transformation techniques on the original curve. There is a list of the performed transformations. (You can also see these in Figure 9)

- Spline interpolation
- Gaussian blur - gets rid of sudden changes
- Top-hat transform
- Top-hat transform blured with Gaussian filter

We perform experiments with and without using these transformations to detect if they have the wanted effect.

5.2.2 Siamese Network training procedure

The pseudocode for the procedure can be found below. (Algorithm 2) Predictions of the similarities made by the network are used as the input. Some processing is done, and the output comprises displacement errors which tell us how far they run from the ground truth.

Algorithm 2 Testing procedure for SNN model

```
    Input: sims[], W, H, offsetSize - similarities predicted by the network
    Output: mean, rmse, stdDev - predicted displacements errors
 1: displacements ← []
 2: slidesCount ← $(W - H) \div offsetSize$
 3: displacementBins ← $((W - H)\div) - ([1..slidesCount]) * offsetSize)$   ▷ Calculate the
    possible displacement when getting the center part of the base image
 4: for similarity in sims do
 5:     peak ← displacementBins[argmax(similarity)]   ▷ here we could apply spline,
    gaussian blur, or Top-hat transformation
 6:     displacements.append[peak]
 7: end for
 8: mean ← Mean of displacements[]
 9: rmse ← Root-Mean-Square Error of displacements[]
10: stdDev ← Standard Deviation of displacements[]
11: return mean, rmse, stdDev
```

5.3 Results

Finally, we look on the quantitative results of the experiments. In the tables below, you can find the computed errors for day, transition and night traversals.

5.3.1 Results for heat maps

There are no improvement of the error when selecting features using eithr selective search or the sky detector.

	w/ Selective Search maps				w/o maps (REFERENCE)			
	Day	Transition	Night	All	Day	Transition	Night	All
N	2999	246	2203	5448	2999	246	2203	5448
Mean	28	80	164	85	23	47	110	59
RMSE	74	137	230	159	66	92	199	137
Std dev	69	110	161	134	61	78	166	124

	w/ sky detector maps				**w/o maps (REFERENCE)**			
	Day	Transition	Night	All	Day	Transition	Night	All
N	2999	246	2203	5448	2999	246	2203	5448
Mean	32	82	170	95	23	47	110	59
RMSE	79	125	245	160	66	92	199	137
Std dev	79	110	165	143	61	78	166	124

The approach of generating heat maps to select more robust features did not bring any significant results. Both the adapted selective search and sky detector did not improve the mean or RMSE (Root-Mean-Square Error)for any conditions. The standard deviation was also not improved. SURF with other detectors are probably resilient to the features we discarded by applying the heat maps. To conclude, the first hypothesis was not approved, and the performance did not get significantly better by selecting features.

5.3.2 Results for Siamese Neural Network

	SNN based				**REFERENCE**			
	Day	Transition	Night	All	Day	Transition	Night	All
N	2999	246	2203	5448	2999	246	2203	5448
Mean	54	49	106	75	23	47	110	59
RMSE	80	64	192	136	66	92	199	137
Std dev	**58**	**42**	**159**	**113**	61	78	166	124

The SNN model's Mean error is significantly worse at day traversals. On the other hand, it has a slightly smaller mean and RMSE error at night traversals. Moreover, we see a lower standard deviation across all traversals.

5.4 Statistical tests

So far, we compared different kinds of errors to show if the proposed methods improved the performance or not. To add more credibility to the experiments, we perform statistical tests. We will use paired test to compare two methods. Typically, we would use a statistical t-test for that purpose. There is an assumption of Normal distribution of the data. Firstly, we test if our data are, indeed, normally distributed.

Normal distribution assumption To determine whether our data are modeled for normal distribution, we calculate skewness and kurtosis. The skewness measures the asymmetry of the probability distribution of a random variable about its mean. In other words, the skewness describes the amount and direction of the horizontal symmetry diversion.

The skewness can have positive or negative values. Generally, if the absolute value of skewness is more than 1, the data distribution is highly skewed (and therefore no normally distributed). If the skewness is close to zero (its absolute value is less than 0.5), the distribution is approximately symmetric. Kurtosis describes the height and sharpness of the central peak of the data relative to the peak of a standard bell curve.

$$\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^3}{(\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2)^{\frac{3}{2}}}$$

$$\text{Kurtosis} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^4}{(\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2)^2} - 3$$

Here, $\bar{x}$ is the sample mean. The calculated values can be seen in the tables below.

