Inaugural-Dissertation
zur Erlangung des Grades
Doktorin der Ingenieurwissenschaften (Dr.-Ing.)
der Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

Visual Place Recognition in Changing Environments

von

Olga Vysotska

aus

Kyjiw, Ukraine

Bonn 2019

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenenen Hilfsmittel angefertigt habe; die aus fremdem Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum	(Unterschrift)

Zusammenfassung

Mobile Roboter, selbstfahrende Autos und andere autonome mobile Systeme müssen wissen wo sie sich in der Umgebung befinden, um effizient navigieren zu können. Diese Fähigkeit bezeichnet man als Lokalisierung. Man unterscheidet dabei zwischen Systemen, die Ihre Position ohne Vorwissen bestimmen können und als globale Lokalisierungsmethoden bezeichnet werden, und Systemen, die von einer gegebenen Startposition aus ein sogenanntes Verfolgen der Position (Tracking) durchführen.

Diese Arbeit beschäftigt sich mit einer speziellen Form der globalen visuellen Lokalisierung, die oft als Ortserkennung bezeichnet wird. Dabei geht es in erster Linie nicht um die Bestimmung der Position des Roboters in Form einer Koordinate in einem festen Koordinatensystem. Der Roboter soll stattdessen basierend auf Kamerabildern einen Ort wiedererkennen, den er bereits in der Vergangenheit besucht und wahrgenommen hat. Auch wenn sich dies für uns Menschen nach einem einfachen Problem anhört, stellt diese Aufgabe für technische Systeme eine große Herausforderung dar. Einer der Gründe liegt dabei in der kontinuierlichen Veränderung des Erscheinungsbilds eines Ortes, bedingt durch Beleuchtung, Wetterbedingungen, Jahreszeiten oder andere auf Menschenhand zurückführende Veränderungen. Auch haben starke Veränderungen der Aufnahmeposition einen signifikanten Einfluss auf die aufgenommenen Bilder und daher die Erkennung eines Ortes. All diese Effekte verändern die Art und Weise, wie der gleiche Ort im Bild dargestellt wird und dies kann zu Situationen führen, in denen es selbst für Menschen schwierig ist Orte wiederzuerkennen. Das zuverlässige Wiedererkennen beliebiger Orte unter solchen Veränderungen ist daher ein komplexes Problem.

In der Robotik spielen visuelle Sensoren oft eine zentrale Rolle, wenn es um die Wahrnehmung der Umgebung geht. Sie sind neben Lasersensoren die vermutlich am häufigsten genutzten Sensoren, was auf Preis, Größe und Gewicht zurückzuführen ist. Daher findet man auch eine Vielfalt an innovativen Anwendungen aus dem Bereich der virtuellen und erweiterten Realität (Virtual and Augmented Reality), des autonomen Fahrens oder der Logistik, die auf Kameras basieren. Fast alle diese Anwendungen benötigen eine Lokalisierung, d.h. sie müssen wissen, wo sich das System aktuell befindet. Daher ist diese Arbeit für autonome

Systeme relevant, die sich auch unter kontinuierlich ändernden Umweltbedingungen zurechtfinden müssen.

Ein zentraler Beitrag dieser Arbeit ist der Ansatz sich von der Erkennung aus Einzelbildern zu lösen und Bildsequenzen zu nutzen. Dies ist möglich, da autonome Systeme ihre Sensordaten sequentiell erfassen und nicht in zufälliger Reihenfolge. Daher formulieren wir in dieser Arbeit das Problem der visuellen Ortserkennung unter starken optischen Veränderungen als das Problem Bildsequenzen, die zu verschiedenen Zeitpunkten aufgenommen wurde, zu registrieren. Eine wichtige Erkenntnis ist, dass dadurch zeitweise Mehrdeutigkeiten in den Datenzuordnungen reduziert oder sogar vollständig aufgelöst werden können. Um die Suche nach Bildsequenzkorrespondenzen zu formulieren, nutzen wir gerichtete azyklische Datenassoziationsgraphen. Die Knoten in einem solchen Graphen modellieren potentielle Übereinstimmungen zwischen Bildern, während die Kanten gleichzeitig die Ordnung der Aufnahmesequenz bewahren. Der kürzeste Weg durch einen solchen Assoziationsgraphen liefert dann die besten Einzelbildkorrespondenzen gegeben die Sequenzinformation.

In dieser Arbeit betrachten wir verschiedene Varianten dieses Problems. Dies beinhaltet Online-Verfahren, um zu jedem Zeitpunkt der Navigation die beste Lokalisierungsschätzung zu berechnen, ohne alle Bilder erneut betrachten zu müssen. Des Weiteren untersuchen wir in wie weit andere Lokalisierungsquellen wie beispielsweise GNSS Informationen mit unserem Verfahren verknüpft werden können. Eine Herausforderung bei der Lokalisierung mittels Datensequenzen ist es, Orte online wiederzuerkennen, auch wenn der Roboter den bekannten Umgebungsbereich verlassen hat, d.h. eine gewisse Zeitlang nicht lokalisiert werden konnte.

Fast alle visuellen oder auf Lasersensoren basierenden Lokalisierungssysteme und so auch die von uns vorgeschlagenen Verfahren, haben den Nachteil, dass Orte zwar *wieder*erkannt werden können, aber die Positionsbestimmung in einer zuvor nicht befahrenen Umgebung nicht gut möglich ist. Um einen Roboter in die Lage zu versetzen sich auch in großflächigen Umgebungen zu lokalizieren, ohne das explizite Sammeln eines Referenzdatensatzes zu verlangen, schlagen wir des Weiteren Verfahren vor, die auf öffentlich zugänglichem Bild- und Kartenmaterial wie Google Street View oder OpenStreetMap Daten aufbauen. Dadurch kann eine explizite Kartensammelphase durch den eigenen Roboter vermieden werden. Unser Verfahren ermöglicht es, automatisch eine Ortserkennung auf dieser Art von Daten durchzuführen.

Alle in dieser Arbeit beschriebenen Ansätze wurden in Form von peer-reviewed Konferenzbeiträgen und Zeitschriftenartikeln veröffentlicht. Darüber hinaus wurden die meisten der präsentierten Beiträge als quelloffene Software veröffentlicht.

Abstract

Localization is an essential capability of mobile robots and place recognition is an important component of localization. Only having precise localization, robots can reliably plan, navigate and understand the environment around them. The main task of visual place recognition algorithms is to recognize based on the visual input if the robot has seen previously a given place in the environment. Cameras are one of the popular sensors robots get information from. They are lightweight, affordable, and provide detailed descriptions of the environment in the form of images. Cameras are shown to be useful for the vast variety of emerging applications, from virtual and augmented reality applications to autonomous cars or even fleets of autonomous cars. All these applications need precise localization. Nowadays, the state-of-the-art methods are able to reliably estimate the position of the robots using image streams. One of the big challenges still is the ability to localize a camera given an image stream in the presence of drastic visual appearance changes in the environment. Visual appearance changes may be caused by a variety of different reasons, starting from camera-related factors, such as changes in exposure time, camera position-related factors, e.g. the scene is observed from a different position or viewing angle, occlusions, as well as factors that stem from natural sources, for example seasonal changes, different weather conditions, illumination changes, etc. These effects change the way the same place in the environments appears in the image and can lead to situations where it becomes hard even for humans to recognize the places. Also, the performance of the traditional visual localization approaches, such as FABMAP or DBow, decreases dramatically in the presence of strong visual appearance changes.

The techniques presented in this thesis aim at improving visual place recognition capabilities for robotic systems in the presence of dramatic visual appearance changes. To reduce the effect of visual changes on image matching performance, we exploit sequences of images rather than individual images. This becomes possible as robotic systems collect data sequentially and not in random order. We formulate the visual place recognition problem under strong appearance changes as a problem of matching image sequences collected by a robotic system at different points in time. A key insight here is the fact that matching sequences reduces the ambiguities in the data associations. This allows us to establish im-

age correspondences between different sequences and thus recognize if two images represent the same place in the environment. To perform a search for image correspondences, we construct a graph that encodes the potential matches between the sequences and at the same time preserves the sequentiality of the data. The shortest path through such a data association graph provides the valid image correspondences between the sequences.

Robots operating reliably in an environment should be able to recognize a place in an online manner and not after having recorded all data beforehand. As opposed to collecting image sequences and then determining the associations between the sequences offline, a real-world system should be able to make a decision for every incoming image. In this thesis, we therefore propose an algorithm that is able to perform visual place recognition in changing environments in an online fashion between the query and the previously recorded reference sequences. Then, for every incoming query image, our algorithm checks if the robot is in the previously seen environment, i.e. there exists a matching image in the reference sequence, as well as if the current measurement is consistent with previously obtained query images.

Additionally, to be able to recognize places in an online manner, a robot needs to recognize the fact that it has left the previously mapped area as well as relocalize when it re-enters environment covered by the reference sequence. Thus, we relax the assumption that the robot should always travel within the previously mapped area and propose an improved graph-based matching procedure that allows for visual place recognition in case of partially overlapping image sequences.

To achieve a long-term autonomy, we further increase the robustness of our place recognition algorithm by incorporating information from multiple image sequences, collected along different overlapping and non-overlapping routes. This allows us to grow the coverage of the environment in terms of area as well as various scene appearances. The reference dataset then contains more images to match against and this increases the probability of finding a matching image, which can lead to improved localization. To be able to deploy a robot that performs localization in large scaled environments over extended periods of time, however, collecting a reference dataset may be a tedious, resource consuming and in some cases intractable task. Avoiding an explicit map collection stage fosters faster deployment of robotic systems in the real world since no map has to be collected beforehand. By using our visual place recognition approach the map collection stage can be skipped, as we are able to incorporate the information from a publicly available source, e.g., from Google Street View, into our framework due to its general formulation. This automatically enables us to perform place recognition on already existing publicly available data and thus avoid costly mapping phase. In this thesis, we additionally show how to organize the images from

the publicly available source into the sequences to perform out-of-the-box visual place recognition without previously collecting the otherwise required reference image sequences at city scale.

All approaches described in this thesis have been published in peer-reviewed conference papers and journal articles. In addition to that, most of the presented contributions have been released publicly as open source software.

Acknowledgements

I would like to take time and explicitly thank people who made this exciting PhD journey possible.

Firstly, I would like to thank my brilliant supervisor Cyrill Stachniss for his guidance, patience, and support throughout these years. Without his care, this amazing scientific journey would not be possible for me. He spotted me in times no one else knew me, took me under his wing, educated me, and gave me the opportunity to evolve as a researcher, a teacher, and a person by sharing his priceless knowledge and experience. I thank him for the unconditional support and kind words when the times got rough and providing an environment where I could feel welcome, appreciated, and valued. I am extremely thankful for sharing the funs and rushes of late night deadlines, conference trips, and scientific discussions. Thank you, Cyrill, for believing in me and giving me a chance.

Secondly, I want to thank my dear friends that shared this experience with me and were by my side in times of happiness and sorrows. I would like to thank Nived Chebrolu for the infinite pleasant talks filled with science and flavored with philosophies. Thank you for teaching me to cherish every moment in life, every creature, and for making me a more optimistic and positive person. I am also grateful to Lorenzo Nardi for being a friend every one wants to have, a friend I could rely on in any situation. Furthermore, I want to thank my American friend and supervisor David Rosen for showing me that math can be easy, answering all my challenging questions and patience to explain things for me over and over again in simpler and simpler English. I also would like to thank Raul Mur-Artal for explaining me tips and tricks about the visual odometry. I am grateful also to Maxim Tatarchenko for scientific and life encouragements and mental support during my PhD. Moreover, I would like to thank my friends and lab mates Andres Milioto, Emanuele Palazzolo, Philipp Lottes, Jens Behley, Thomas Läbe, Ignacio Vizzo, Maren Bennewitz, and Ribana Roscher for making my time in the office a pleasant one and for sharing the joys and funs at the various conferences. I would also like to thank Susanne Wenzel for sharing the office with me and introducing me to the world of photogrammetry. I would also like to express my sincere gratitude to Birgit Klein, who supported me throughout this time in a battle against administrative hassle and gave me the invaluable lessons on the German language no traditional courses could teach.

There are many more people with whom I collaborated and interacted throughout these years and I would like to thank them for shaping my world in their own unique ways: Giorgio Grisetti, Michael Ruhnke, Wolfram Burgard, Tim Caselitz, Noha Radwan, Pratik Agarwal, Nichola Abdo, and Christoph Sprunk.

My family has a special place in my life and I would like to thank my parents for their care and support since I can remember myself and a little bit before that. Thank you for always have confidence in me, guiding me, and pushing me towards new horizons and beyond.

Last but not least, I would like to thank Igor Bogoslavskyi for the years of continuous, strong, comprehensive support, pure faith in me and my abilities, kind and strong words of encouragement and a sane dose of criticism. Thank you for always... always being there for me.

Contents

Chapter 1

Introduction

Autonomy is one of the crucial prerequisites for the robots to become a part of our daily lives. For a system to be truly autonomous it should fulfill the range of the tasks: understanding where it is, reasoning about the environment, being able to perform tasks and all of these without human intervention. The first step for the mobile systems to achieve full autonomy is to be able to estimate its location in the environment and understand how the environment looks like: is it dynamic or static, are there buildings or people nearby, where the robot can go, and where the environment is unfavorable for the robot to be in. The process of estimating the robot's pose or *localization* is only possible if there is a representation of the environment, called a *map*. Therefore, before the robot can start localization it needs the map, or to build a map to localize within. Typical robot operations start with building a map of the environment and the localization stage comes afterward. Another approach is to continuously perform localization simultaneously to mapping and is known as the Simultaneous Localization and Mapping (SLAM) problem. In both cases, localization is one of the main parts. For the robots to reason if they are in location, where they have been previously, they need to have an ability to recognize the place given the sensor inputs.

Nowadays, one of the popular sensors are the cameras due to the range of attractive properties. The cameras have typically small size so that it can be carried by a small drone, are easier to deploy in comparison to LiDAR, energy efficient, and costs less. Another property of the camera sensor mounted on a robotic system is the fact that the images come in sequential order in the form of image streams. In particular, the approaches that perform visual localization rely on the stream of images to estimate the robot's pose based on visual input.

To be able to localize a robot within a given environment, the system should be able to recognize a similar place given the sensor measurements. Visual place recognition solves the problem of recognizing similar places based on visual data. There exist a variety of approaches that perform visual place recognition from

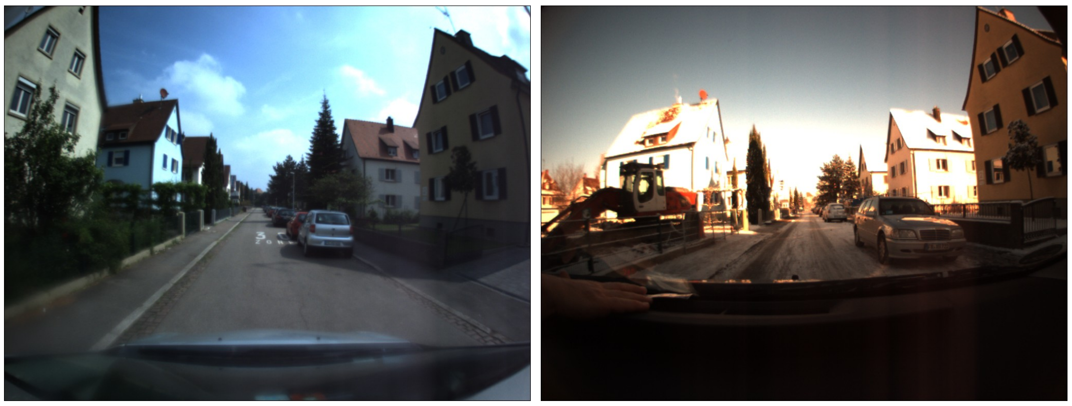


Figure 1.1: The same place in the environment can undergo dramatic visual appearance changes induced by seasons, time of day, and dynamic objects change. Left: Image taken in summer afternoon. Right: Image taken during winter morning.

the visual input among others are ORB-SLAM by Mur-Artal *et al.* [95] and LSD-SLAM by Engel *et al.* [40]. Both systems use local image feature descriptors to match places and achieve persistent localization results. However, the problem is far from being solved for robots to be deployed in outdoor environments for extended periods of time. Nowadays, the main challenge of recognizing if two images represent the same place comes from the fact that the visual appearance of the places may change dramatically between the points in time when the environment was mapped and the times the robot revisits the same place again.

Typical changes that make a place look differently are viewpoint changes, due to rotated and shifted camera, illumination changes that come from the natural, as well as artificial sources, seasonal changes or changes, caused by dynamic objects. The visual appearance changes in outdoor environments have a bigger effect on recognition because they tend to severely change the appearance of the place. Figure 1.1 depicts an example of how seasonal changes as well as the time of day changes influence the visual appearance of the same place. Another particular challenge in this setup is the absence of a building in the winter image. In this thesis, we investigate the problem of visual place recognition in changing environments that experience dramatic visual appearance changes.

The task of recognizing if two images show the same place can be challenging not just for the computer systems but also for humans, especially if the appearance was influenced by substantial changes. For humans, the task of associating images becomes simpler whenever the images come in the sequential order. In that way, people first establish a data association between a pair of images and later make a decision if the consecutive pairs also match.

In typical robotics applications, the information from the sensor also comes in a sequential manner rather than in random order. We exploit this property and approach the problem of visual place recognition as a problem of matching image sequences. Ordering image into sequences diverges us from traditional image retrieval approaches to visual place recognition but allows to robustly deal with dramatic appearance changes.

1.1 Main contributions

Representing a place recognition as a sequence matching problem comes naturally when considering a typical robot operation. First, the robot navigates through the environment, collecting a sequence of images, which is typical serves as *reference* or *database* sequence and later tries to match the current observations typically referred to as *query* sequence.

To approach the problem of matching image sequence, we draw inspiration from the work of Naseer *et al.* [99]. In their work, the authors propose to use a graph structure that binds potential image correspondences between the sequences while preserving the sequential information within individual sequences, i.e. reference and query. The estimated network flow through such a graph results in potential image associations between the sequences and provide the results for the place recognition system. In Chapter 3, we provide a detailed explanation about the data associations graph structures. One of the weak sides of the proposed graph search algorithm though is the necessity to instantiate a cost matrix between all pairs of images between two sequences. One of the contributions of this thesis is the incorporation of pose priors into the graph search procedure that allows us to dramatically reduce the number of image-to-image comparisons. Moreover, by adding information from such priors we eliminate the need to formulate a network flow problem and solve the data association task using the more efficient topological sorting method.

In Chapter 4, we bring our place recognition approach closer to the real world application by turning the offline matching procedure from Chapter 3 into an online approach for matching image sequences. In this way, we are able to make the image associations decisions on the fly, i.e., for every incoming query image, as opposed to the previous approach where the search can only be performed whenever the complete sequences are collected.

The graph formulation from Chapter 3 and Chapter 4 imposes a constraint for the image sequences. It assumes that the robot drives roughly similar trajectories in the environment. This constraint, though, is easily violated in the real world due to the variety of reasons. For example, if both sequences temporally visit different places in the environment, the query sequences visits the place not mapped with reference sequence, etc. Chapter 5 proposes a novel hashing based technique that allows to robustly deal with different trajectories.

The techniques for visual place recognition described before taking into account only two image sequences. In Chapter 6, we extend the graph search idea to account for multiple reference sequences that form a map of image sequences.

As mentioned before, the localization stage typically requires a map to localize within. Collecting a map can be a resource consuming operation. As a further contribution of this thesis, we show how to use publicly available information such as Google Street View to avoid an expensive mapping phase and additionally be able to localize within the global reference frame, see in Chapter 7.

Publicly available information can also enhance the SLAM systems, so that in Chapter 8, we propose a novel approach to use the building information from OpenStreetMap to maintain the consistency of the robot built maps especially in the case of 2D laser range finder that is prone to accumulate a drift. Moreover, we present an approach that facilitates better planning by estimating the localizability information on the map. This allows for planning better routes so that the localization system is less likely to fail. This approach takes into account the particular properties of the sensor. Thus, this thesis tackles several challenging problems in the context of robot place recognition and localization.

1.2 Publications

Parts of this thesis have been published in the following peer-reviewed papers:

- O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Efficient and Effective Matching of Image Sequences Under Substantial Appearance Changes Exploiting GPS Prior. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015

- O. Vysotska and C. Stachniss. Lazy Data Association for Image Sequences Matching under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 2016

- O. Vysotska and C. Stachniss. Exploiting building information from publicly available maps in graph-based slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016

- O. Vysotska and C. Stachniss. Relocalization under substantial appearance changes using hashing. In *Proc. of the IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2017

- O. Vysotska and C. Stachniss. Improving slam by exploiting building information from publicly available maps and localization priors. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, 85(1):53–65, 2017

- O. Vysotska and C. Stachniss. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, 2019

To facilitate further the research in place recognition, we have open-sourced our code for the community:

- `https://github.com/PRBonn/online_place_recognition`
- `https://github.com/PRBonn/vpr_relocalization`
- `https://github.com/ovysotska/image_sequence_matcher`

We further published a dataset on visual place recognition in changing environments at:
`http://www.ipb.uni-bonn.de/data/visual-place-recognition-datasets/`

Chapter 2

Basic techniques

2.1 Matching a pair of images

In this section, we will give a brief overview of how to compare two images. We furthermore explain how we use to describe images throughout this thesis to be able to perform robust visual place recognition in changing environments.