	Siamese Neural Network data				**SURF (REFERENCE) data**			
	Day	Transition	Night	All	Day	Transition	Night	All
N	2999	246	2203	5448	2999	246	2203	5448
Skew	4.44	1.61	3.45	4.80	6.23	2.76	2.62	3.81
Kurtosis	41.17	2.87	11.84	26.65	51.43	8.41	7.02	16.67

Wilcoxon test If the data are not normally distributed, we have to use an alternative test. As our paired data samples are not independent, the Mann-Whitney U test cannot be used. Given the above, we use the Wilcoxon signed-rank test. This test produces a p-value which enables us to either reject the null hypothesis in favor of the alternative hypothesis. If the p-value is less or equal to 0.05, we will reject the null hypothesis at a confidence level of 95%.

One-sided test to compare two methods The null hypothesis for the one-sided test says that the first method (reference) is significantly better than the other one. In other words, that the median of differences between the samples is negative. The alternative hypothesis tells the opposite.

h_0: The median of differences is positive, stating the (second) method does not significantly improve the performance over the reference (first) method.
h_1: The median of differences is negative, stating the (second) method significantly improves the performance over the reference (first) method.

We tested the reference feature-matching algorithm using the SURF detector with the Siamese Neural Network-based image alignment method. As shown below, the resulting p-values for the test are greater than 0.05 for each of the dataset conditions. Hence, the null hypothesis that the median is positive cannot be rejected at a confidence level of 95%.

Accordingly, we can state that the SNN-based method does not significantly improve the performance over the SURF-based one.

	Wilcoxon test SURF w/ SNN			
	Day	Transition	Night	All
N	2999	246	2203	5448
Wilcoxon, W-val	1507966	10662	820667	5081326
Wilcoxon, p-val	1.00	0.99	1.00	1.00

5.5 Main Findings

Ultimately, the goal was to predict displacements in between two images and thus align them to correct the wheeling motion of a robot. The robot can get lost when it records large errors in a few consecutive images. High accuracy across the dataset, or specificly its mean, does not guarantee large errors won't occur. The standard deviation for the Siamese Neural Network is lower under every traversal's condition. It follows that with the standard deviation, the Siamese network actually increases the overall stability of the navigation.

6 Conclusion

We addressed the problem of bearing-only navigation systems for mobile robots. These systems enable the robot to follow a path over a long period where seasonal and illumination changes occur. We primarily focused on improving the robustness of such systems. This system is based on estimating the correct alignment of two consecutive images which the robot captures.

Based on the BearNav [5] that copes well with the indoor task, we tried to adjust the system to be more robust. (see Section 3 for details) The robustness and stability of long-term navigations depend extensively on the dataset and conditions given. We used the 'Consolidated' dataset for testing purposes since it contains adverse conditions. It represents both the urban environment and nature. It has some imperfect images to, once again, simulate the real-life conditions.

Firstly, we follow a feature-matching localization algorithm used in BearNav and extend it by selecting features used for the matching. A state-of-the-art sky detector was used here to withdraw sky regions. The idea is that these areas do not contribute any informative value. Thus, we could discard the features located in these regions. It can be concluded that it had no positive impact on the performance of SURF [8] or other commonly used detectors. The stability did not improve using the sky detector as well.

Secondly, we implemented another detector to identify significant features. It is based on the Selective Search [13] algorithm, where it takes the bounding boxes to compute a per-pixel density heat map. Again, we apply this heat map to select features for the feature-matching algorithm used in BearNav. Given the above, the performance was not increased by the produced heat maps. We did not encounter any improvements made on the stability too.

Thirdly, we proposed an image alignment based on the research made in Siamese Neural Networks (SNNs) [34]. Neural Networks are not based on per-pixel operations, and they include two Convolutional Neural Networks with shared parameters. Further, we use the Convolutional Neural Network's abstract semantic information. Siamese Neural Network is commonly used to detect similarities or changes in a pair of images. We use it to predict a displacement of two consecutive images and ultimately align the pair for navigation needs. The alignment is performed via a sliding window approach, where a center-cropped patch is slid across a reference image. The highest peak of the computed similarity scores is selected as the correct match. SNN performed with similar accuracy for the night and transition traversals as the reference algorithm. Nevertheless, the standard deviation was significantly lowered across all traversals. In other words, it did not get significantly worse predictions during challenging adverse conditions, while the stability of the localization was increased. As stated once, the robustness and stability of the localization are crucial for mobile robots operating long-term. Single significant errors might permanently get another robot lost. Given the above, the SNN fulfills our efforts as the algorithm is more stable and more accurate for adverse conditions. It is worth mentioning that the system's overall accuracy did not outperform the state-of-the-art systems.

As the results from the SNN model are promising, we can conclude that the idea of using Neural Networks is valid, and further investigations and experiments could be done.