To reason if two images show the same place in the environment, an autonomous system must be able to compare both images. One of the most intuitive ways to do that is to compare the images on a pixel per pixel basis, i.e., computing the difference between corresponding pixels and summing up these differences over all pixels. The drawback of this method, though, lies in the fact that a slight difference in the pixel intensity values caused by illumination changes or by a slight shift in pixels results in large total differences, even though the images exhibit the same place in the environment and may even appear visually highly similar to humans.

In the real world, however, most of the times the images taken from the same place are effected by view-point changes with varying rotation, translation or tilt. Thus, a more robust way to compare images under such conditions is to extract local feature descriptors such as SIFT [82], SURF [20], or ORB [110]. Then, two images can be compared by considering the corresponding local features between the images and checking how many correspondences agree with each other geometrically, for example, using RANSAC [42]. See Figure 2.1 for an illustration.

A computationally more efficient way to compute a matching cost, i.e., a similarity between the images, for a pair of images using local features is to quantize an image into a histogram of so-called visual words, also known as bag-of-words [122] and to compare two histograms using Euclidean or cosine distance.

As shown by Valgren *et al.* [136], local features tend to lose their descriptive performance in the presence of dramatic visual appearance changes, like seasonal

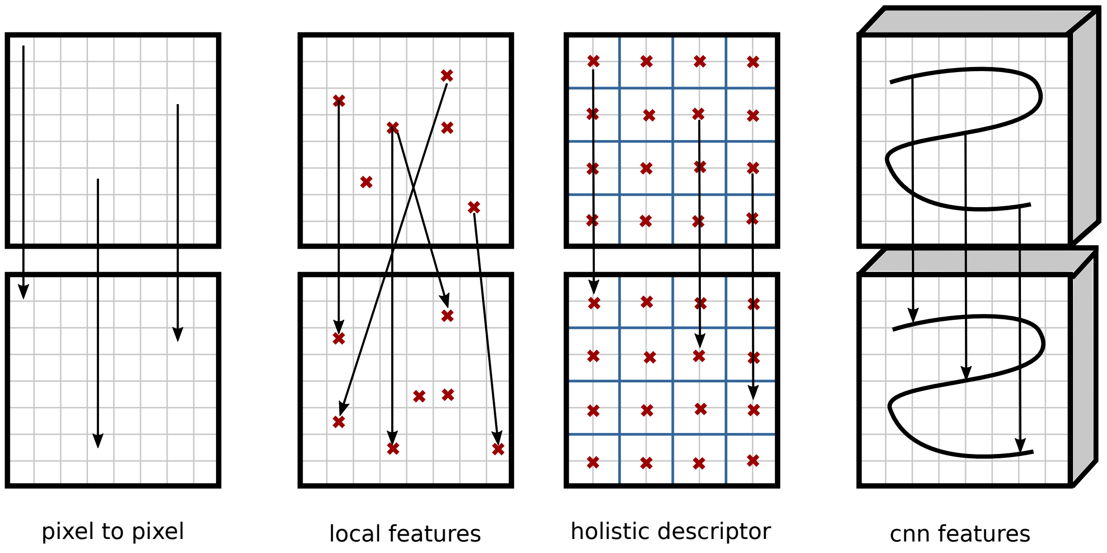


Figure 2.1: Approaches to compare a pair of images. From left to right: On per pixel bases; describing image with local features and compare them; describing images using grid of local descriptors and extracting feature vectors from the convolutional neural network. Red crosses correspond to the extract local feature descriptors like SIFT, SUFT, etc.

changes, weather condition changes, illumination changes, etc. Holistic, or whole image descriptors, tend to have a better performance under drastic photometric changes, due to their better ability to disambiguate similar local feature descriptors. A holistic descriptor defines a fixed grid over an image and extracts in every cell a local descriptor, see Figure 2.1 3$^{\mathrm{d}}$ column. In this way, we constraint the area in which the correspondent local feature descriptors should be located in the other image and thus successfully find feature matches which would be considered as weak and discarded as outliers otherwise. Due to the superior performance of holistic descriptors in visually challenging situations, we use tessellated HOG descriptor [35] as an image description method in Chapter 3 of this thesis.

In recent years, the boost in the neural network developments for object detections gave rise for improving image feature description. The features extracted from the certain layers of convolutional neural networks have been shown to perform better for the place recognition tasks [130] than traditional local features descriptors as well as their holistic grid counterparts. In Chapter 4, we show that features from the Overfeat [119] convolutional network produce matching costs that are more distinct than the costs obtained form tessellated HOG descriptors. The matching costs are more distinct if they result in a broader range of values between the perfect match (identical images) and images that exhibit different places. We obtain a holistic descriptor by concatenating vectors from the feature volume out of 10^{th} layer of the Overfeat CNN, as sketched in Figure 2.1, 4$^{\mathrm{th}}$ column. Afterward, we compare the images by computing a cosine distance between the concatenated vectors. Concatenating the CNN feature vectors in a certain order corresponds to fixing the position of that feature volume in a particular part of the image, which may be seen as a form of a holistic descriptor.

Even though both holistic descriptors, i.e., those coming from a CNN or not, tend to exhibit a better performance than traditional local feature descriptors in presence of drastic visual appearance changes, their robustness is limited in cases of changed viewpoint or large translations of the camera in between the images. To perform visual place recognition under the drastic visual appearance changes induced by environmental changes as well as provoked with camera motion, we use in Chapter 6 features from the recently proposed NetVLAD [8] network. This network was particularly trained for visual place recognition task using pairs of images collected from different places, times of day, and seasons. The key property that makes this network robust against viewpoint change in the idea of integrating the VLAD [9] feature aggregation scheme on the upper layer of the network. This allows combining the best of two worlds: powerful CNN features and unconstrained feature location with an image.

2.2 Evaluations of image matches

A solid experimental evaluation is a key component to confirm the correctness, relevance, and robustness of any approach. In this section, we describe the experimental evaluation setup and metrics that we use through the thesis to analyze the different properties of our visual place recognition approach.

2.2.1 Precision recall

Computing a precision-recall curve is one of the standard methods to estimate the performance of a visual place recognition approach. The terms precision and recall are frequently used in pattern recognition, information retrieval, and classification. The **precision** is defined as a fraction of relevant instances among the retrieved instances and **recall** is a fraction of relevant instances that have been retrieved over the total amount of relevant instances, see Figure 2.2 for an illustration. In context of classification, the dots (filled/unfilled) in the illustration denote classification instances, which come from ground truth data. Consider a task of determining whether there is a cat in the image, a filled dot represents the fact that there is a cat in the image and unfilled dot corresponds to a fact that there is no cat in the image. The area marked with a circle is the output of a classifier which determines if there is a cat in the image. As can be seen from the illustration, the classifier reports some images to contain a cat correctly (green area) and some images wrongly, i.e., the classifier reports an image to contain a cat but in fact there is no cat in the image (red area). These qualities are

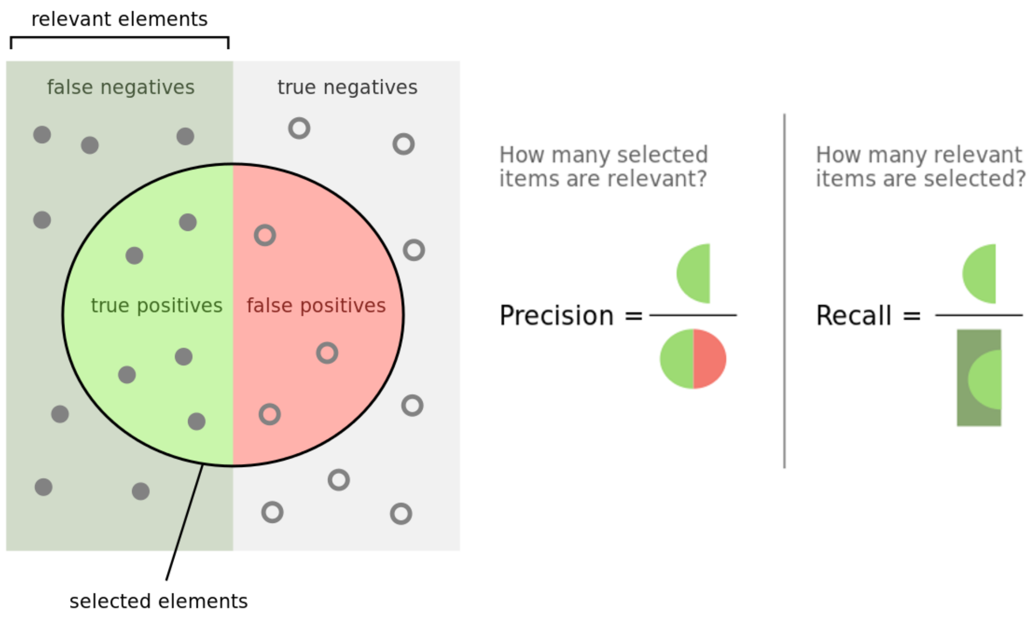


Figure 2.2: Illustrative example for estimating precision and recall. All the dots (filled / unfilled) correspond to the items to be retrieved. The ones in the circle correspond to a result of estimation, e.g., classification result. Courtesy: *Wikipedia*.

known as *true positive* (TP) and *false positive* (FP) respectively. Furthermore, if a classifier fails to find some of the images that exhibit cats, these decisions are called *false negatives* (FN) since it was a negative decision from a classifier and it is a false decision. Additionally, one can measure in terms of how good is the classifier of not finding irrelevant items, e.g., not finding a cat in the images where there is no cat. This is called a *true negative* (TN). Precision answers the question "how many of the retrieved items are relevant?", whereas recall shows "how many relevant items are selected?". The term precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.1}$$

The term recall is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.2}$$

Considering the previous cat classification example, the performance of the classifier on a collection of images is determined using two values, precision and recall. Since most of the approaches under evaluation depend on one or multiple parameters, varying these parameters may lead to variations in the output, e.g., different precision and recall for the cat classifier. Precision recall curve is one of the standard metrics to evaluate the behavior of an algorithm under changing parameters. It is obtained by computing the individual precision and recall values for varying input parameters of the system. An example of such curves can be

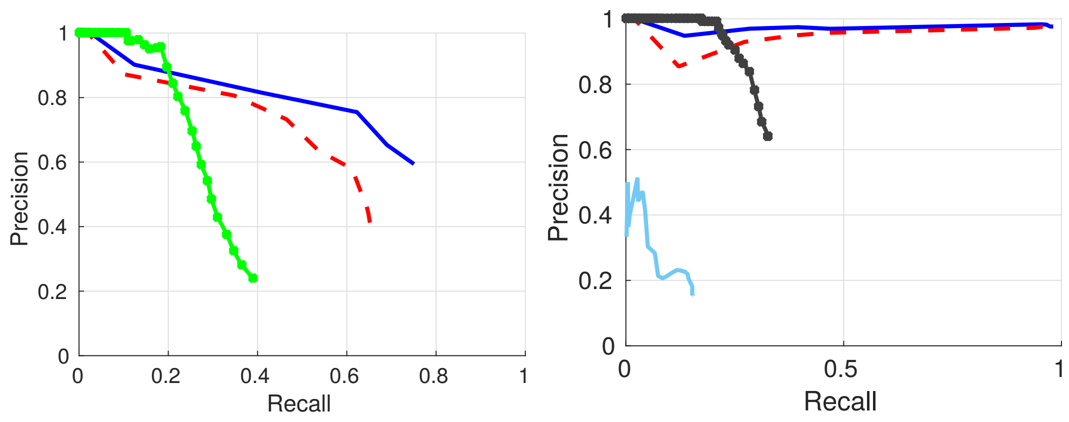


Figure 2.3: Examples of the precision recall curves to evaluate different approaches. The closer the precision-recall curve to the upper right corner of the plot, the better is the performance of an algorithm. Left: blue curve is closer to the top corner so the corresponding algorithm performs better. Right: Example of near perfect performance of two methods (blue and red) and poor performance of a light blue curve.

seen in Figure 2.3. Since both precision and recall are defined through a fraction, the maximum value that they can obtain is 1, corresponding to perfect precision and recall and the minimum value is 0. For example, a perfect cat classifier would report 100% precision with 100% recall. In other words, the closer the algorithm's curve approaches the right upper corner of the plot, the better is the performance of an algorithm, with perfect performance being exactly in the upper right corner. The left image shows a typical precision recall curves for three different algorithms. In this case, the blue algorithm performs better than the others since it is closer to the upper right corner and it reaches 60% precision over around 75% recall, whereas other algorithm reaches lower maximum recalls of 63% and 40% with lower precision. In the right image, one can see other examples of precision recall curves, where blue and red correspond to almost perfect algorithms and black and light blue exhibit a poor performance for the algorithms.

Precision recall curves are common tools to analyze the performance of the algorithm and thus are used frequently in this thesis. It should be noted, however, that depending on the requirements of algorithm the importance of the precision and recall can matter differently. For example, if a system should be highly conservative and only report true positives regardless of recall, e.g., only detect actual cats, e.g., no dogs classified as cats allowed, then the green algorithm in Figure 2.3 (left) is a better choice since it provides up to 15% recall on 100% precision whereas other approaches do not provide a precision of 100%.

We can naturally adopt the precision-recall metric to evaluate the performance of visual place recognition approaches by redefining the key terms. The typical task of visual place recognition is given an input image find the corresponding

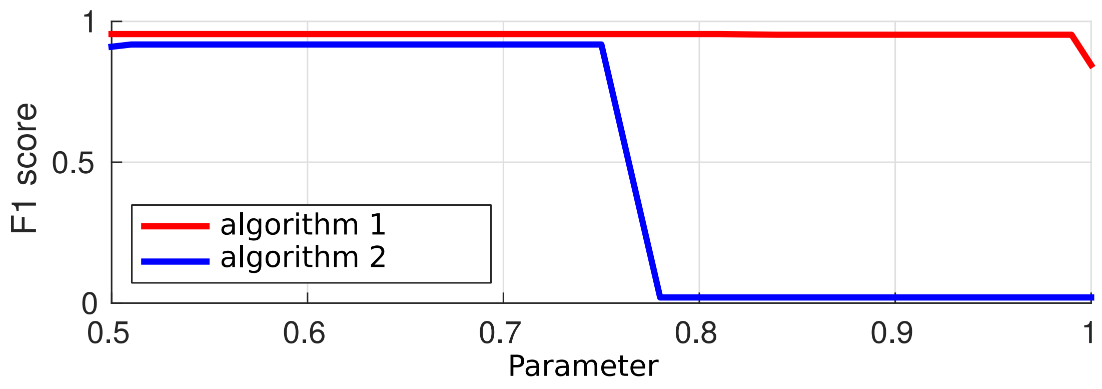


Figure 2.4: Example of two F1 curves generated by varying a Parameter in a given range. Algorithm 1 performs better than Algorithm 2, since it has higher F1 scores.

image in the database of images or report that there is no similar image in the database. In our work, we present a visual place recognition approach that reports for every input query image if there exists a single image in the database that represents the same place and which image it is. Whenever the algorithm makes a decision and the ground truth supports it, we consider such an image association as a *true positive* TP. If, however, the ground truth disagrees with this assignment than the decision is considered as *false positive* (FP). Additionally, our algorithm reports the cases where there was no similar image found in the database. If the place indeed does not exist in the database, our algorithm made a correct decision and we consider it as *true negative* (TN). If, on the other hand, the place exists, but our algorithm failed to detect is, we mark this decision as *false negative* (FN). Afterwards, we use the Equation (2.1) and Equation (2.2) to obtain precision recall curves.

2.2.2 F1 score

Precision-recall metric gives an estimate about the performance of an algorithm in terms of two quantities: precision and recall. As was noted before, depending on the desired properties of the system one may prefer to weight precision over recall or vice versa. One of the ways to evaluate an algorithm is to weight precision and recall equally. In this thesis, we use *F1 score*, which is defined as a harmonic mean of precision and recall, as:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.3}$$

To make a more informative decision about the performance of an algorithm, we evaluate the F1 score for a parameter range and then visualize the results in form of F1 curve, as in Figure 2.4. The F1 score takes the value between 0 and 1. It is more appropriate to use the harmonic mean to combine precision and

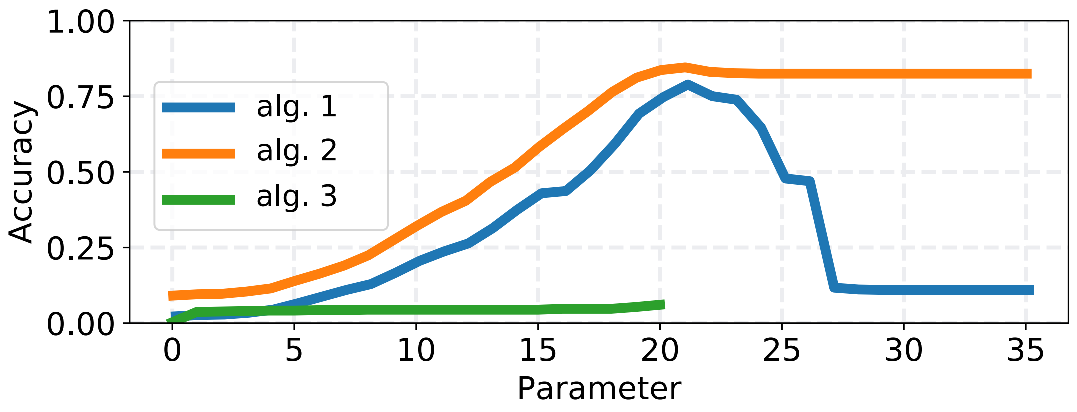


Figure 2.5: Accuracy plot example. Alg. 2 performs better than Alg. 1 and 3 because it achieves higher values of accuracy.

recall values, as oppose to the averaging, since both precision and recall are the rates and not actual quantities. The higher the F1 score is the better the average performance of an algorithm.

2.2.3 Accuracy

As stated before, precision-recall values as well as F1 score can be used to evaluate a performance of a visual place recognition approach. Both of these measures, however, do not include true negatives (TN) term. This means that these measures do not reward the algorithm for correctly not associating a place with a wrong place in the environment. This, however, is an important property of a robust place recognition approach and should be considered. We want our system not just to correctly recognize the previously mapped places, but also be able to estimate whenever a new place is being observed, i.e., that there exists no similar image in the database. To account for true negatives, we further compute the *accuracy* term as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.4}$$

Running an algorithm with varying parameter values result in obtaining accuracy curves, see Figure 2.5 for an example. The accuracy takes the values from 0 to 1 with 1 being the maximum accuracy. As can be seen from Figure 2.5, we compare three different algorithms and Algorithm 2 has a better performance since it achieves higher accuracy in comparison to Algorithm 1 and 3.

Part I

Visual place recognition

Chapter 3

Image sequences matching as a graph search problem

In robotics, the problem of visual place recognition plays an important role as a part of localization, loop closure, and SLAM in general. A promising way to approach this problem is by analyzing a stream of visual data rather than individual images, since the robot continuously obtains data from its sensors in temporal order. In this thesis, we approach a problem of recognizing similar places by comparing streams, i.e., sequences, of images rather than individual images of the places. This turns visual place recognition problem into a task of matching image sequences. More precisely, given two sequences of images taken in different points in time referred to as reference and query sequence, the task is to decide for every image in the query sequence if there is a matching image in the reference sequence and which one it is.

One of the approaches for finding a matching image from a database of reference images is to compare the query image to all images in the database. By performing this operation for every image in the query sequence, we obtain a so-called *matching* or *cost matrix*. An example of the cost matrix is depicted in the Figure 3.1. Every element of this matrix stores the "matching cost", cost defined based on image similarity. For more details how to compute the matching cost see Section 2.1. The brighter the squares in Figure 3.1 are, the smaller is the corresponding matching cost, that means the more visually similar the images are. Darker squares correspond to the fact that the images visually look dissimilar. An example in Figure 3.1 (left) shows two image sequences, each containing four images. Here, the reference sequence was collected in winter whereas the query sequence was collected in summer. Visually both sequences appear quite different. For example, in the winter sequence, the trees lack foliage in comparison to the summer one and we can the see the buildings behind the tree better. This changes the visual appearance of the place together with different camera

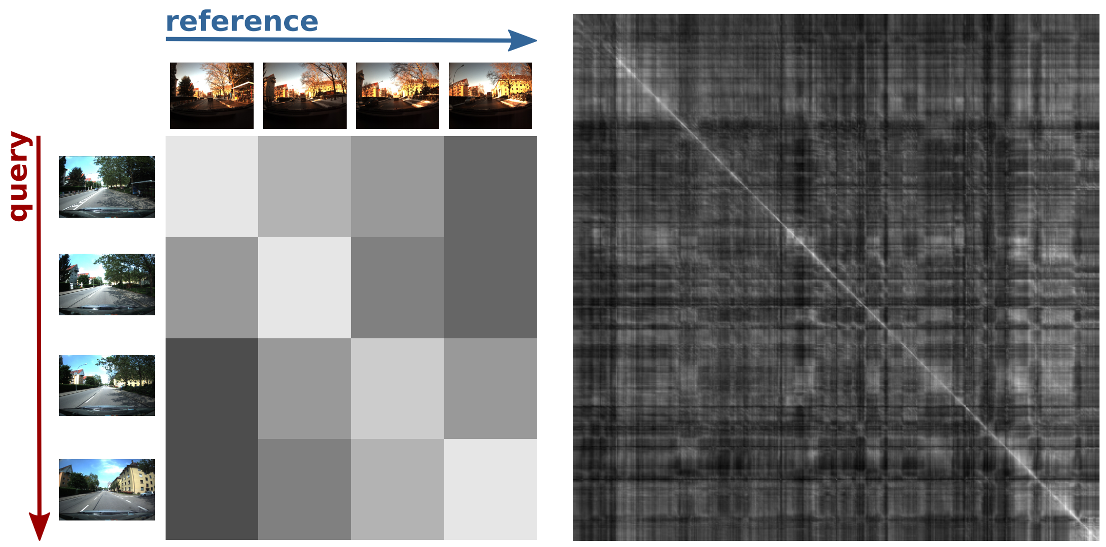


Figure 3.1: Left: A toy example of a cost matrix. Brighter values correspond to the smaller matching costs, darker values to the bigger matching cost. Right: Cost matrix from a real world dataset with sequences consisting of 500 images both query and reference sequences.

placements and natural illumination changes. In this example the trajectories are artificially synchronized, i.e., the first image of the query trajectory shows the same place as the first image in the reference trajectory as well as second image in query corresponds to the second image in the reference, etc until the last. These image correspondences between the sequences lead to a brighter pattern on a diagonal of the cost matrix. Thus, to solve the sequence matching problem, we are interested in finding the bright patterns in the cost matrix, since they are likely to correspond to the images that represent the same place. Figure 3.1 (right) shows a cost matrix from a real word dataset. In this dataset, both image sequences are image-to-image synchronized, which should result in a perfectly diagonal bright pattern as well as in the previous toy example. However, in addition to a diagonal pattern, we can also spot the brighter parts in the cost matrix, particularly towards the right of the cost matrix. This indicates that there are other images in the reference dataset that look visually similar to the query image but do not actually correspond to the same place. This can happen due to inability of a used image descriptor to unambiguously describe the images or effects like glare or occlusion. Therefore, selecting an image from the reference database with the smallest matching cost leads to poor results in visual place recognition. For example, Figure 3.2 shows the result of selecting the best match from the reference dataset based on the smallest matching cost. The right matrix is constructed using the popular HOG features [35], which are less discriminative for this example than features computed through a CNN from which the left matrix is constructed. Even though it can be seen that the results in the left image better than in the right, i.e., more red dots belong to the bright line pattern, in

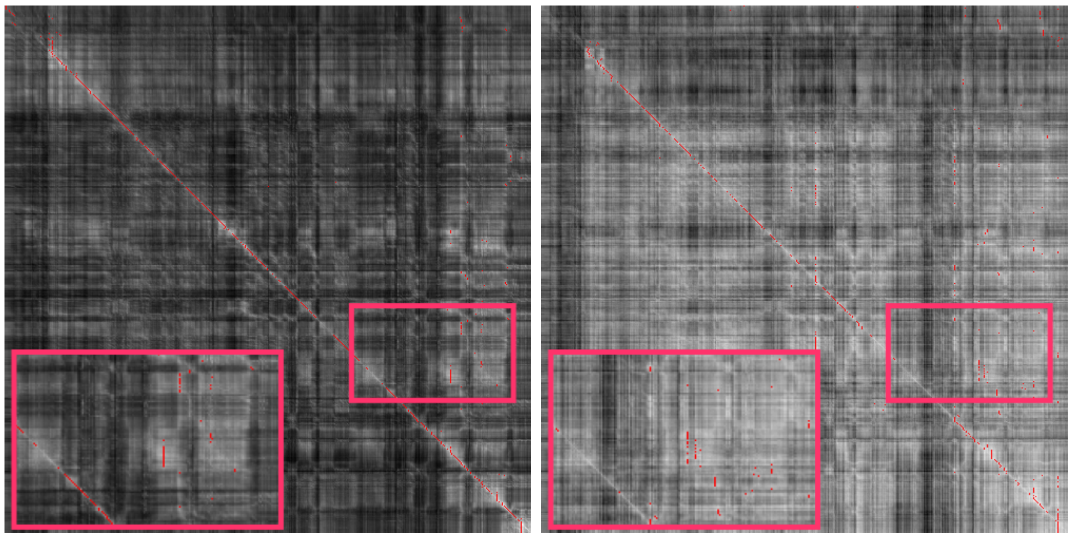


Figure 3.2: The strategy of selecting an image match with the lowest matching cost (brightest pixel in the row) fails to reliably find correct matches, especially when the features are not discriminative enough, i.e., the difference between the correct pattern and the rest of the matrix is not big enough, as can be seen in the right matrix. Red dots correspond to the an image pair (query image, dataset image) that are found as matches.

general selecting the match with the smallest cost is not suitable for visual place recognition in changing environments. To robustly establish correct image associations under strong appearance changes, Naseer *et al.* [99] proposed to perform a graph based search for image correspondences within the cost matrix. The authors build a directed acyclic graph based on the precomputed cost matrix and solve the image association problem through solving the network flow problem. We approach the problem of visual place recognition in a similar way as Naseer *et al.* [99] by constructing a graph and performing a shortest path search in this graph. The shortest path in such a graph represent the set of image associations between the image sequences with smallest accumulated matching cost. Thus, the shortest path will prefer to go over the nodes with high similarity preserving the sequentiality of the data as well as possible. We discuss the construction of the graph in the following section.

3.1 Constructing a graph

In this section, we will describe how to formulate a pattern search in the cost matrix as a graph search problem. To be able to perform the matching between the reference and query sequence, we define the query image sequence to be an ordered set of images $\mathcal{Q} = \{q_i\}_{i=0,\dots,Q-1}$ and the reference $\mathcal{D} = \{d_i\}_{i=0,\dots,D-1}$ respectively, where $D = |\mathcal{D}|$ and $Q = |\mathcal{Q}|$ are the number of images in the reference and query sequence respectively. As a data structure, we use a directed acyclic

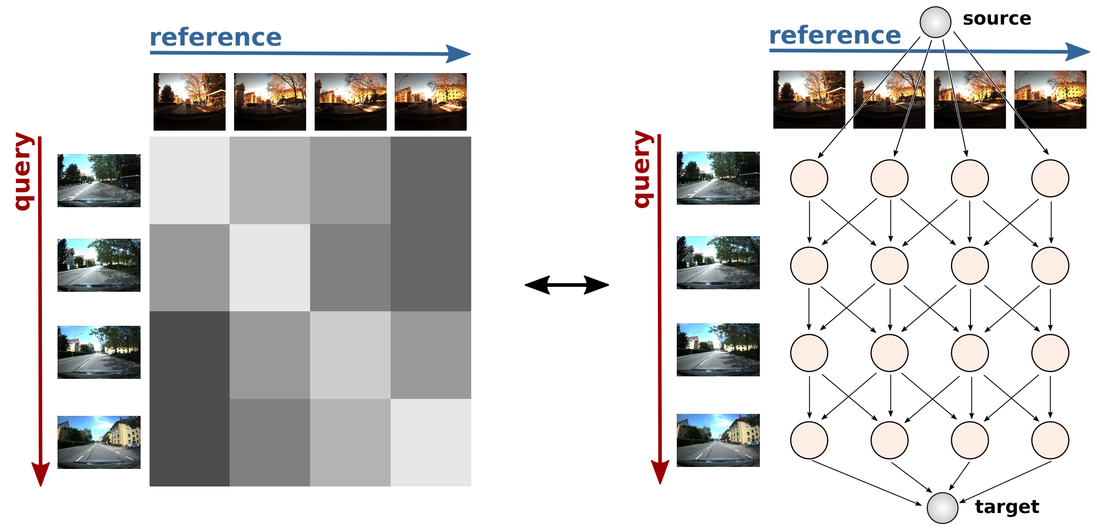


Figure 3.3: How to build a graph structure given a cost matrix. Yellow circles denote nodes in the graph, arrows denote transition between the nodes.

graph $G = (X, E)$, also called DAG. A node in the graph $x \in X$ corresponds to the fact that an image from a query sequence is compared to an image from the reference sequence, see Figure 3.3 for visualization. The edges between the nodes correspond to the possible transitions between the image pairs. These transitions encode a possible movement of the robot that led to obtaining such image sequences in the first place. Recognizing the correct transitioning between the images leads to faster recognition of the next images in the query sequence. Graph structure is an elegant way to preserve the sequential information while comparing two sequences. The concept of edges can naturally connect the matching decisions between image pairs. In this section, we present how to formulate the task of matching image sequences as a graph search problem. The construction of the graph starts from a *source* node x^s. In typical graph search problem formulations, the source node marks the place, where the search should start. The source node is connected to all possible pairs for the first image in the query sequence, since in the beginning it is unknown what a potential image matching pair could be. The source node gets connected to the matching nodes x_{0*} through the set of edges $E^s \in E$:

$$E^s = \{(x^s, x_{0j})\}_{j=0,\dots,D-1}. \tag{3.1}$$

To mark the termination of the matching process, all the nodes that correspond to the last query image q_Q are connected to the *target* node x^t through the set of edges $E^t \in E$:

$$E^t = \{(x_{Qj}, x^t)\}_{j=0,\dots,D-1}. \tag{3.2}$$

The edges in between the nodes in the graph model possible transitions between the matching image pairs. The following paragraph describes how establishing

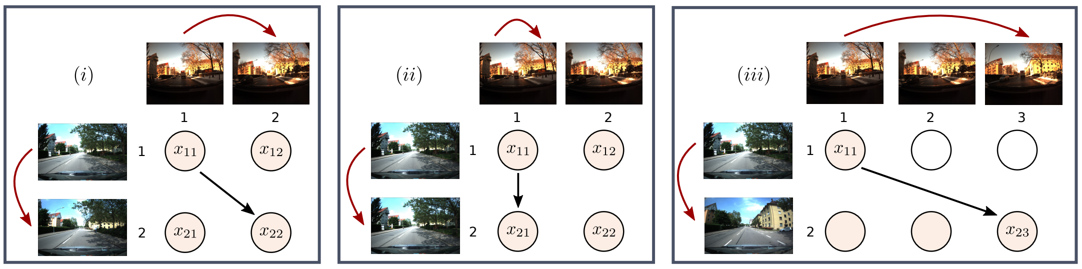


Figure 3.4: Constructing the edges between the matching nodes based on the potential image transition within the query and reference sequences. Left: query and reference cameras move at the same speed. Middle: query camera is staying whereas reference is moving. Right: query camera moves twice as fast than reference one.

edges between the nodes preserves the sequential information of two sequences. There are several type of situations that we would like to capture:

(i) the cameras in query and reference sequences move with approximately the same speed and frame rate, so that they capture roughly the same places at similar relative index increment;

(ii) the camera in the query sequence is not moving whereas the camera in the reference sequence is moving;

(iii) the speed (or frame rate) of the reference sequence is higher/lower than the speed of the query sequence.

Consider type (i), when the cameras moving with the same speed, see Figure 3.4. Let us assume that we know that image 1 in the query sequence was taken at the same place as image 1 in the reference sequence. If we then move from image 1 to image 2 in the query sequence, we can expect to move from image 1 to image 2 in the reference sequence, given the sequentiality of the image stream is preserved. The edge between the node x_{11} and x_{22} makes sure that this transition is possible within the graph structure. Type (ii) situation, e.g. the camera is standing still in the query sequence but moving in the reference sequence, can occur within the normal city drive, for example, in presence of traffic lights or road crossings. To capture this situation, we connect the nodes x_{11} with the node x_{21} as in the Figure 3.4 (middle). The third situation (iii), which might happen if the camera in the query sequence is moving faster than the camera in the reference sequence. This leads to the effect that image 2 from the reference sequence does not have a corresponding image in the query sequence, since going from image 1 to image 2 in query corresponds to the transitioning from image 1 to 3 in reference sequence as in Figure 3.4 (right). The more different speed or frame rate variations we would like to capture, the more outgoing edges every node must have. The number of outgoing edges is defined by a parameter K, which we refer to as the *fanout*

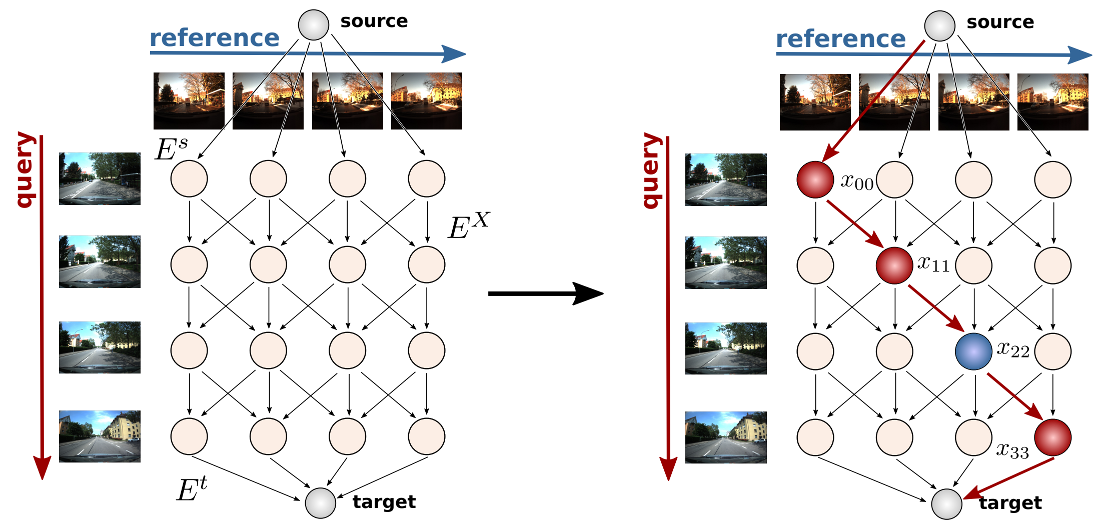


Figure 3.5: We establish correspondences between image sequences by searching for the shortest path in the graph. Red nodes represent real matches, blue node represent a hidden node (the image pair is not considered as a match).

throughout this work. In the graph shown in Figure 3.3 (right), one may also see edges going to the left sides of the nodes. These edges cover situations, in which the query camera is moving backwards with respect to the reference sequence. To cover these situations, we connect the matching nodes with a set of edges $E^X \in E$:

$$E^X = \{(x_{ij}, x_{(i+1)k})\}_{k=j-K,\dots,j+K}. \tag{3.3}$$

The sets E^X, E^s, E^t form the whole set E of the edges in the graph $E = E^s \cup E^X \cup E^t$. To compute the least-cost path on graphs from source to target, we need to specify the weights on the edges. All edges from the set E^X have the weights that correspond to the entries of the cost matrix that correspond to the node $x_{(i+1)k}$ in Equation (3.3). For example, if the entry $(1, 2)$ of the cost matrix has a value 0.2, then all incoming edges of the node x_{12} will have weight 0.2. This weight represents the cost of matching the images 1 and 2. The edges in the set E^t have zero weight, since they are only used to model the termination of the search process and every node that corresponds to the last image in the query sequence should lead to a terminal node with the same cost. After the graph construction is completed, we perform the search for the least-cost, here also called shortest path. Since we work with directed acyclic graph with unbounded non-negative weights, we can apply Dijkstra algorithm to find the shortest path from source node x^s to the target node x^t. All nodes that belong to the shortest path, except source and target nodes, represent established image correspondences. For example, if the resulting path goes through the nodes x_{11}, x_{22}, x_{34} this means that image 1 from query corresponds to the image 1 from the reference sequence, query image 2 corresponds to reference image 2, and query image 3 corresponds to image 4

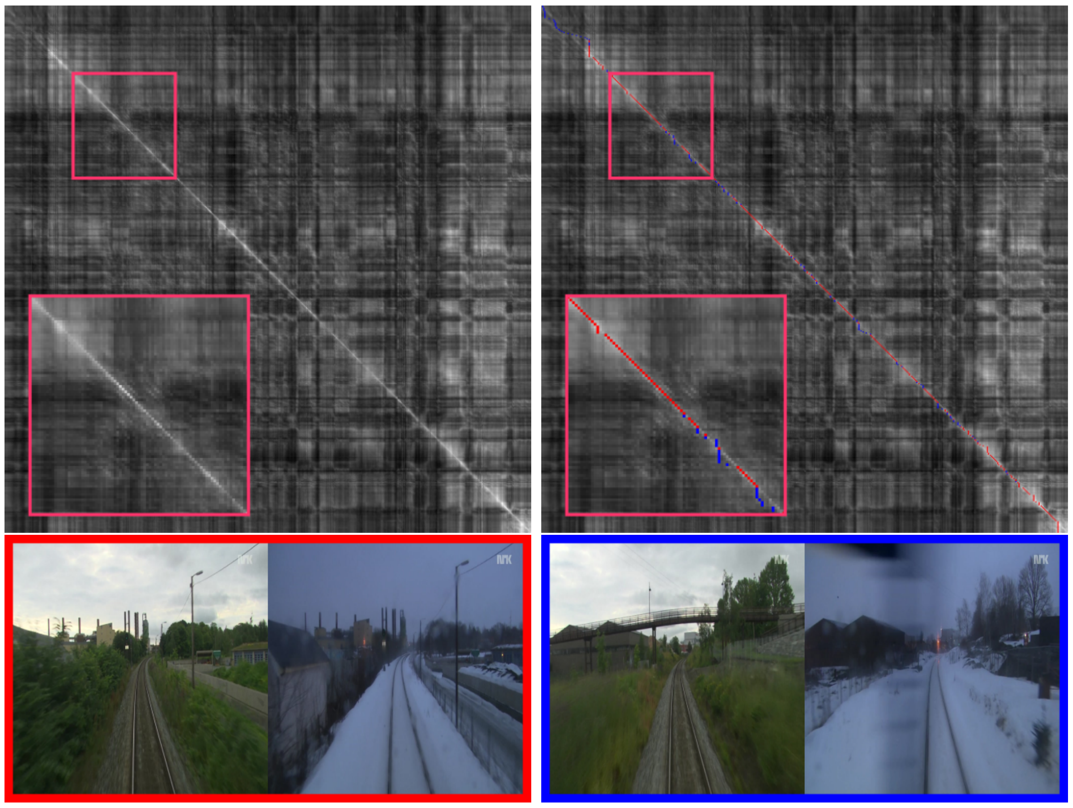


Figure 3.6: Up: Matching results that the proposed graph structure is able to obtain on a challenging Nordland dataset. Down: Image pair that corresponds to a real node (red, left), image pair that corresponds to the hidden node (blue, right).

from the reference sequence. Whenever we compare two images, we obtain a cost representing how visually similar two images are. What this cost does not tell us is whether the images actually represent the same place or not. To distinguish between images that represent the same place from those that do not based on matching cost, we use a cost threshold m to which we refer to as *non-matching* cost. The non-matching cost specifies that every image pair where matching cost is smaller than m is considered as a *match*, whereas if the matching cost exceeds m, we consider that pair of images to be from different or visually dissimilar. Sometimes it may happen that due to severe visual appearance changes, occlusion or glare, the images that actually represent the same place, have matching cost exceeding the non-matching threshold. To be able to consider the nodes that support sequential information, but have high matching cost, we refer to those nodes as *hidden*. Hidden means that they are the part of the shortest path, however, they do not result in valid image correspondences. Figure 3.5 shows the case, where the node x_{22} supports the path hypothesis, however the matching cost is higher than m which makes it a hidden node. The result of the shortest path search is x_{00}, x_{11}, x_{33}, which means that the corresponding images are considered to match and node x_{22} is skipped.

3.1.1 Real world example

By deploying the proposed graph structure, our image sequences matching algorithm is able to establish data association between the image sequences that exhibit dramatic visual appearance changes. To evaluate the performance of our algorithm, we use a popular public dataset called *Nordland dataset*[1]. This dataset consists of four videos collected by the Norwegian Broadcasting Corporation NRK. Each video covers a 728 km train ride between two Norwegian cities across four seasons, i.e., winter, summer, fall, and spring. The particular feature of this dataset is that all four videos are time- and pixel-wise manually synchronized, which never happens in reality, but makes this dataset valuable for evaluations. For the first experiment, we sampled 500 images with 1 fps from the summer and winter sequences respectively and computed the similarity matrix. A couple of example images from this dataset are depicted in the Figure 3.6 in the second row. Our algorithm takes as an input the cost matrix that is depicted in Figure 3.6 (left), constructs the graph, and searches within it producing the path hypothesis depicted in Figure 3.6 (right). Red pixels represent found image correspondences and blue ones represent the fact that the data associations are supported by the sequentiality of the data, however, reveal a high matching cost. In the second row of the figure, one can see the image associations that result from a reported match (red, left) and an image pair that was not found as correspondence, i.e., corresponds to the hidden node (blue, right). The hidden pair represents the same place, however, this fact is hard to recognize not just due to the seasonal and lightning changes, but also due to the fact that an important structure as a bridge is missing in the winter image. An additional factor that increases the matching cost and thus makes the images dissimilar is the wiper blade trace in the middle of the bottom right image.