References

[1] Nick Hawes, Christopher Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrova, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Kortner, Rares Ambrus, Nils Bore, John Folkesson, Patric Jensfelt, Lucas Beyer, Alexander Hermans, Bastian Leibe, Aitor Aldoma, Thomas Faulhammer, Michael Zillich, Markus Vincze, Eris Chinellato, Muhannad Al-Omari, Paul Duckworth, Yiannis Gatsoulis, David C. Hogg, Anthony G. Cohn, Christian Dondrup, Jaime Pulido Fentanes, Tomas Krajnik, Joao M. Santos, Tom Duckett, and Marc Hanheide. The strands project: Long-term autonomy in everyday environments. *IEEE Robotics Automation Magazine*, 24(3):146–156, 2017.

[2] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[3] Z. Chen and Stan Birchfield. Qualitative vision-based mobile robot navigation. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2686–2692, 2006.

[4] Paul Furgale and Timothy Barfoot. Visual teach and repeat for long-range rover autonomy. *J. Field Robotics*, 27:534–560, 09 2010.

[5] Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přeučil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, 2010.

[6] Tomas Krajnik, Filip Majer, Lucie Halodová, Jan Bayer, Tomas Vintr, and Jan Faigl. Navigation without localisation: reliable teach and repeat based on the convergence theorem. *arXiv preprint arXiv:1711.05348*, 2017.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[9] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011.

[10] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew Davison. Kaze features. 10 2012.

[11] Pablo Fernández Alcantarilla. Fast explicit diffusion for accelerated features in non-linear scale spaces. 09 2013.

[12] D. Austin, L. Fletcher, and A. Zelinsky. Mobile robotics in the long term-exploring the fourth dimension. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 2, pages 613–618 vol.2, 2001.

[13] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[14] Yehu Shen and Qicong Wang. Sky region detection in a single image for autonomous ground robot navigation. *International Journal of Advanced Robotic Systems*, 10:1, 10 2013.

[15] Randall Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5, 02 1987.

[16] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 1442–1447 vol.3, 1991.

[17] G. Grisettiyz, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2432–2437, 2005.

[18] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M.E. Munich. The vslam algorithm for robust localization and mapping. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 24–29, 2005.

[19] Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, 2003.

[20] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[21] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[22] David Prokhorov, Dmitry Zhukov, Olga Barinova, Anna Vorontsova, and Anton Konushin. Measuring robustness of visual slam, 2019.

[23] Stephanie Lowry. Similarity criteria: evaluating perceptual change for visual localization. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2019.

[24] Keisuke Yoneda, Naoki Suganuma, Ryo Yanase, and Mohammad Aldibaja. Automated driving recognition technologies for adverse weather conditions. *IATSS Research*, 43(4):253–262, 2019.

[25] 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.

[26] Feras Dayoub and Tom Duckett. An adaptive appearance-based map for long-term topological localization of mobile robots. 09 2008.

[27] Mathias Bürki, Marcin Dymczyk, Igor Gilitschenski, Cesar Cadena, Roland Siegwart, and Juan Nieto. Map management for efficient long-term visual localization in outdoor environments. pages 682–688, 06 2018.

[28] Peter Mühlfellner, Mathias Bürki, Michael Bosse, Wojciech Derendarz, Roland Philippsen, and Paul Furgale. Summary maps for lifelong visual localization. *Journal of Field Robotics*, 33, 06 2015.

[29] Lucie Halodová, Eliška Dvořáková, Filip Majer, Jiří Ulrich, Tomas Vintr, Keerthy Kusumam, and Tomáš Krajník. *Adaptive Image Processing Methods for Outdoor Autonomous Vehicles*, pages 456–476. 01 2019.

[30] Stephanie M. Lowry, Michael J. Milford, and Gordon F. Wyeth. Transforming morning to afternoon using linear regression techniques. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3950–3955, 2014.

[31] H. Porav, W. Maddern, and P. Newman. Adversarial training for adverse conditions: Robust metric localisation using appearance transfer. pages 1011–1018, 2018.

[32] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. 12 2017.

[33] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale, 2019.

[34] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. pages 378–383, 12 2016.

[35] Tomáš Krajnık. *Large-scale mobile robot navigation and map building.* PhD thesis, Ph. D. thesis, Czech Technical University in Prague, 1999, Draft, 2011.

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[37] Niko Sünderhauf, Peer Neubert, and Peter Protzel. Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, page 2013, 01 2013.

[38] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[39] Wei Zhang and J. Kosecka. Localization based on building recognition. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pages 21–21, 2005.

Appendix

CD Content

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

Directory name	**Description**
text	bachelor thesis in pdf format
sources	source codes

Table 1: Content of the attachment