In this section, we presented an approach to find image correspondences between two image sequences given a precomputed cost matrix based on ideas taken from Naseer *et al.* [99]. By using the proposed graph structure, we are able to preserve the sequentiality of the input data within the search procedure and robustly find the data associations despite strong visual changes. We construct a full matching matrix from the two image sequences, since there is no additional prior information available about the potential camera location. Adding this additional prior information leads to narrowing the search space and reducing the number of image matching operations, which can be an expensive operation to perform. In the next section, we will describe how to incorporate the pose prior information, that for example can come from a rough GPS prior, into the graph construction procedure and thus to reduce a computational overhead of

[1]http://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute- season-by-season

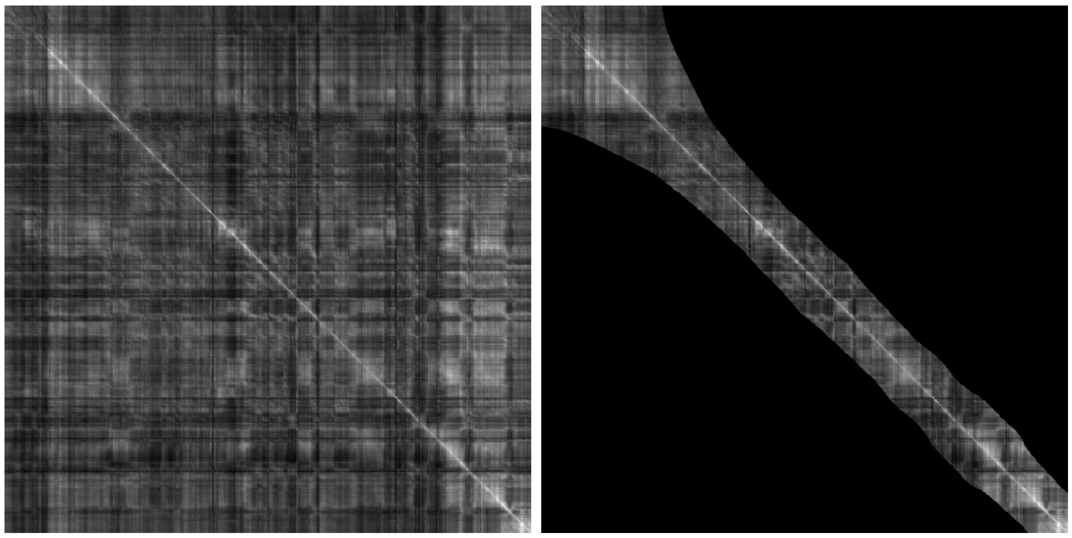


Figure 3.7: Left: Full matching matrix. Right: Sparse matching matrix considering a 1 km location prior.

constructing the full matching matrix.

3.2 Efficient matching using pose priors

In this section, we address one of the key limitations of the previous approach when it comes to long term operation. The previous approach relies on a dense image matching matrix with the size equal to the number of images in the reference sequence times the length of the query sequence. This is a bottleneck when dealing with large scale datasets. In this section, we present a technique that avoids building up the full matching matrix by exploiting a rough pose prior, for example stemming from a low quality consumer GPS receiver. We achieve this by proposing a modified version of the data association graph, which is used for identifying the sequences of matched images. As a result of that, only a small fraction of the computationally expensive image comparisons, at least if executed in large quantities, needs to be conducted. As a by product of the new graph topology, we can effectively deal with loops. This was not possible within a shortest path search presented in the previous section. Furthermore, we do not need to formulate a network flow problem to deal with loops as proposed by Naseer *et al.* [99].

Throughout this section, *we assume that a rough pose prior is available*— from any global pose prior source such as a GNSS system or similar. Given this pose prior, we can avoid instantiating the majority of nodes in the graph — a node is only needed if the distance between the sensor locations was less than the prior, for example, $d_{pose} < 500\,m$. As a result of that, only a fraction of the matching matrix C needs to be computed, which substantially limits the total number of image

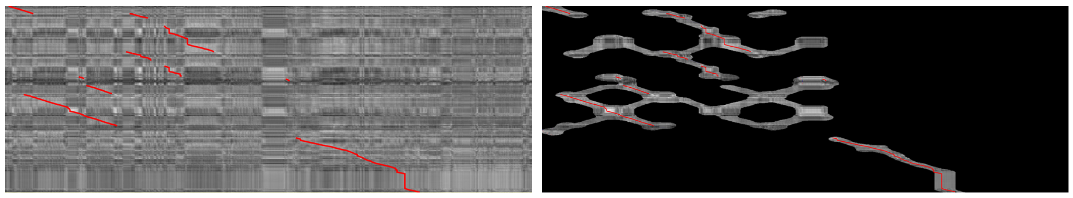


Figure 3.8: Realistic scenario. The image sequences are not synchronized, which results in disconnected components when considering a rough GPS prior.

comparisons that have to be conducted. Figure 3.7 shows a full cost matrix C for the Nordland dataset and the respective sparse matrix C'. The sparse matrix is obtained by considering for every image image in the query sequences only those reference images that are located within a $1\,km$ radius. The black pixels in the matrix visualizations represent the fact that an image pair has not been compared, i.e., the node has not been instantiated. As can be seen, by adding the pose priors, we are able to leave out a substantial amount of expensive image matching operations and still preserve the desired pattern in the cost matrix. For the image sequences of this type, where the query trajectory roughly follows the reference one, the approach presented in previous section works well. Note that when images are synchronized, i.e., the corresponding image identifiers in query and reference sequences are the same, we expect a brighter diagonal pattern in the cost matrix. However, if we consider more realistic scenarious, where it is not guaranteed that the images in both trajectories are synchronized, adding the pose priors and following the proposed graph construction procedure may lead to the fact that the data association graph consists of unconnected components. This prevents in turn the deployment of the shortest path algorithm. Exactly this case is depicted in Figure 3.8. Here the vehicle in the reference dataset visits the same place in the environment multiple times, which results in repeated components along the columns, as well as query dataset, which results in the repeating components along the rows. To robustly deal with such situations, we propose an adapted graph construction scheme that is able to connect the disjoint components and then allows us to proceed with the shortest path search. Additionally, the use of the pose priors marks explicitly if the same place in the environment was visited multiple times. This can be recognized whenever there appears a repeating component along the rows of the cost matrix. We determine multiple disjoint components along a row by treating a matrix row as a continuous array of values. Then, we search for subparts of this array separated to by zero values, which corresponds to the black pixels or no information in the cost matrix. The subparts form a disjoint components and reveal similar places within the reference dataset. We can directly consider this fact while constructing the graph and in this way to robustly deal with loops.

Our main data structure is still a data association graph $G = (X, E)$, where

X is a set of nodes and E is a set of edges. The set of nodes X consists of four type of nodes: the *start state* x^s, the *goal state* x^t and the potential matching nodes x_{ij}. A matching node x_{ij} represents the fact that the images i and j match, whereas a hidden node models the fact that no matching between both images could be found. Visiting a node comes at a cost, which is proportional to the similarity of the images given the global HOG descriptor. Then, the localization problem can be described as a search for the shortest path through the graph.

3.2.1 Edges

As in Section 3.1, the set of edges E specifies, in which way the nodes can be traversed. In this extended approach, we also use the three types of edges E^s, E^t, E^X, which are similar to those defined before. In addition to that, we define two further types of edges (E^d and E^q) to appropriately exploit the pose priors, so that $E = \{E^s, E^t, E^X, E^d, E^q\}$. The construction of the graph starts with edges E^s that connect the start node x^s with a set of matching and hidden nodes, defined as

$$E^s = \{(x^s, x_{fj})\}_{j \in N(f)}. \tag{3.4}$$

In Equation (3.4), f refers to the index of the first image in the query sequence for which a database image exists that has been taken in a distance smaller than d_{GPS} from f. The term $N(i)$ is a set of indices of neighbouring images from the database sequence for a query image i and is defined as:

$$N(i) = \{j \mid j \in \mathcal{D} \wedge \operatorname{dist}(i, j) < d_{pose}\}, \tag{3.5}$$

where $\operatorname{dist}(i, j)$ is distance between the location at which the images with index i and j have been taken according the pose prior. Intuitively, we connect the source (or starting node) with ony those images from the reference sequence that lie in direct proximity to the query image according to a rough pose prior. The next set of edges is E^X. It models the connection between the nodes as:

$$E^X = \{(x_{ij}, x_{(i+1)k})\}_{\substack{i=0,\dots,Q-1,\\ j \in N(i),\\ k=j,\dots,(j+K) \text{ with } k \in N(i+1)}} . \tag{3.6}$$

These edges model the potential transition from one image in the database sequence to another, where the transition between subsequent images in query sequence occurs. This edges are similar to the the edges from set E^X defined in Section 3.1, however, only connects the nodes whenever the images agree with the pose prior. The value of the fanout parameter K specifies the possible paths that are exiting from a node. The difference to the previous formulation of E^X

is the fact that x_{ij} are only connected to those children nodes that are located within a specified neighbourhood.

The set of edges, E^t, connects nodes created for the last query image l for which $N(l) \neq \emptyset$ to the goal state x^t:

$$E^t = \{(x_{lj}, x^t)\}_{j \in N(l)}. \tag{3.7}$$

Traversing such an edge corresponds to the end of the matching process as the goal state has been reached.

The presented graph structure considers the neighborhoods $N(i)$ and encodes the pose prior constraints. Exploiting pose information yields a serious reduction in the number of image comparisons that have to be performed. In analogy to the dense matching matrix C, this corresponds to a *sparse matching matrix* C'. Thus, the set E^X in this formulation can be seen as a set of edges that connects the consecutive rows of C' as this corresponds to the temporal order of the images in the query image sequence. The proposed structure, however, can lead to multiple disconnected components in C', see also Figure 3.9. These components in the matrix lead to an unconnected data association graph. Thus, the current graph topology may prevent to find the shortest path from the start to the goal.

In order to compute the matching sequence as a shortest path problem, we need to connect the individual components so that every node has at least one parent node and one child node. For this, we use the new sets of edges E^d and E^q to connect the nodes among components. Two situations can occur in this context.

First, a component can appear if the vehicle visited a place more than once while recording the database images. This leads to nodes that have no parent. To reconnect such components, we introduce a new hidden node $\breve{x}_i^*$, see Figure 3.9. This node serves as a connector node and does not correspond to any pair of images, so it can only be in a hidden state. We re-connect the components via $\breve{x}_i^*$ using edge set E^d defined as:

$$\begin{aligned} E^d = \{(x_{(i-1)k}, \breve{x}_i^*), (\breve{x}_i^*, x_{ij})\}_{k \in N(i-1)} & \\ \forall x_{ij} \text{ with } \mathrm{par}(x_{ij}) = \emptyset, & \end{aligned} \tag{3.8}$$

where par(x) is the set of parents of x, i.e., all nodes that have an outgoing edges to x. The newly introduced node $\breve{x}_i^*$ exists once per row and connects all the components, which represent the same place in a real world. See Figure 3.9 for an illustration. Note that in such a situation there are nodes x in a component that do not have any outgoing edges, i.e., $\mathit{child}(x_{ij}) = \emptyset$. We refer to nodes without children as frontier nodes x^F.

Second, a component can appear if the vehicle visits an area that has not been mapped in the database and then returns to a known place. This situation can

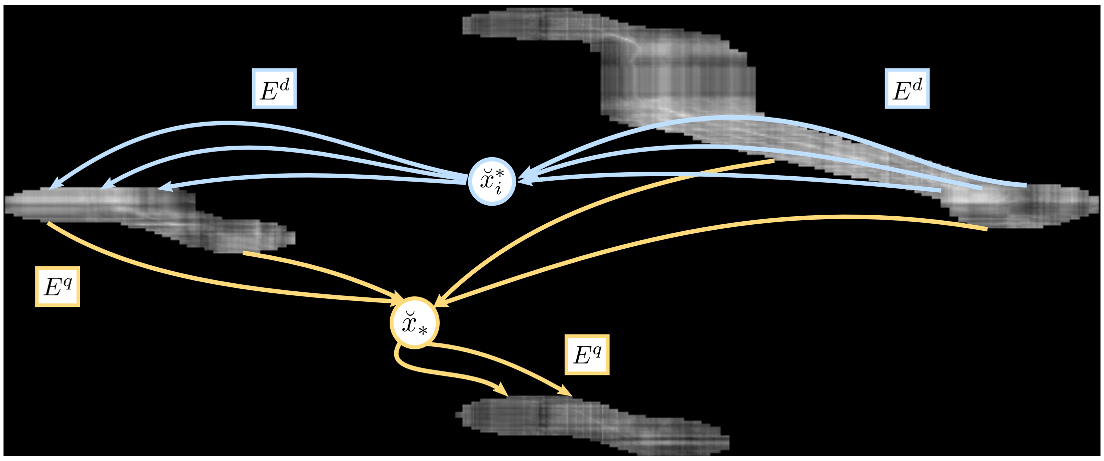


Figure 3.9: Illustration of a sparse matching matrix C' and the process of connecting the separate components using E^d and E^q.

easily be detected if an image i from the query set has no neighbors within the range of the pose prior, i.e., $N(i) = \emptyset$. At the point in time when query images are again close to database images, we connect them through a new hidden node $\breve{x}_*$ and the edge set E^q defined as:

$$E^q = \{(x_{i'j'}, \breve{x}_*), (\breve{x}_*, x_{ij})\}_{x_{i'j'} \in x^F} \quad \forall x_{ij} \text{ with } \mathrm{par}(x_{ij}) = \emptyset. \tag{3.9}$$

Intuitively, the edge set E^q connects the nodes from the frontier $x \in F$ with the nodes in the first row of the new component. Afterwards all the elements from the set x^F are removed as they now have a child node. The node $\breve{x}_*$ exists for every query break, i.e., subsequences, where query dataset deviates from the area mapped in database.

3.2.2 Edge costs

So far, we defined the vertices and edges of the data association graph but have not specified the cost associated to an edge. As finding the best matching sequence will be approached using a shortest path, the costs are associated to the ability to match two images.

The costs for sets E^s, E^t, E^X stay the same as in the previous formulation. Namely, the costs for E^t are set to zero, since they are only needed to terminate the search. The costs for E^s and E^X depend on the values of the entries in the matrix C'.

The edges E^d enable the graph search to treat multiple images from the same places in database alike. Thus the costs for edges from any node to $\breve{x}_i^*$ is zero. This enables free transition between the parts of the sequences that represent the same place in the environment. The costs for the edges from $\breve{x}_i^*$ are the values from matrix C' that correspond to the entering node.

For the edges in E^q, we use the following cost

$$w = m(i - i' - 1) + c'_{ij}, \tag{3.10}$$

where i' is a row index of the node $x_{i'j'}$ from the frontier set $x \in F$, i is a first row index of the new component and c'_{ij} is the value of the ij in the matrix C'. Intuitively, the weight w for an edge in E^q takes into account the number of query images that were skipped, or in other words traversed through the hidden nodes with the non-matching cost μ, and the cost c'_{ij} of entering the node x_{ij} within the new component, taken from the cost matrix C'. This definition of the cost replicates the cost that we would generate if using the *dense* matching matrix and moving between the components through hidden nodes. Thus, the cost is proportional to the distance in rows between the components times the cost of traversing a hidden node.

3.2.3 Normalization of the edge costs

As mentioned by Milford and Wyeth in [92], it is important to normalize the matching cost in the matrix C and thus C'. We apply the normalization used in the implementation by Naseer *et al.* [99], which normalizes the cost values by the mean of cost values over each column. As we do not compute the full cost matrix C due to the exploitation of the pose prior, we cannot compute the same normalization. We therefore approximate it using sampling. In more detail, we sample fixed number of additional image pairs (in our implementation, we use 30 additional image pairs) from the same column and use this for obtaining an approximation of the normalization constant. We compute the normalization constant by taking into account the mean μ_j^{known} of the known values along each column j from the matrix C' and the approximated mean μ_j^{sampled} of unknown values for the column j. We compute the μ_j^{sampled} by computing a fixed number of image comparisons from the unknown parts of column j. Then, the normalization constant in our formulations is computed by:

$$z_j = \frac{\mu_j^{\text{known}} n_j + \mu_j^{\text{sampled}}(Q - n_j)}{Q} \tag{3.11}$$

where n_j is the number of known values in column j and Q the number of query images. Afterwards, we normalize every column of matrix C' with the respective normalization constant z_j.

3.3 Complexity

For the complexity analysis, we assume that the covered areas are substantially larger than the GPS range, so that for all query images, only a bounded number of elements from the database is within the d_{GPS}. The highest complexity is the task of finding the images in the database that have been taken near a given location according to the pose prior. We achieve this through a kd-tree, yielding a logarithmic complexity in the size of the database. Overall, this results in $\mathcal{O}(Q \log D)$. Due to the directed acyclic graph structure, the shortest path can efficiently be computed via topological sorting. This yields a complexity of $\mathcal{O}(|X| + |E|)$.

3.4 Experiments

The evaluation is designed to illustrate the performance of our approach and to support the following statements:

(i) we can exploit GPS pose priors to substantially reduce the computational load of the image matching process;

(ii) we can naturally handle loops without the need of using network flow algorithms as proposed by Naseer *et al.* [99], and

(iii) we can either improve the matching results or perform comparably to our previous work.

All our experiments have been conducted using real world data, recorded in summer and winter. The data has been obtained with a bumblebee camera mounted in a regular car. The algorithm works with an image resolution of 1024×768, where no cropping, undistortion or other preprocessing is done. Examples for matching image pairs from the datasets can be seen in Figure 3.11. We evaluate the performance of our algorithm by precision-recall curves, which are computed based on manually labeled ground truth image matches. We calculate precision as $\frac{\text{TP}}{\text{TP+FP}}$ and recall as $\frac{\text{TP}}{\text{TP+FN}}$, see also Section 2.2. A match is considered as a *true positive* (TP) if the found match and the manually provided match differs by up to three images within the sequence. If an image pair is not within the specified boundaries then it is considered as a *false positive* (FP). All the ground truth pairs that were not found by algorithm are considered as *false negatives* (FN). To obtain the precision-recall curves, we vary the parameter μ from small to large values. If μ takes a value that is smaller than the smallest element in the matrix, all potential matches will be rejected. With increasing values for μ more and more potential matches will be accepted. We compare the performance of our

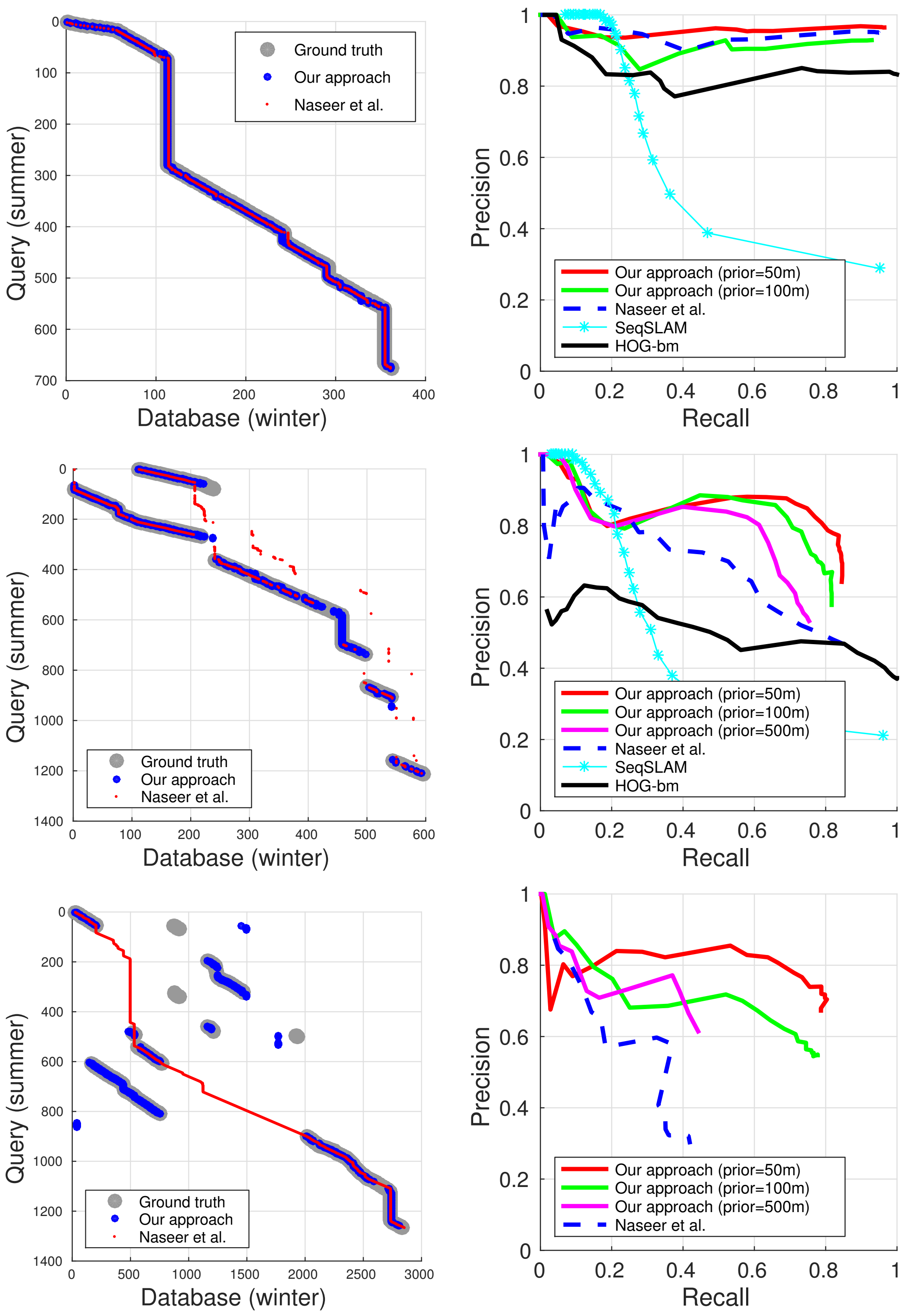


Figure 3.10: Experimental results on 3 datasets. The images in the first column show the matches, including ground truth and the plots in the second column show the precision recall plots. *First row:* Comparison of our approach the method Naseer *et al.* [99], openSeqSLAM and a heuristic that always selects the best match in C on a dataset that consists of a sequence of 3 km. *Second row:* Comparison between the same approaches on a dataset containing a loop in a query sequence. *Third row:* Comparison to the method of Naseer *et al.* on a third dataset containing several loops in database as well as in query.

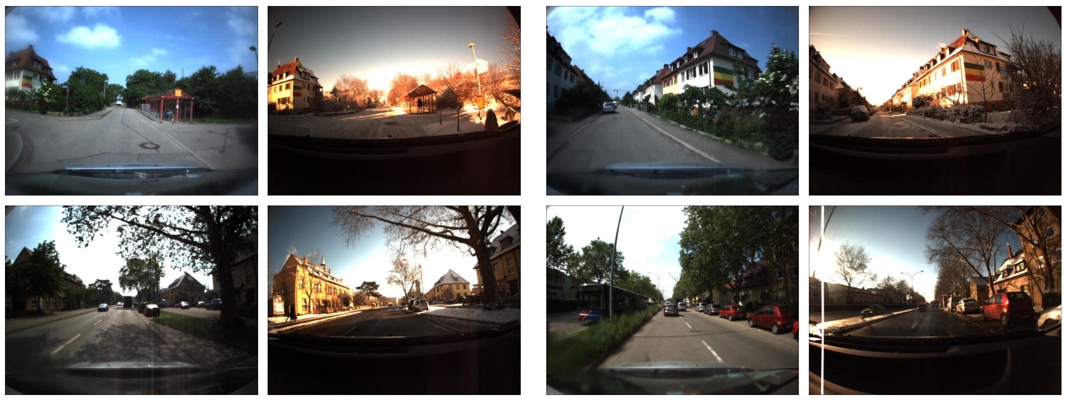


Figure 3.11: Four example image pairs from the database and query set illustrating the perceptual change over the seasons.

approach to the approach of Naseer *et al.* [99] that formulates the image sequence matching problem as a network flow problem. Furthermore, we evaluate it with respect to the state of the art method for visual place recognition in changing environments SeqSLAM proposed by Milford *et al.* [92]. SeqSLAM finds the pattern in the precomputed cost matrix by sequentially fitting the line. We use the open source version OpenSeqSlam provided by the authors. The results are labeled as "SeqSlam". Additionally, we compare our approach to brute force search, where for every row of the cost matrix we select the image from the database with the smallest matching cost, referred to as "HOG-bm".

The first experiment is designed to show that our approach performs comparable to the approach [99], which constructs the full matching matrix and this is computationally much more demanding. Figure 3.10 presents the matching results for three datasets of different lengths and camera trajectories of different complexity. The first dataset, depicted in the first row of the figure, consists of a query image sequence of 676 images that roughly follows the trajectory reference (database) trajectory of 361 images. The fact that a query sequence follows the same trajectory as the reference one can be inferred from a roughly diagonal/ straight line pattern of the ground truth results, except for the part from 100 to 300 query image, when the car was staying on the traffic light, see right image. The performance of our search approach achieves up to 95% precision for 95% recall in case of 50 m GPS prior, see left image, and 90% precision over approximately 90% recall for the 100 m priors, which is the same performance as Naseer *et al.*. Our approach, however, leads to 84% less operations of image comparisons (creating nodes) than [99], since only 16% of nodes are computed, reducing for this dataset from $244,037$ to $38,334$ image comparisons considering 100 m GPS pose prior. This reduction leads our approach to find the matching images 5 times faster than [99], reducing from 277 to 52. This can also be seen in the Table 3.1 in the third column that corresponds to the dataset 2.

The second experiment shows a bigger and more challenging dataset that contains a loop within a query sequences, i.e., the robot revisited twice places stored in the database (approx. the first 200 images). This can be seen in Figure 3.10 second row, left image. From this plot, we can see that a query loop has occurred because for the same values in database sequence there correspond two values in the query sequence according to the ground truth labels. Additionally, the query sequence here visits the places not covered by a reference sequences at all. This can be seen by the absence of the ground truth labels for the query images from around $750-850$ and $950-1150$. The same image also shows that our approach—although solving the data association problem using topological sorting—can handle the loops better than the network flow solution in [99], since our image associations (blue) follow ground truth matchings (gray) better than matchings from [99] (red). More quantitative results for this dataset is provided by the right figure in the same row of Figure 3.10. By using $50\,m$ GPS prior, our approach reaches 80% precision over 80% recall. For this dataset our approach outperforms [99] given $50\,m, 100\,m$, and $500\,m$ pose priors. As can also be seen from the plots OpenSeqSLAMand a best match strategy based on the HOG descriptors ignoring the sequence information (called “HOG-bm”) performs worse. It should be noted, however, that SeqSLAM was not designed to handle loops and blackouts within the image sequences. By using pose priors, we are able to not just handle loops in the trajectories, but also substantially reduce the number of image-to-image comparisons also for this dataset. By using the GPS prior of $50\,m$, we can achieve the reduction of 94%, which leads processing 11 times faster, reducing computation time from $798\,s$ for [99] to 70. Further timings can be found in Table 3.1 in the column corresponding to the dataset 3.

Finally, we used a third dataset, a more challenging one, since it contains multiple loops in database and in query sequences. The database loops can be seen in Figure 3.10 last row left, by noticing the fact the ground truth labels for some query images correspond to several locations in the reference sequence. For example for the query image 500 there are 3 corresponding (gray) areas within the reference dataset. The sparse cost matrix corresponding to this dataset can also be seen in Figure 3.8. Here, the car was driving in circles around perceptually similar blocks. Similarly to before, the usage of a GPS prior enables us to better match the corresponding parts of the query to database trajectory. Looking at precision-recall curves for this dataset, we observe that our method with $d_{GPS} = 50\,\mathrm{m}$ gives the best results. This is due to the fact that distance between the parallel streets in the dataset is smaller then $100\,\mathrm{m}$ and using $d_{GPS} = 50\,\mathrm{m}$ leads to clearly disconnected components in cost matrix. The reduction with respect to the image-to-image comparisons is 94%, since only 6% of nodes are computed, which is similar to previous dataset. However, on this dataset we are able to

Table 3.1: Performance comparison between the approach of Naseer *et al.* [99] and our approach with different GPS priors $500, 100$ and $50\,m$. Every cell stores the total number of image-to-image comparisons that was computed, total time for finding image matches and time to search a path given a constructed graph. **Nodes used** specifies the percentage of the nodes that were computed in comparison to total possible nodes, as used in [99]; **Times faster** specifies how many times faster is our approach in comparison to [99].

	Dataset				
	1	2	3	4	5
Q	79	676	1,213	1,266	1,428
D	943	361	596	3,601	1,476
Naseer *et al.*	74,498	244,037	722,948	4,558,866	2,107,728
[99]	100s / 0.7s	277s / 2s	798s / 6s	4,843s / 55s	2,305s / 19s
GPS, 500 m	74,498	134,791	298,432	2,620,748	1,102,689
	100s / 0.7s	155s / 1s	325s / 2s	2,734s / 23s	1,283s / 10s
Nodes used	**100%**	**55%**	**41%**	**58%**	**52%**
Times faster	1.0	1.7	2.4	1.7	1.8
GPS, 100 m	50,643	38,334	72,788	621,369	106,672
	47s / 0.2s	52s / 0.23s	107s / 0.5s	660s / 4s	138s / 0.6s
Nodes used	**68%**	**16%**	**10%**	**14%**	**5%**
Times faster	2.0	5.3	7.4	7.3	16.7
GPS, 50 m	38,313	26,841	45,457	288,312	76,171
	33s / 0.1s	42s / 0.1s	70s / 0.2s	316s / 1s	100s / 0.25s
Nodes used	**51%**	**11%**	**6%**	**6%**	**4%**
Times faster	3.0	6.6	11.4	15.3	23.0

compute the image matching 15 times faster, by reducing the computational time from $4,843\,s$ which is around 1 hour and 20 minutes to $316\,s$ corresponding to only 5 minutes. Further timings are presented in Table 3.1 dataset 4.

The next experiment is designed to illustrate that exploiting the GPS prior can lead to a substantial reduction in the computational requirements. Table 3.1 summarizes the timing results of evaluating the algorithms on datasets of different size and complexity. The table depicts timings and number of image comparisons for five datasets of different sizes, where Q denotes the number of images in the query sequence and D number of images in the reference/database sequence. An image sequence is considered more complex if it has one or more loops inside. Every entry of the table stores three values. On top of the cell, the number of image-to-image comparisons that needed to be performed by a given algorithm. The second row of each cell stores the total time in seconds needed to find the matching image pairs between the image sequences and the time for the search to find the image matchings given the fact that the graph was already constructed. As can be seen from the table, the datasets of smaller size tend to show a smaller gain in terms of the overall reduction of image comparisons and thus the number of nodes in the data association graph compared to larger datasets. For example, a smaller dataset 1 uses 51% of the nodes for the 50 m prior whereas larger dataset 5 uses only 4% of the nodes. This is due to the fact that smaller datasets typically cover more near-by places and thus yield a denser matching matrix. In general, we can say that the larger the area that the dataset covers, the bigger the gain of our method. For example, dataset 5 has more images than dataset 1 and thus has a node reduction of 96% whereas the dataset 1 only has $100\% - 51\% = 49\%$ reduction. However, not just the size of image sequences influences the performance of the approach, but also the nature of the trajectories, i.e., the number of loops the sequences have. For example, trajectory 4 has twice as many potential node correspondences as dataset 5, but the number of created nodes is 3 times higher. This happens because the dataset 4 contains multiple loops and thus more nodes need to be created to cover possible matching hypothesis.

Most of the computation time is spent on the comparison of the image descriptors. The computation of the global HOG descriptor per image, described in [99], takes around 23 ms and matching two descriptor takes around 6 ms—but this has to be done often. The GPS priors help to reduce the overall number of comparisons and thus lead to a substantial reduction of the computation times.

3.5 Conclusion

We proposed an approach to visual image matching under substantial appearance changes by exploiting sequence information. We extended our recent approach [99] so that it can exploit noisy GPS pose priors and at the same time substantially reduces the number of required image comparisons. This enables us to run our method online. In addition to that, our approach can naturally handle loops in the input image sequences. We implemented and tested our approach using real world image and GPS data acquired in summer and in winter. Our comparisons suggest that our approach can increase the matching performance while reducing the computation time and in this way outperforms the existing methods under consideration.

Chapter 4

Lazy data associations for online image sequence matching

In the previous chapter, we have described our graph-based image sequence matching strategy that is able to operate in outdoor environments that change their visual appearance dramatically. This search procedure operates on the given, i.e. precomputed, cost matrix. To match two image sequences, we first match every image in query sequence with every image in the reference sequence. Based on the cost matrix, in which every element encodes the information how similar two images are, we proposed a graph structure that is able to naturally incorporate the sequentiality of both data streams and to find the image correspondences between the two sequences. One of the disadvantages of this approach, however, is the necessity to build up the full cost matrix. As we have shown in Chapter 3, we can partially overcome this problem in case a rough pose prior is available. Even though, we have shown in our experimental evaluations that we are able to dramatically reduce the number of required image-to-image comparisons, the proposed approach still needs the complete query and reference sequence to construct a graph, i.e., before being able to start the search. This fact, makes the previously proposed approach an offline approach, i.e., it can only operate when the complete sequences are available and there is no possibility to add further query images whenever the graph construction reached the final node. In context of place recognition for the robotic systems, this is a serious limitation, since it is essential for the robot to get the localization information in a timely manner, while it is operating in the environment.

In this chapter, we propose an algorithm for visual place recognition that is able to operate in an **online** fashion. It makes an image-to-image association decision for every incoming query image and thus making it applicable for the real world robot localization applications. To target the problem of computing the image comparisons to all reference images, we keep the number of image-to-image comparisons small by constructing the graph only around the current most promising matching path hypothesis, i.e., around the brighter patterns similar to the Figure 3.7 (right), but without any prior pose information. To be able to keep track of multiple shortest path hypotheses, we took inspiration from the ideas of the lazy data associations proposed by Hähnel *et al.* [61] in context of LiDAR scan matching. In this way, we can revise previously made data association decisions in case new information is available that allows us to make better, more informed data association decisions. In the following section, we describe the graph structure for the online matching as well as an efficient heuristic that allows us to keep the number of image-to-image comparisons small. In Section 4.2, we show the experimental evaluations that confirm the following claims:

(i) our approach has the ability to run in an incremental fashion so that only few nodes are expanded so that online localization is possible,

(ii) our heuristic is well-suited to find a competitive solution in most real world situations,

(iii) our algorithm is able to exploit additional location prior information and can in this case also handle loops in robot's trajectories.

4.1 Lazy matching for online operation

To turn our offline system into an online one, we need to ensure that we can make image association decisions for every incoming query image. Namely, for every incoming image, we want to know if there is a corresponding match in the database and if so, to which image of the database it corresponds to. The database itself is organized as a list fo regular files. The graph structure proposed for offline matching builds on top of the full cost matrix (unless pose prior information is available). This leads to instantiating a large number of nodes in the graph. As a reminder, the image-to-image comparisons are costly operations and a large number of them needs to be executed. To be able to do online matching, we need to limit the number of those comparisons. We do so by analyzing at every step if the currently considered node is ”worth expanding”, i.e., if adding the children of this node to the graph will lead to finding the shortest path. The decision of whether to expand the node or not is made based on an efficient heuristic. This allows us to eliminate the instantiation of the majority of nodes that are ”far away” from the shortest path hypothesis. Additionally, instead of constructing the full graph and perform a search in it, as it was done for the offline approach, we now build and search on-the-fly. We construct only a portion of the graph relevant for the current best path hypothesis and update our search

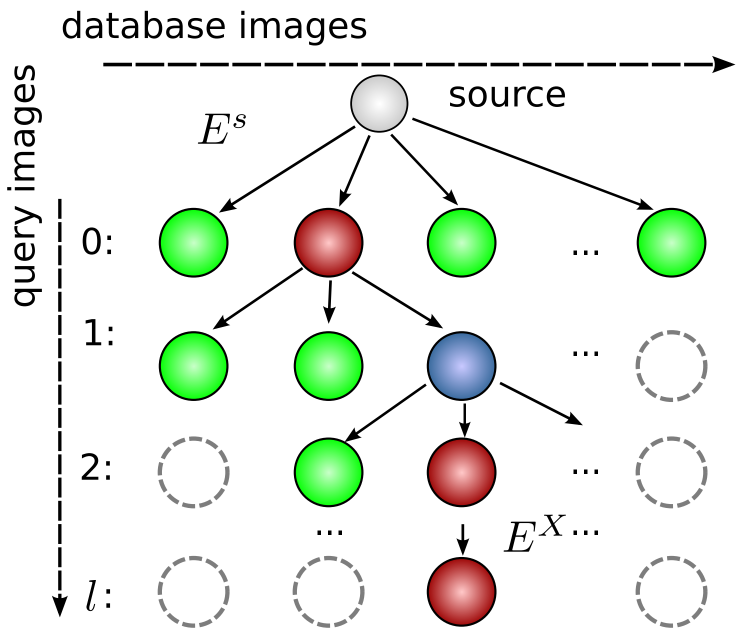


Figure 4.1: Schematic illustration of the graph structure for the search. To perform an online localization our algorithm compares only image pairs that correspond to the green nodes and expands the green area on the fly. Red nodes correspond to matches of similar images along the path through the data association graph and blue indicates matches along the path with a low similarity.

after this construction. In this section, we provide more details about the graph construction and search as well as about our efficient heuristic. Moreover, we discuss how the ideas of lazy data associations are naturally incorporated in our graph-based search procedure.

4.1.1 Data association graph

We start by describing the structure of the data association graph needed for on the fly computations. Similar to the offline approach, we use a directed acyclic graph $G = (X, E)$ as our main data structure for modeling the data association problem. We solve the sequential image matching task by finding a shortest path in this data association graph G. To build up the data association graph on the fly, we only need to compare images if our search algorithm expands the corresponding node in G. The key idea of this data association graph is the following. A node in the graph represents a potential match between two images. We aim at finding the best combination of matching images by searching a path through this graph, see Figure 4.1 for an illustration, where the cost of visiting a node depends on the similarity of both images. In more detail, we propose the following graph structure, which differs from the one used in Chapter 3.

Nodes. We have two types of nodes in X: the root or start node x^s and matching nodes. A matching node x_{ij} models a match of the image $i \in \mathcal{Q}$ with the image $j \in \mathcal{D}$. The more similar two images are, the more likely is the fact that they represent the same place. The similarity of an image pair is defined as before as $z_{ij} \in [0, 1]$, where $z_{ij} = 1$ means that both images appear identical. The similarity z_{ij} is computed by comparing the images $i \in \mathcal{Q}$ and $j \in \mathcal{D}$ only through their global image descriptor using the cosine distance.

As we are building the graph online, new nodes x_{ij} need to be created as soon as a new image i is recorded. Adding a node x_{ij} to the graph, however, comes at a *computational cost* as we need to compare images to compute z_{ij}. Thus, for building up the graph, we seek to avoid instantiating unnecessary nodes x_{ij}, i.e., nodes, which are not part of the matching sequence. The ones that are unlikely to be belong to the shortest path.

Edges. Similar to the nodes, we use two types of edges $E = \{E^s, E^X\}$ according to the types of nodes the edges connect. The set of edges E^s connects the source node x^s with the matching nodes x_{0j} corresponding to matching the first query image with any database image $j \in \mathcal{D}$, i.e.,

$$E^s = \{(x^s, x_{0j})\}_{j \in \mathcal{D}}. \tag{4.1}$$

The second set of edges E^X connects the matching nodes and are the same as in Chapter 3, see Equation (3.3), i.e.,

$$E^X = \{(x_{ij}, x_{(i+1)k})\}_{k=j-K,\dots,j+K}, \tag{4.2}$$

where K is a fanout parameter that influences the nodes that are connected between the query images i and $i+1$. The fanout basically models that the robot can move at different speeds through the environment or that the cameras can operate at different frame rates. The larger the K, the larger is the branching factor of the graph and, thus, the value of K impacts the speed of the search as described in the following sections. In our current implementation, we use a fanout parameter of $K = 5$. In the remainder of this chapter, the nodes $x_{(i+1)k}$ are referred to as the children $\mathrm{ch}(x_{ij})$ of the node x_{ij}.

Weights/Costs. Each edge $e \in E$ has a weight or cost associated to it. This weight $w(e)$ is related to the similarity score z_{ij}. The weight of an edge $e = (x_{ij}, x_{i'j'}) \in E^X$ is inverse proportional to the similarity of the node to which this edges leads to, i.e.,

$$w(e) = \frac{1}{z_{i'j'}}, \tag{4.3}$$

where $z_{i'j'}$ is a similarity score computed when comparing image i' and j' using the cosine distance.

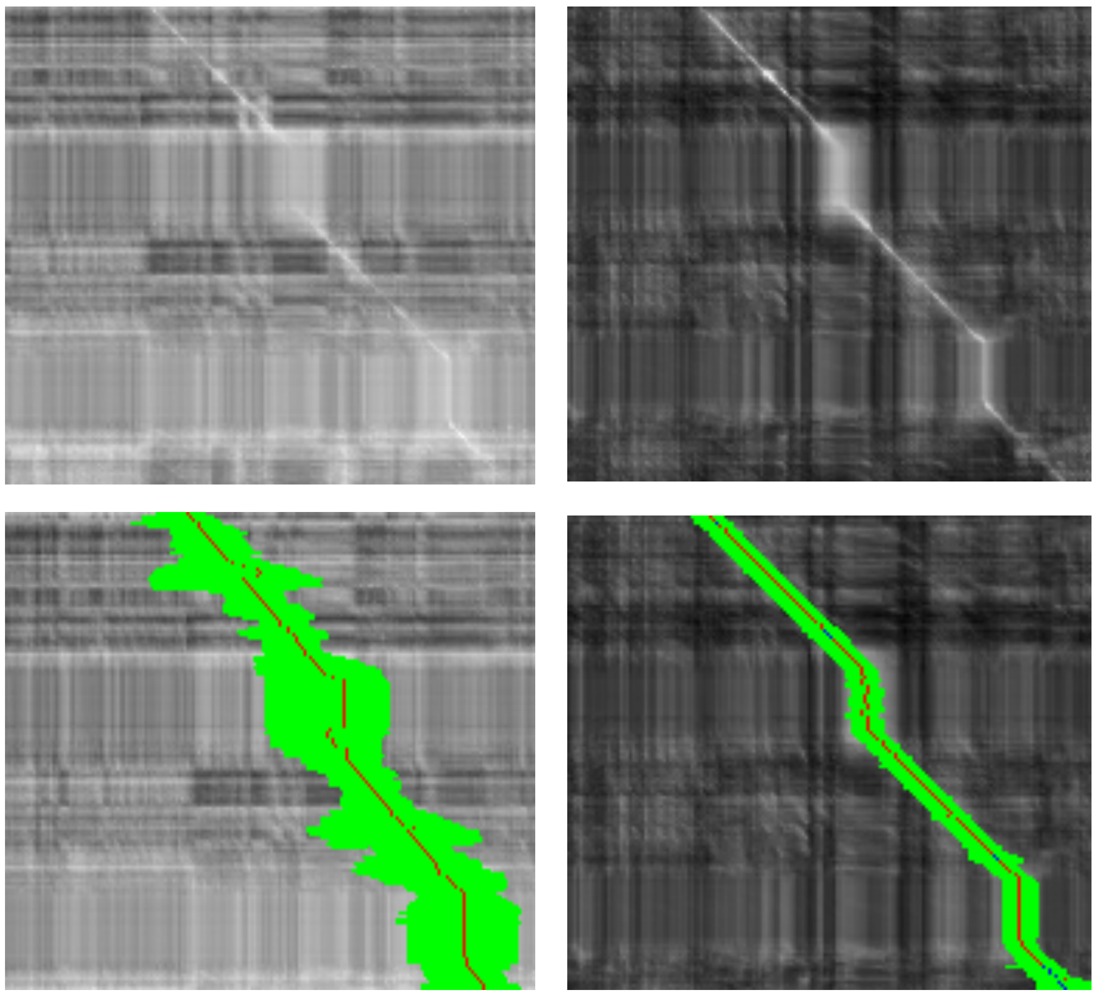


Figure 4.2: Similarity matrix computed using tessellated HOGs as in [99] (left) and OverFeat features (right). As can be seen in the first row, the OverFeat features yield more distinct similarity values. This leads to a smaller number of nodes that are instantiated in the data association graph (green), as depicted in the second row.

4.1.2 Computing image similarity based on features from deep convolutional neural networks

The similarity computation between the two images has to be done often and thus we are in general interested in a fast computation. At the same time, the quality of the similarity function is of high importance. The more distinct the value of z_{ij} are for images taken from the same places vs. those from other places, the easier is the data association problem. As a result, the more distinct such values are, the better the performance of our graph search algorithm as less nodes will need to be expanded.

In our initial approach, we computed the tessellated HOG descriptor for every image and then compared them using the cosine distance. The obtained cost difference between the best match and the worse match was sufficient to find good solution with an exhaustive search. In the context of our lazy data association approach with a search heuristic, we experience problems to find matching sequences reliably without expanding the majority of the nodes in the graph. Therefore, we changed the image descriptors in this work to the features from the pre-trained deep convolutional neural network OverFeat as proposed by Sermanet *et al.* [119] due to its better matching performance. OverFeat is built using a network trained on the ImageNet dataset consisting of 1.2 million images and was published in 2016. We used the 10th layer as a global image feature as suggested by Chen *et al.* [32]. Using OverFeat features instead of HOG directly improves the performance of our algorithm and supports the lazy approach. To give an intuition about the matching similarity, Figure 4.2 depicts the similarity of comparing all possible combinations of images from database $\mathcal{D}$ and query $\mathcal{Q}$ computed with the tessellated HOG descriptor (left) and OverFeat (right). Brighter values indicate a higher similarity. As can be seen from the images, the OverFeat features lead to more distinct values (higher contrast) and thus less nodes need to be expanded during the search (green area).

4.1.3 Image sequence matching through graph search

The sequence of matching images between $\mathcal{Q}$ and $\mathcal{D}$ can be computed by a path search from the start node x^s to any node x_{l*}, with $*$ referring to any index in $\mathcal{D}$ and l being the most recent image in $\mathcal{Q}$. Every node that is a part of the shortest path corresponds to a selected data association, i.e., a match. In our implementation, we only keep an index of images and feature descriptor in the memory and load individual images on demand from disk.

The computationally most demanding process for building and searching in such a data association graph is instantiating all nodes as a large number of possible matches has to be computed. For online localization, we are interested

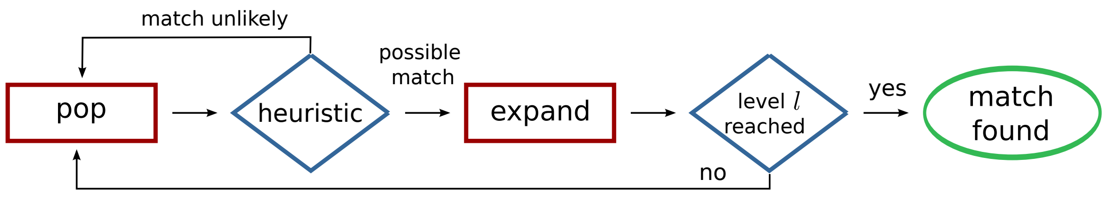


Figure 4.3: Illustration of searching for a match for an input image.

in keeping the computational efforts small and in avoiding the creation of nodes that do not represent potential matches. To address this issue, we propose an approach that limits the number of image comparisons and results in an efficient algorithm siutable for online operation.

Our work is motivated by the ideas of lazy data associations in the context of SLAM proposed by Hähnel *et al.* [61] for constructing pose graphs. Hähnel *et al.* build up a data association tree and expand in each round the node with the highest log likelihood of representing a match between laser range scans. This is similar to a greedy search in a data association tree.

In our case, we go a step further and seek to performing an *informed* search through the graph, while the graph is built on the fly. One popular way to perform an informed search is the A* algorithm using a heuristic, which allows for estimating the cost from the currently expanded node to the goal node. Defining such a heuristic for path planning in a Euclidean space is trivial as the physical Euclidean distance can always be estimated. In the space of image features, however, defining such a heuristic is difficult. For our matching problem, defining a heuristic means we need to predict how well the images that *we will receive in the future* will match our database images —this is in general a difficult task. Furthermore, A* requires that the heuristic is a predefined function and does not change during the search. We, however, try to predict the matching cost based on the images that we have received so far. This means, our heuristic is updated *during the search*, which prevents the application of A*. Therefore, we take a different approach to the search problem. Our search procedure takes into account the estimated matching cost and works as follows.

Similar to A*, we use an open-list F of nodes that are still under consideration. This open-list is realized through a priority queue. In contrast to A*, the key of our priority queue for a node x_{ij} is the cost $g(x_{ij})$ of reaching x_{ij} from the source x^s. Our search and simultaneous graph construction starts with creating the source node x^s and connecting it to the matching nodes according to Equation (4.1). This step requires to instantiate $|\mathcal{D}|$ nodes if no further information about the first possible match is provided.

For a new incoming image referred to as l, we use the following procedure to update the graph as well as the matching sequence (see Figure 4.3 for a brief illustration): Whenever a new image is obtained, we pop a node from F. We use our heuristic, which will be described in the remainder of this section, to estimate if the popped node x_{ij} is worth expanding or if it is unlikely to be part of the matching sequences. Thus, in contrast to A*, the heuristic is not considered for the key computed for the priority queue. If the node is unlikely to be part of the matching sequences, we continue with the next node in F. Otherwise, we expand the node x_{ij} by computing the matching costs for its children $\mathrm{ch}(x_{ij})$ and by connecting the node x_{ij} with $\mathrm{ch}(x_{ij})$ using the edges define in E^X, see Equation (4.2). If a node in $\mathrm{ch}(x_{ij})$ lies on the l^{th} level of the graph, it represents the so far best match for the most recent image and the search terminates for this input image. Otherwise, we proceed expanding nodes from F. These steps are summarized in Algorithm 1, where the function `updateGraph` adds the nodes from set $\mathrm{ch}(x_{ij})$ to the graph.

In contrast to searching for the new path every time the graph gets updated, we only update the found path. To perform a search update step, we only keep track of the last node of the currently best hypothesis $\hat{x}$ and not the full path. The last node of the path is sufficient to retrieve the full path if required. Whenever we have updated the graph structure, we would like to see if recently added information changes our data association decisions. By obtaining a node from $\mathrm{ch}(x_{ij})$, we get an alternative hypothesis for the shortest path in the graph. To update the search,we first check if the alternative path hypothesis has at least the same length as the best current hypothesis, i.e., if it has reached the level l. If not, the alternative hypothesis is neglected on this step. Then the alternative path hypothesis becomes current best path hypothesis if its accumulated cost is smaller than the accumulated cost of the current best hypothesis.

The above described heuristic must estimate the sum of the matching costs for reaching the l^{th} level (the most recent query image). The key problem here is that defining an *effective* and *admissible* heuristic, properties that are needed for optimal informed search [112], is hard due to the small amount of background information that can be exploited to predict future image similarities.[1] Therefore, we take an alternative approach to come up with a heuristic that provides a good estimate of the cost but is not guaranteed to be admissible in the sense of A^* search. A heuristic is admissible if it never overestimates the cost for reaching a goal. We take a statistical approach and approximate an *expected* lower bound for the *average* cost of the unexpanded and thus unknown nodes. We do so by taking into account the average cost of the *best* path found so far as a prediction of the expected cost of individual matches along a new path. This is a reasonable assumption because opening of alternative path hypothesis is an

[1] That is fundamentally different to planning in the 2D or 3D world where the Euclidean distance can serve as an effective and admissible heuristic.

Algorithm 1 Constructing and searching step for an incoming image

```
1: $q_l$ - incoming image feature with index $l$
2: $F$ - frontier of potential nodes to expand
3: row_reached = false
4: while F not empty and row_reached = false do
5:     $x_{ij}$ = F.top()
6:     F.pop()
7:     if !node_worth_expanding($x_{ij}$) then
8:         continue
9:     // expand the node $x_{ij}$
10:    ch($x_{ij}$) = getSuccessors($x_{ij}$)
11:    updateGraph(ch($x_{ij}$))
12:    updateSearch(ch($x_{ij}$))
13:    if ch($x_{ij}$) is in row $l$ then
14:        row_reached = true
```

expensive procedure, since it involves constructing parts of graph not seen before, and to accept it as the alternative hypothesis has to be at least as good as our current best matching hypothesis in terms of the accumulated cost. Furthermore, we exploit the fact that we know the number of images *obtained so far* and we know that the shortest path will have $l + 1$ nodes (start node plus one matching node for each image). This allows us to formulate the expected cost $f(x_{ij})$ for a node x_{ij} to a node on the l^{th} level, i.e., x_{l*}, as the computed cost from x^s to x_{ij} expressed through $g(x_{ij})$ plus the estimate cost as:

$$f(x_{ij}) = g(x_{ij}) + \underbrace{\alpha(l - i)\mu_{\text{cost}}(\hat{x})}_{\text{heuristic}}, \tag{4.4}$$

where $\alpha \in [0, 1]$ is a factor to trade off the quality of the solution and the number of nodes that needs to be expanded. For $\alpha = 0$, we obtain a greedy search behavior and for $\alpha = 1$, we may not expand enough nodes to find a good solution, because the heuristic becomes very concervative and prevents expanding of many hypothesis. The term $(l - i)$ is the number of images that have to be matched to end the sequence and $\mu_{\text{cost}}(\hat{x})$ is the average cost of the best path found so far, see also Figure 4.4. By increasing α, the heuristic will prefer the nodes closer to the furthest expanded row, since the bigger the α the bigger the contribution of $(l - i)$ rows to the predicted accumulated cost and our search procedure prefers the path with the smallest accumulated cost.

In sum, the data association graph constructed in the proposed way using robust features allows us to design a useful, but not guaranteed to be admissible heuristic for the search for data associations. This in turn means that we are

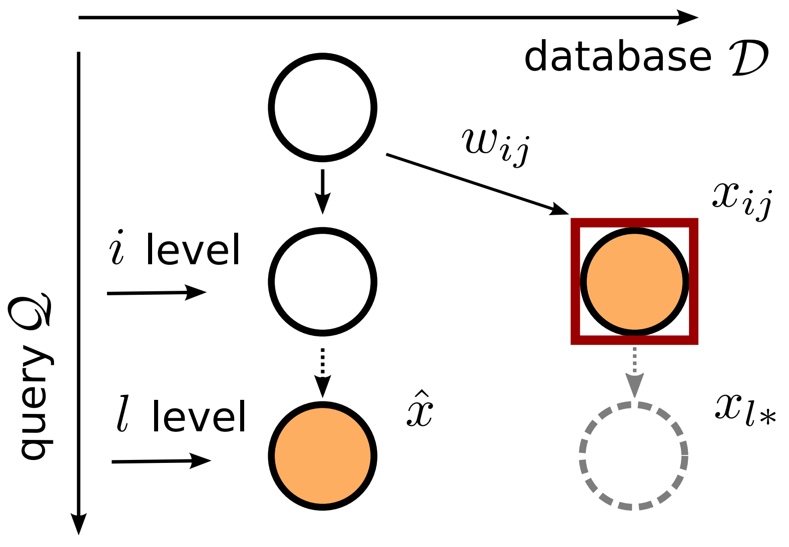


Figure 4.4: Illustration for the graph expanding procedure. Orange nodes are nodes in the F. The red square indicates that the element x_{ij} will be the next one in F. The dashed gray line represent nodes and edges not computed yet.

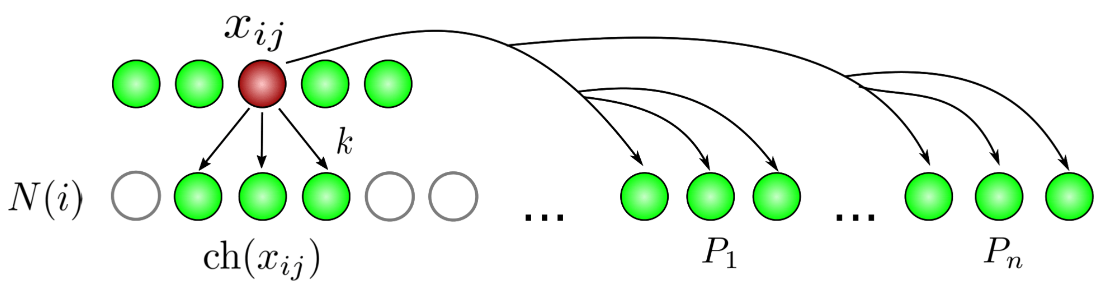


Figure 4.5: Keeping connectivity though additional edges when using location priors. Green: nodes that are expended and added to the graph; gray: potential neighboring nodes according to the prior, but not encountered in the graph search and thus not instantiated.

not guaranteed to find the optimal solution but enables a fast search for image matching across seasons that can be executed online.

4.1.4 Exploiting location priors for online matching

In case a rough location prior, for example from a noisy GPS, is available, we can further improve the matching procedure and can also better deal with loops in the database as well as query sequences, i.e., place revisits of the robot/vehicle.

The graph construction described in Section 4.1.1 can naturally be extended to account for location prior information also for online matching. The overall procedure of constructing the graph stays the same but an additional location prior allows us to identify for every query image i the set of possible neighboring images $N(i)$ as

$$N(i) = \{j \mid j \in \mathcal{D} \wedge \mathrm{dist}(i, j) < d_{max}\}, \tag{4.5}$$

where $\mathrm{dist}(i, j)$ is the distance between the position at which the images with the indices i and j have been taken according to the location prior. All elements not in $N(i)$ will be discarded based on the prior information and thus several matching hypotheses do not need to be computed.

As reported in Section 3.2, incorporating such location prior information may lead to disconnected graph components. For an incremental search, it is however easy to connect different components by extending the definition of the children $\mathrm{ch}(x_{ij})$ of a node by

$$\mathrm{ch}(x_{ij}) \leftarrow \mathrm{ch}(x_{ij}) \cup P_1 \cup \ldots \cup P_n, \tag{4.6}$$

see Figure 4.5 for an intuitive definition of the components $P_1, \ldots, P_n$. The figure depicts the situation, where for a query image i there exist several places in the reference sequence that are within the distance d_{max}. In this situation, the "direct" children $\mathrm{ch}(x_{ij})$ of x_{ij} are getting connected to x_{ij} based on fanout relation using set of edges E^X as described previously. Since for other components $P_1, ..., P_n$ the entry point is unknown x_{ij} gets connected to all nodes in the respective components. Thus, we ensure that if the path stays in the same component, the procedure of building the graph is not changed. If, however, the path "jumps" to another component, we account for this possibility given the prior. In case of a "jump", the current best match hypothesis gets connected to all nodes from other components as illustrated in Figure 4.5.

4.2 Experimental evaluation

Our evaluation is designed to illustrate the performance of our approach and to support the following three claims:

(i) our approach has the ability to run in an incremental fashion so that only few nodes are expanded so that online localization is possible,

(ii) our heuristic is well-suited to find a competitive solution in most real world situations,

(iii) our algorithm is able to exploit additional location prior information and can in this case also handle loops in robot's trajectories.

Throughout our evaluation, we rely on multiple publicly available datasets, see Figure 4.6. We use the summer-winter dataset as in previous chapters referred to as Freiburg and the Nordland dataset, which is a four season train ride through

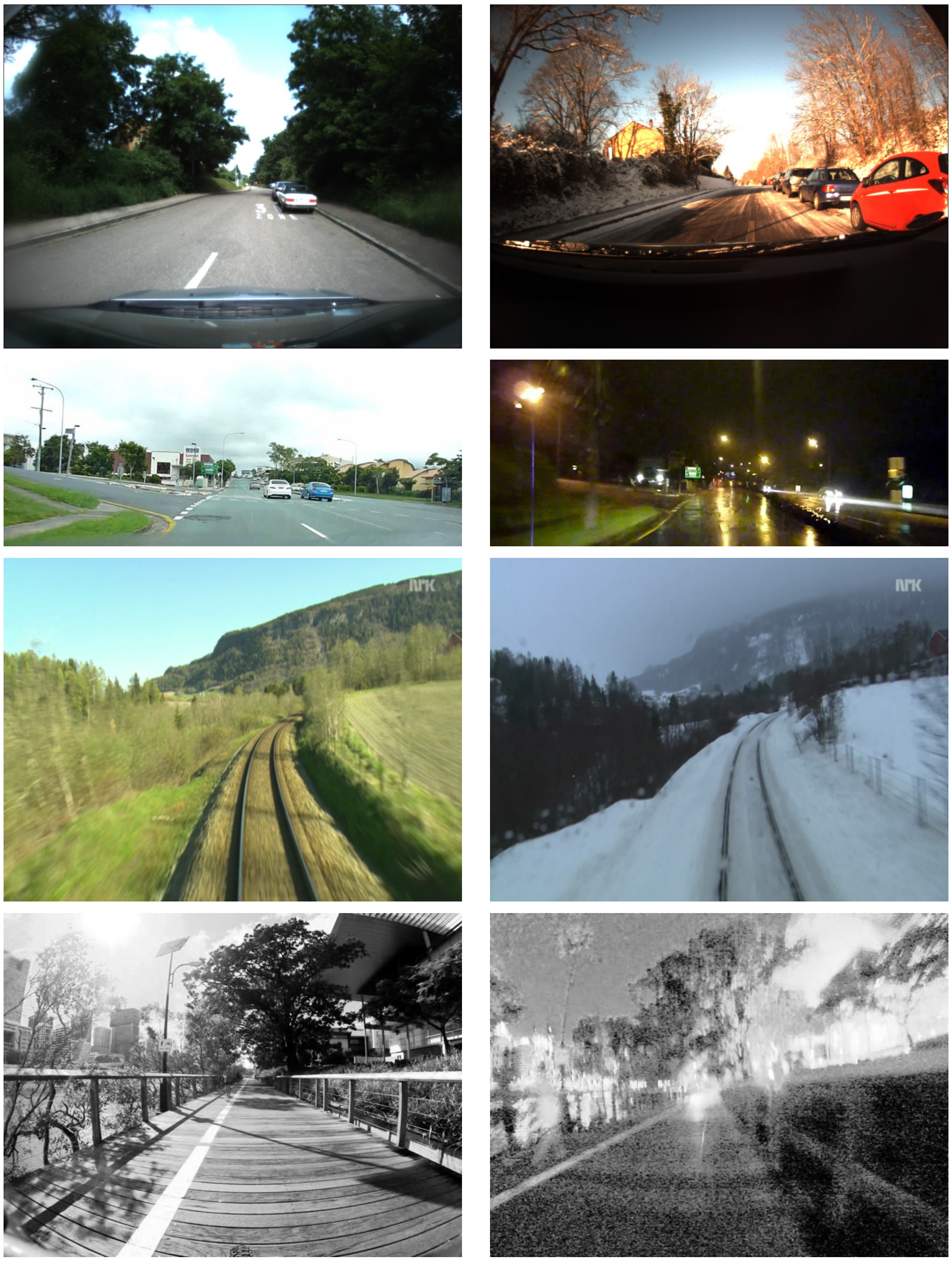


Figure 4.6: Examples of typical image pairs taken at the same places within multiple datasets. The image pairs are successively found by our approach. First row: Freiburg dataset; second row: Alderley dataset; third row: Nordland dataset. Fourth row: day/night scene from the VPRiCE'15 Challenge dataset.

Norway. We also use the Alderley dataset [92] recorded during a sunny day and a rainy night. Finally, we used the datasets that have been selected for the VPRiCE Challenge 2015[2]. The challenge consists of $4,022$ query and $3,756$ database images organized as a single sequence although it resembles multiple different datasets stitched together.

Besides setting the performance of our online method in relation to full offline matching, we compare it to OpenSeqSLAM [92] as well as to a baseline approach that uses approximate Euclidean nearest neighbor search to find the most similar image according to the OverFeat features using the popular fast approximate nearest neighbour approach called FLANN [94]. As we will see in the remainder of this work FLANN approach is not well-suited to solve the across season matching problem.

4.2.1 Matching performance

The first experiment is designed to illustrate the capabilities of our approach. Figure 4.7 depicts the full matching matrix from a subset of the VPRiCE dataset with strong seasonal changes. Our algorithm compares $29,317$ image pairs out of $5,693,135$ possible individual image comparisons that approaches such as [99] need to compute. This yields a reduction of the computation time of 99.5%, while obtaining a comparable matching performance. We obtain reductions by more than 95% for most datasets. In general, the larger the dataset the bigger the savings. Also the distinctiveness of the similarity score plays a role for our algorithm. As it can be seen in Figure 4.8, the block of the similarity matrix in the upper left corner shows no distinct matching pattern. This means that several images appear similar when compared through the feature descriptor. As a result, our approach expands a comparably large number of nodes, indicated by the green elements in the right image. Note that our algorithm does not compute the full similarity matrix as it is shown here, we depict it for visualization purposes only.

The second set of experiments is designed to show how the proposed heuristic influences the matching performance based on the Freiburg, Nordland, and Alderley datasets. We compare the matches of our online method with those of our previous offline approach Chapter 3 using the full matching matrix but replacing the previously used HOG features by OverFeat. The results in Figure 4.9 and Figure 4.10 illustrate that our heuristic leads to matching results comparable to offline search because the precision-recall curves mostly overlap, while the number of image comparisons that need to be performed drops dra-

[2]https://roboticvision.atlassian.net/wiki/spaces/PUB/pages/14188617/The+VPRiCE+Challenge+2015+Visual+Place+Recognition+in+Changing+Environments

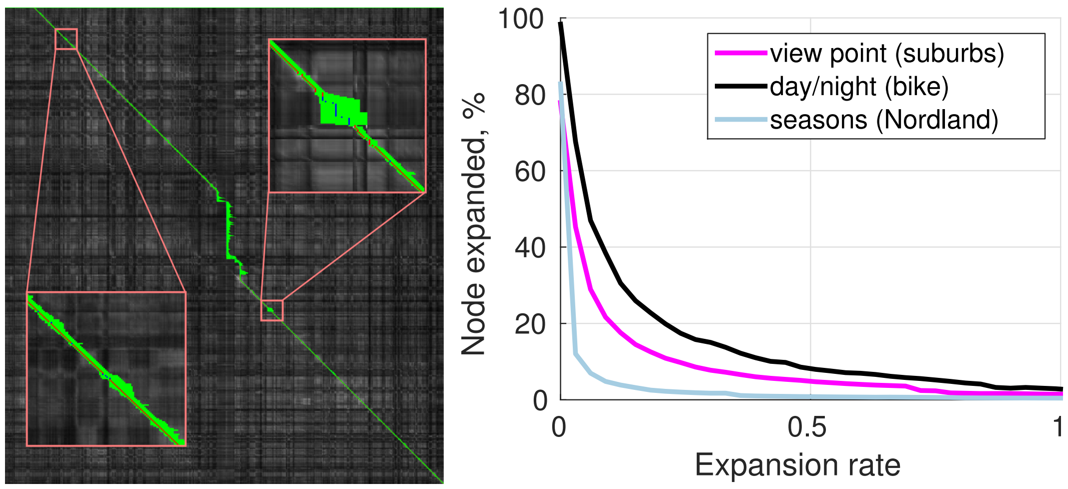


Figure 4.7: Left: visualization of the graph structure for the dataset with dramatic seasonal changes (Nordland sequence from VPRiCE). The algorithm compares the images only for the nodes marked with green. Other nodes are computed for visualization only. Right: Plot of the dependency between the expansion rate α and the number of matching cost computations, expressed in percentage from total number of nodes.

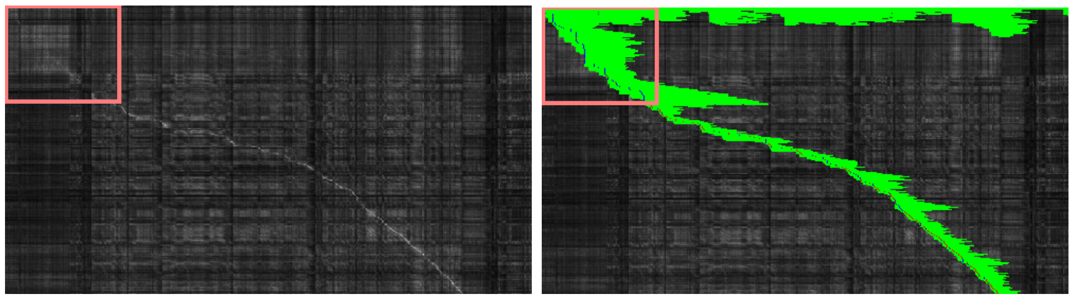


Figure 4.8: Full matching matrix (left) and the nodes expanded by our algorithm (green nodes in the right image). The similarity matrix is computed for visualization only. The squares highlights an area in which most images are hard to distinguish, which leads to a larger node expansion.

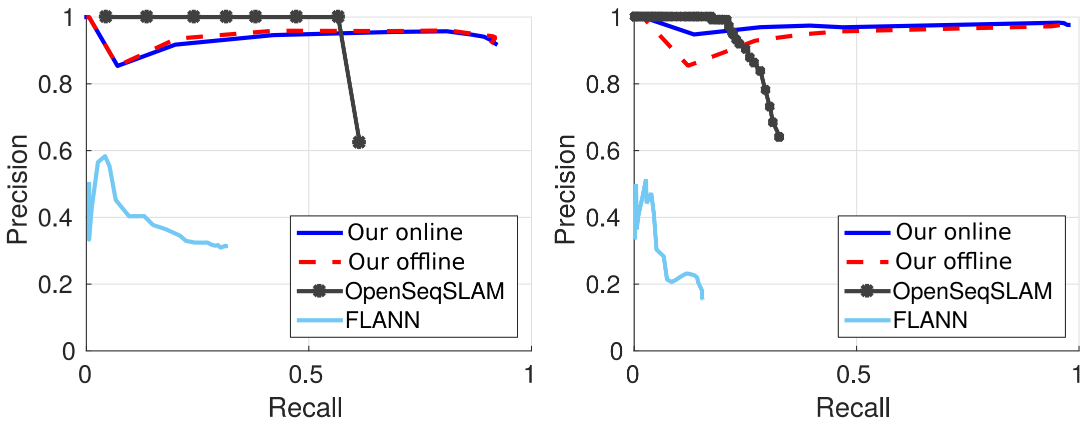


Figure 4.9: Precision-recall plots for the datasets Nordland (left) and Freiburg (right).

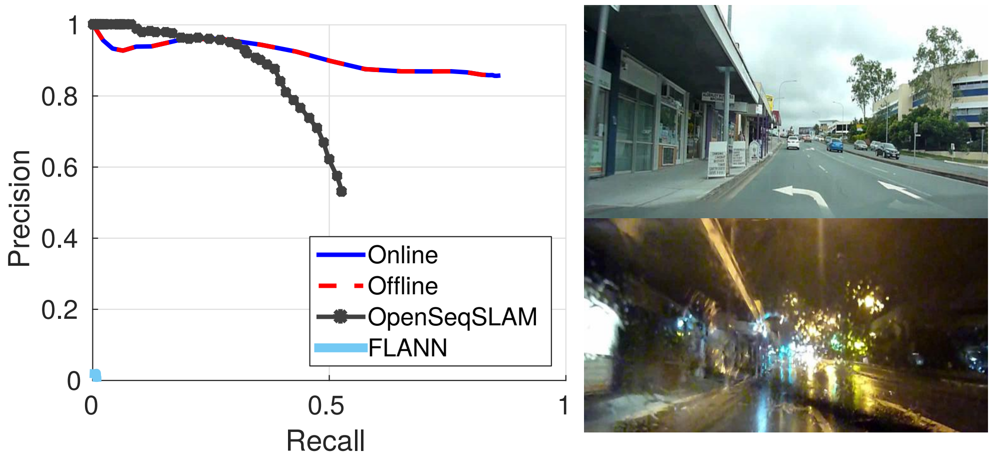


Figure 4.10: Performance evaluation on the Alderley dataset.

matically from 100% for $\alpha = 0$ to less than 10% for $\alpha = 0.6$, see Figure 4.11. As all precision-recall plots illustrate, we outperform OpenSeqSLAM as well as the FLANN baseline. Our approach reaches around 90% precision for 90% recall for Nordland and Freiburg datasets as well as around 90% precision for 80% recall for more challenging Alderley dataset. As expected FLANN shows poor performance on the proposed datasets due to poor feature discriminability under strong visual appearance changes.

We also participated with our approach in the place recognition challenge VPRiCE 2015 conducted at ICRA 2015 and CVPR 2015 workshops. The evaluation has been performed by the challenge organizers. Our online algorithm achieved 3$^{\text{rd}}$ place with precision 0.680 and recall 0.755 in the test settings. The approach that scored first [93] is an offline method, i.e., a method that must know the whole dataset beforehand, and the second place [56] focuses on the design of new features for image comparisons and thus could even be combined with our method as they are rather orthogonal.

4.2.2 Node expansion

The third experiment is designed to evaluate the expansion of nodes in the data association graph in more detail. The evaluation illustrates that we can achieve online performance as only a comparably small number of nodes in the data association graph get expanded in every step. The two major factors that influence how the graph expands are the distinctiveness of matching costs and the expansion parameter α. We varied the expansion parameter of our heuristic in Equation (4.4) between 0 and 1. Zero leads to a greedy search, while $\alpha = 1$ approximates the expected cost by the average cost of the best path. Figure 4.11 (middle) shows the dependency between the graph size and the applied

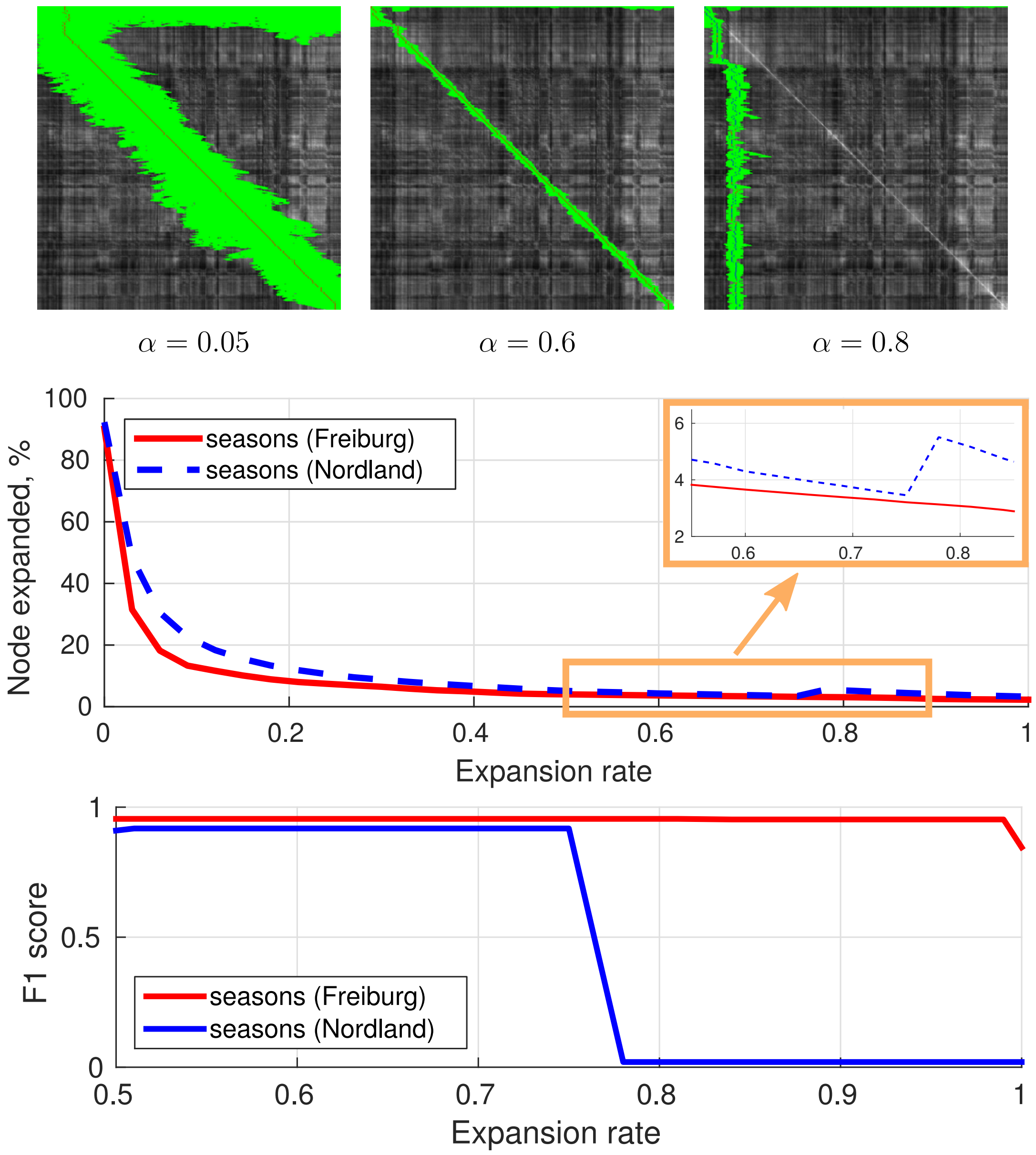


Figure 4.11: In overall selecting the bigger expansion parameter α leads to a decrease in node expansion, while preserving the accuracy of the solution. The middle plot also shows that constraining α close to 1 may prevent finding the correct path. It leads to degradation in accuracy (bottom) and may lead to increase in node expansion, depending on the underlying data (middle).

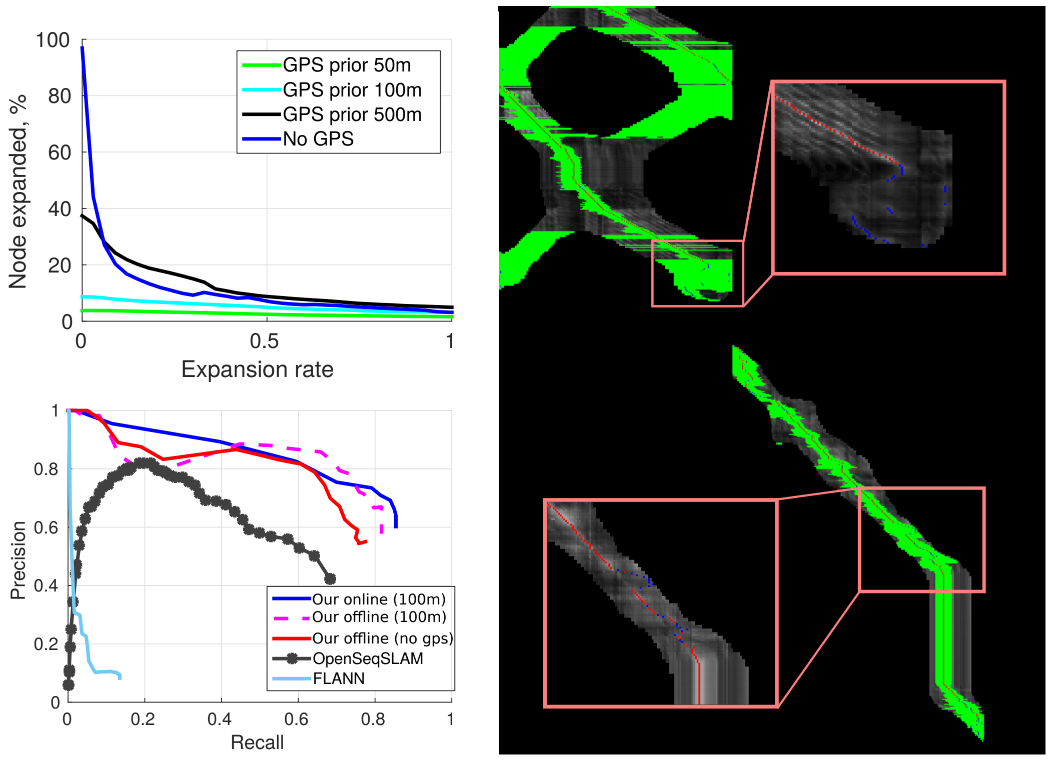


Figure 4.12: Exploiting location priors enables handling the loops in image sequences. Right: Example of the similarity matrix constrained with 100m GPS prior and overlaid graph search results. Top left: comparison using precision-recall plots. Bottom left: node reductions relative to the uncertainty of the location prior.

expansion rate for the Freiburg and Nordland dataset. Roughly speaking, the closer the expansion parameter α is to 1, the smaller the resulting graph will be and vice versa. Constraining α to the values close to 1, is likely to prevent the algorithm from finding the optimal path, see Figure 4.11 (top, right) for an example. In this figure, the matching nodes, which are computed using our approach, are colored in green. All other are depicted for illustration purposes only and do not need to be computed in practice. In these cases, the *reductions* in the number of expanded nodes are $\{65\%, 95.7\%, 95\%\}$ (from left to right). The figure also shows the F1 score illustrating that too large values for α can lead to a decay in matching performance. As a result of this, in all our experiments, we select $\alpha = 0.6$ as a good trade off between matching performance and computational savings.

4.2.3 Exploitation of additional location priors

The next experiment is designed to show that in presence of additional but potentially noisy location priors, our algorithm is able to handle loops in the query and database trajectories as well as deviations from the database, i.e., visiting unknown areas. Figure 4.12 (right) depicts a similarity matrix between a query and database sequence for the scenario in which the location of the robot is known up to 100 m, for example obtained from a consumer GPS receiver operating under suboptimal conditions. The green area corresponds to the nodes that are instantiated in the graph construction and search, while black areas correspond to the nodes excluded due to the location prior. Also in this settings, our algorithm is able to avoid instantiating unnecessary nodes while correctly finding the path. Figure 4.12 (top) shows that the expansion can be reduced to 20% in comparison to the full offline method. Additionally, the figure shows that the the gain in node reduction is smaller the better the pose is known from the prior, which is an expected result. Exploiting location priors furthermore allows us to handle loops better, see for example Figure 4.12 (bottom). As can be seen, there is almost no decrease in performance of our approach in comparison to the offline method also using OverFeat and GPS prior (Offline 100 m). Moreover, this experiment shows that we can deal with loop better than offline approach without pose prior (Offline, no gps). We furthermore outperform SeqSLAM by reaching 60% precision over 85% recall, while SeqSLAM for this dataset is reaching 70% recall with 40% precision. It should be noted though that original SeqSLAM was not designed to deal with loop in the sequences. Due to the low discriminative properties of the features FLANN method shows a poor performance in these situations.

4.2.4 OverFeat vs. HOG features

We also analyzed the performance of the matching approaches using HOG features, as used in previous chapters and the pre-trained OverFeat features by Sermanet *et al.* [119]. We found that the OverFeat features outperform the HOG features for place recognition under strong appearance changes as they provide more distinctive matching scores, see also Figure 4.2. For the HOG features, the ratio between the best match and the worse match using the cosine distance is 1.46 compared to 4.28 for OverFeat. Thus, using HOGs is less effective for the search method presented in this chapter as a larger number of nodes of the data association graph would be expanded (see green area in the last row of Figure 4.2). In this sense, we confirm the results by Chen *et al.* [32] that the 10th layer is well-suited for place recognition tasks.

4.2.5 Timing

Our approach can run online with around 1 fps on a standard notebook computer. Breaking down the timings of the individual components shows that computing the OverFeat descriptor takes the largest amount of time with approx. 500 ms. Expanding a single node, i.e., comparing two descriptors, takes 8 ms. The incremental update of the shortest path takes around 40 ms on average.

4.3 Conclusion

In this chapter, we proposed an incremental approach to visual image sequence matching under substantial appearance changes for online operation. The key idea is to apply a lazy data association approach and to define a heuristic for the search in the data association graph that estimates the similarity of images. This enables us to realize an online approach for image sequence matching under substantial appearance changes. We furthermore illustrated that noisy location priors can be exploited during online search. We implemented and tested our approach using real world image sequences acquired in summer and in winter as well as under different weather conditions. Our comparisons to other methods as well as the results from the VPRiCE 2015 place recognition challenge suggest that our approach provides competitive results and avoids expanding large portions of the data association graph or building a large matching matrix.

Chapter 5

Hashing-based relocalization for place recognition with flexible trajectories

In previous chapter, we proposed a graph-based visual place recognition approach which is able to establish image correspondences in online fashion. For every incoming query image, the proposed approach finds the matching image in the reference sequence while preserving the sequentiality of the input data. One of the limitations of the previous approach that prohibits its real world application is an assumption that image sequences should be "weakly" synchronized. By weakly synchronized we mean that the image sequences should cover roughly the same place in the environment, but do not need to be image-to-image synchronized. The approach proposed in Chapter 4 can handle the loops in the trajectories only in the presence of uncertain pose prior. In this chapter, we relax the assumption of weakly synchronized image sequences and allow for ”flexible” robot trajectories, the ones that contain loops in query and reference sequence, as well as the parts of the query trajectory that deviates from the reference one recorded beforehand. We therefore present an algorithm that matches image sequences collected by cameras taken different, partially overlapping trajectories in the environment.

Localization under appearance changes is essential for robots during long-term operation. Our approach builds upon the work presented in Chapter 4 on graph-based image sequence matching and extends it by incorporating a hashing-based image retrieval strategy in case of localization failures or the kidnapped robot problem. In this chapter, we present a variant of hashing algorithm that allows for fast retrieval of high-dimensional CNN features. Our experiments suggest that our algorithm can reliably recover from localization errors by globally relocalizing the robot. At the same time, our hashing-based candidate selection is substantially faster than state-of-the-art locality sensitive hashing.

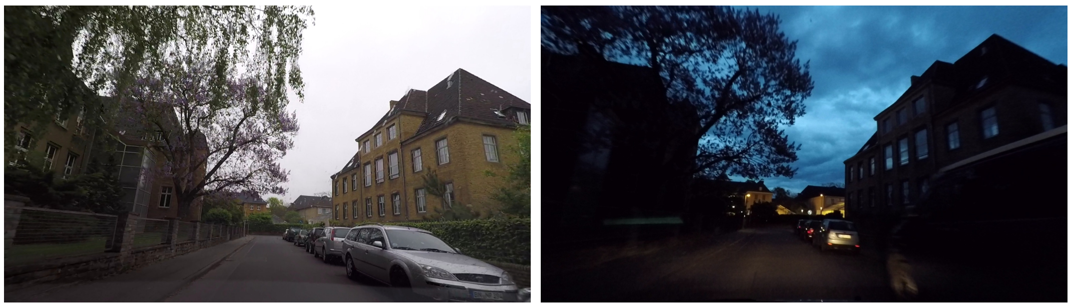


Figure 5.1: Challenging image pairs for place recognition systems. Both images have been recorded at the same place but during different times resulting in strong appearance changes. The approach presented in this paper identifies such corresponding images via sequence information and can handle loops in the database sequences, recover from localization failures, as well as deal with the kidnapped robot problem.

This chapter extends our previous approach along several dimensions. First, we provide a way for dealing with loops in the reference or database sequences and introduce new edges into the data association graph that is build up on the fly. Second, we provide an efficient way for relocalizing the robot in case it got lost. Both extensions naturally integrate with and extend Chapter 4 so that the same search approach and heuristic is used to build upon. It furthermore does not affect the online nature of the solution and the data association graph is still built incrementally. The online graph construction procedure stays the same as in Chapter 4 with proposed edge sets E^X, E^s. In this chapter, we only describe the additional edges that allow for matching flexible trajectories. We design our experimental evaluation section to support the following claims, our approach

(i) is able to quickly relocalize after entering mapped area without additional pose prior,

(ii) can handle the kidnapped robot problem,

(iii) be executed in online fashion, and

(iv) robustly deals with loops in the reference sequence.

5.1 Robust image matching costs with CNN features

To align image sequences, we need to match the individual images. Our approach represent each image by a single high-dimensional feature vector. In their extensive study, Chen *et al.* suggest that the 10th layer of the convolutional neural network OverFeat [119] produces robust features for changing environments. The size of the output feature vector depends on the size of the input image. We

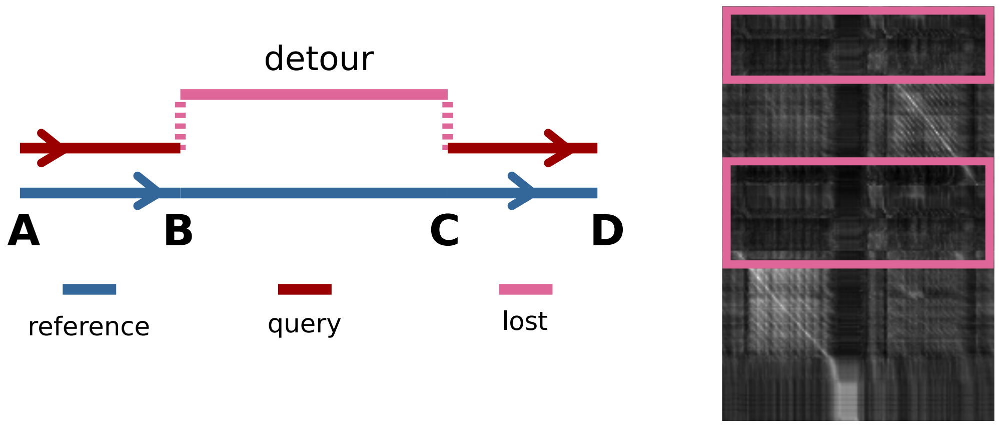


Figure 5.2: Left: Sketch of a query detour during which the robot is lost. Right: An example of a cost matrix, where the query trajectory makes two detours (marked with rectangles).

opted for the smallest acceptable size of 450×250 pixels, which results in feature vector of approx. $200,000$ dimensions. Note, however, that our algorithm is not limited to this kind of features and is applicable with other alternatives as from VGG-16 [121], Net-VLAD [8], or PoseNet [71].

5.2 Efficient relocalization

No localization system is free of failures. Thus, it can happen that a robot gets lost, i.e, it cannot establish a correspondence between its current observations and the model anymore. In our case, this means that the (sub-)sequence of images, which the robot is currently acquiring, cannot be matched to a reference sequence anymore. A common reason for that happening in practice is the fact that the robot moves along a so far unseen trajectory, for example, when leaving the previously mapped area. Figure 5.2 (left) shows a sketch of such a situation and Figure 5.2 (right) depicts how a detour in the query sequence influences the corresponding cost matrix. Whenever, the robot deviates from the mapped route, the cost matrix does not show any bright pattern as in the areas highlighted with pink rectangles. There exists no matching candidate and a robust place recognition system should account for such a situation.

A good relocalization system should be able to detect whenever the robot leaves as well as reenters the previously mapped area in order to resume or restart localization. To detect whether the robot is lost, we analyze the nodes of the best current matching hypothesis within the sliding window over time. If the percentage of the hidden nodes, i.e., images with low similarity, within this window exceeds 80%, i.e., only 20% of the images can be matched to the reference sequence, we consider the robot as *lost*. The size of the window depends on the framerate of the camera and potentially also on the speed of the robot[1].

A straightforward but computationally demanding way to find a reentry point is a brute force search through the whole reference database. Instead, we propose to use a hashing strategy for identifying potential reentry points. This results in comparing a query image only to the subset of database images that are mapped to the same hash key. Hashing techniques are known to be robust and efficient to find image duplicates. In contrast to standard (cryptographic) hashing such as MD5 or SHA1, hashing for image retrieval should assign similar features to the same or neighbouring buckets of the hash table, i.e., to similar hash keys. This property is referred to as *locality sensitive*. Locality sensitive hashing (LSH) proposed by Gionis *et al.* [53] was one of the first approaches to apply hashing for image retrieval problems.

We found that the use of an improved version of the LSH, called Multi-Probe LSH proposed by Lv *et al.* [84] is better suited for image matching tasks than LSH [53]. Multi-Probe LSH builds on top of the LSH but specifies an intelligent strategy to probe specified buckets in multiple hash tables to get the higher probability of finding similar images. In our work, we use Multi-Probe LSH in the following way: The moment the robot is considered *lost*, the algorithm starts to hash every incoming image q_i and looks up candidates $C(q_i)$, stored in the hash buckets, according to the probe strategy. More information about the probing strategy can be found in [84]. After potential matching candidates are retrieved, we add the corresponding nodes to the graph

$$E^{\text{reentry}} = \{(x_{(i-1)j}, x_c)\}_{c \in C(q_i)}, \tag{5.1}$$

where i is the id of the current query image q_i, the term $x_{(i-1)j}$ corresponds to a node representing current best matching hypothesis, and c refer to ids of the images in the reference dataset that were retrieved as matching candidates based on hashing.

Originally, Multi-Probe LSH was designed to match images that were taken under similar conditions and was used with relatively low dimensional features, e.g., 64 or 192 dimensions. In this work, however, we rely on high dimensional features (around 200K dimensions) as they show a better matching performance under changing conditions. This naturally leads to an increase in the querying time for computing the potential candidates from the database.

To tackle this issue, we propose an alternative hashing algorithm designed to explicitly take into account the high-dimensionality of the data and thus improve the querying time without compromising matching performance.

As in every hashing algorithm, the first step is to construct the hash table H. We start with binarizing the feature vectors that represent the images from the

[1]In our experiments using car in an urban environment, the size was set to 10 frames.

reference dataset. We inherit the binarization strategy for CNN features proposed by Arroy *et al.* [11]. To receive the binary values, we first scale the feature vector so that the dimension with largest spread lies in the interval $[0, 255]$ and afterwards all dimensions higher than a middle value 128 are assigned to 1, others 0. Formally:

$$f^{\text{int}} = (f^{\text{cnn}} - \min(f^{\text{cnn}}))\frac{255}{\max(f^{\text{cnn}}) - \min(f^{\text{cnn}})}, \tag{5.2}$$

$$f^{\text{bin}} = \begin{cases} 1, \text{if} \quad f^{\text{int}} >= 128 \\ 0, \text{if} \quad f^{\text{int}} < 128 \end{cases} \tag{5.3}$$

where f^{cnn} is a feature vector from a CNN, f^{int} is normalized feature vector and f^{bin} is final binarized feature vector.

Let us assume we need to hash a dataset that consists with N features indexed with $n \in [0, N]$ and each of the feature has D dimensions indexed with $d \in [0, D]$. Then, every entry in the hash table $H[d]$ stores the set of indices of all the features that have a value of 1 in dimension d:

$$H[d] = \left\{n \mid f_n^{\text{bin}}[d] = 1\right\}. \tag{5.4}$$

In the query phase, the incoming image q gets binarized using the same procedure according to Equation (5.3). Afterwards, we extract a set of indices $A(q)$ of features dimensions that take the value of 1 for image q:

$$A(q) = \{d \mid f_q^{\text{bin}}[d] = 1\}, \tag{5.5}$$

where $|A| = M < D$ and typically $M \ll D$. We collect all feature indices from the hash table that take a value of 1 for the dimension stored in A, i.e.,

$$H[A] = \hat{\cup}_{a \in A} H[a], \tag{5.6}$$

where $\hat{\cup}$ denotes the set union preserving duplicates. We intentionally keep the duplicates in the set to further select those feature candidates that have high number of occurrences in $H[A]$. This represents the fact that the query feature q and candidate features from the database share a substantial set of feature dimensions taking a value of 1. Thus, they are more likely to represent the same place. For an illustration of the hashing procedure, consider the toy-example in Figure 5.3. This small example exhibits the hashing of reference sequence as well as querying the candidates given an image q. Here, for simplicity we consider the reference dataset to consist of three feature vectors, i.e., images. Then we construct a hash table H by considering the dimensions of the feature vectors. The keys of H correspond to the dimension IDs and the buckets store the IDs of the features that have a value 1 in the dimension corresponding to the key of

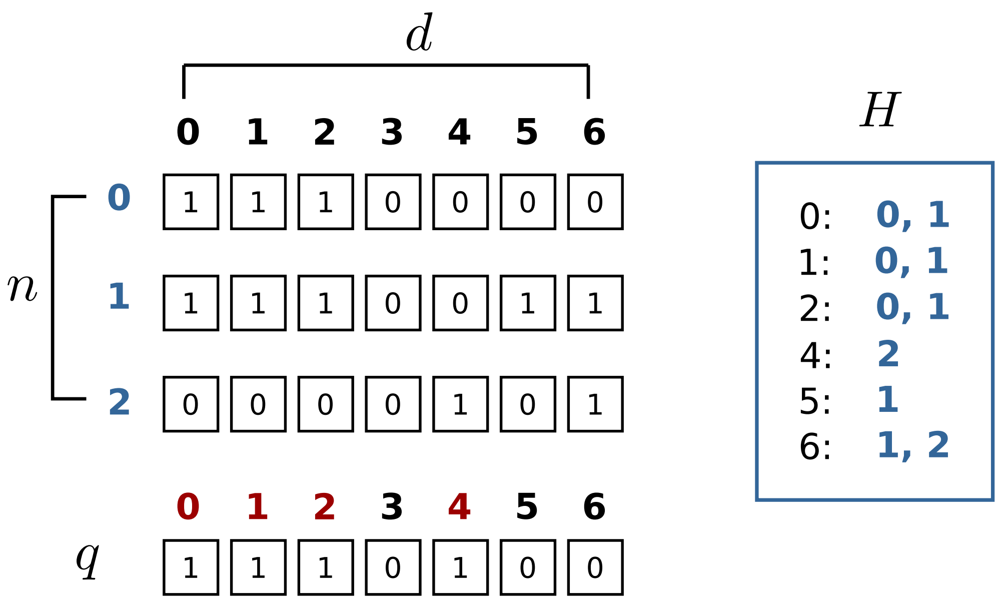


Figure 5.3: Example of the proposed hashing algorithm. Here the dataset consists of 3 feature vectors of dimension 7 each. An entry of hash table $H[2]$ stores the IDs of the feature vectors 0 and 1, since for both of them, dimension 2 has the value of 1. For a query feature, the set of dimensions that take a value of 1 is $A = \{0, 1, 2, 4\}$. By collecting the values from H, the set of potential matching candidates is 0 with occurrence 3, 1 with occurrence 3 and 2 with occurrence 1. The resulting matching candidates for query q are $\{0, 1\}$.

that bucket. The quering procedure starts with selecting a set of dimensions IDs in the query feature that have a value 1, in this case the set is $\{0, 1, 2, 4\}$ and we collect the features from H that also have the following dimensions activated. In this case, we count reference feature 0 and 1 three times and feature 2 once. Thus, this results that the query candidates are image 0 and 1.

5.3 Loopy reference sequences

While recording the reference image sequence, it can happen that the robot moves along the same route multiple times. In practice, this situation occurs frequently when considering a typical urban mapping run using a car. In Figure 5.4 (left), we provide a sketch of a trajectory that shows the situation in which reference trajectory visits the same place in the environment twice from 'B' to 'C'. The cost matrix for a corresponding real world situation is depicted in Figure 5.4 (right). In this case after visiting the place 'C', there are two possibilities to proceed, either visiting 'C-D' or 'C-F'. As the query trajectory follows the 'C-F' route, we can see a brighter pattern on the right lower part of the cost matrix in the Figure 5.4 (right), whereas if the query trajectory followed the 'C-D' direction, the pattern would appear in a left lower part of the matrix. The previous version of our approach cannot handle such a situation flexibly, because the query sequence

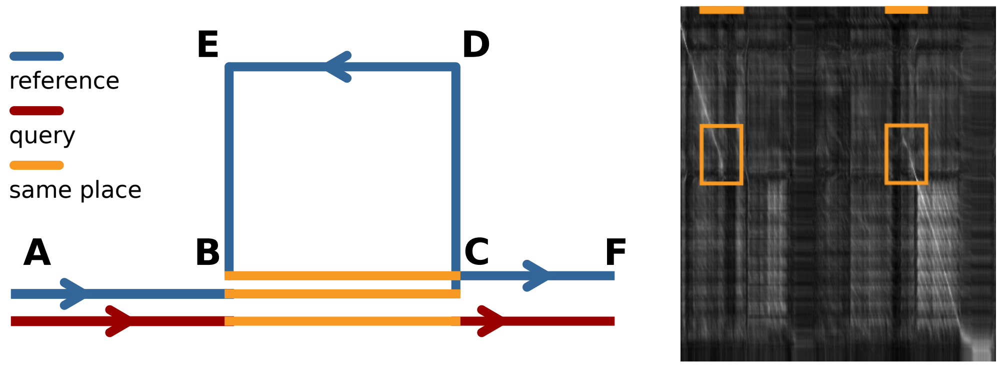


Figure 5.4: Left: Sketch of similar places situation. Right: an example of the cost matrix with a reference dataset visits the same place twice (marked with yellow rectangles).

is expected to roughly follow the reference one.

In this chapter, we extend the approach so that the search algorithm can flexibly "jump" between similar places in the reference sequence visited multiple times. The ability to "jump" is also established when later matching a query sequence by creating the edges in the graph between the nodes that correspond to the similar places in the environment. For example, if we know that image a corresponds to image b within the reference sequence, then whenever the algorithm is requested to expand the graph from the node x_{ia}, it will also expand from the node x_{ib}

$$E^{\mathrm{sim}} = \{(x_{ia}, x_{(i+1)k})\}_{k=b-K,\ldots,b+K}, \tag{5.7}$$

where K is the previously introduced fan-out parameter that compensates for different robot speeds or camera frame rates. The same thoughts hold if the graph gets expanded from the node x_{ib}.

To be able to establish such nodes, we need to identify which images in the reference dataset represent the same place. Since the evaluations must only be performed on the reference dataset, finding of the similar place can be done offline using standard place recognition algorithm such as FABMAP2 [34] since the images stem from the the same appearance within the reference trajectory.

5.4 Experimental evaluation

Our experiments are designed to show the capabilities of our method and to support our key claims, which are: Our approach is able to

(i) quickly relocalize after entering mapped area without additional pose prior,

(ii) handle the kidnapped robot problem,

(iii) be executed in an online fashion, and

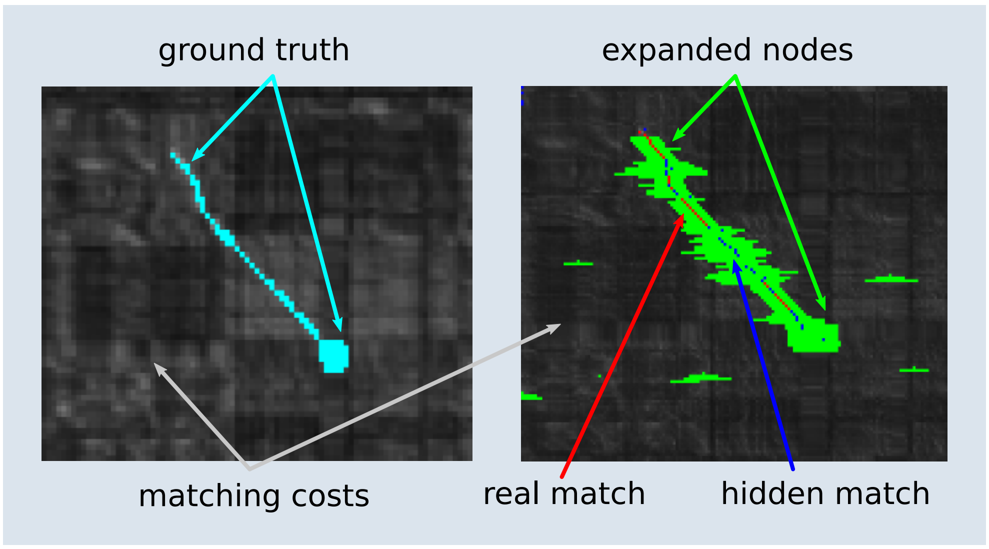


Figure 5.5: Example of possible outputs in our experiments. The cost matrix stores the costs of matching individual images (not used in our algorithm). Expanded nodes - matching costs computed in our algorithm. Real matches - image pairs that represent the same place and hidden match - image pairs that support the path hypothesis, but have low matching cost. Ground truth matches that represent the same place in reality.

(iv) deal with loops in the reference images sequence.

We furthermore provide comparisons to the existing methods [99, 140, 92]. We perform the evaluations on own datasets as well as on publicly available ones. To support our claims, we have selected the datasets that explicitly represent a particular challenge for localization. Some of them stem from the *Freiburg datasets* used in [138, 140] and others from the VPRICE Challenge dataset. Additionally, we collected a more challenging dataset in terms of trajectory shapes. We collected the data in Bonn with a car and a dashboard camera. The query as well as reference trajectory contains several revisits of the same places. Here, we use the datasets collected in the morning with slight rain and overcast as well as in the evening and very late evening on different days. Example images can be seen in Figure 5.1.

Note, that throughout all the experiments the cost matrix was only computed for visualization purposes and is not needed for our algorithm. The matching algorithm only computes the matching costs for the image pairs visualized in green. To enhance the visualization of the cost matrix for larger datasets, we also overlayed the ground truth results, see Figure 5.5 for further notations.

5.4.1 Matching performance and localization recovery

The first set of experiments is designed to show that our approach is able to quickly relocalize after the system has identified that it cannot find matching

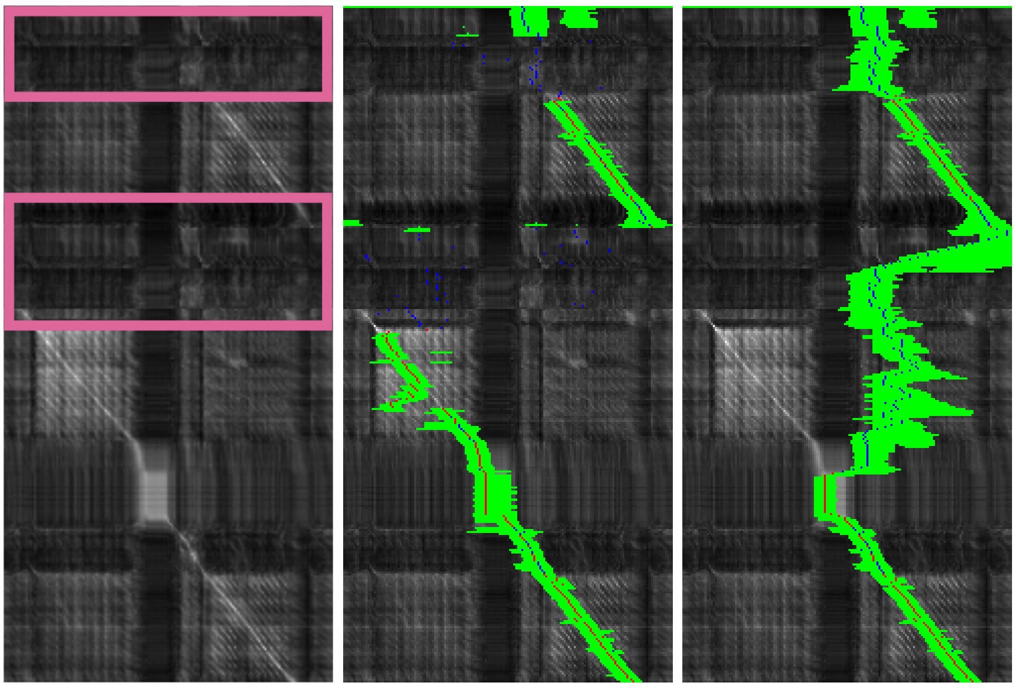


Figure 5.6: Example of a cost matrix for matching two trajectories, where the query trajectory deviates from the reference one twice. Once at the beginning and then in the middle. The places are marked with pink rectangles. Left: matching matrix. Middle: result of proposed algorithm. Right: Result using previous approach. Note that the full cost matrix for matching is only shown for visualization and does not need to be computed by our approach.

images for a certain amount of time. This typically has the reason that the robot is navigating outside the mapped area or that the robot has been "kidnapped and teleported" to a different location in the map. In our experimental evaluations, we consider a robot to be on a detour whenever it is lost but its input query images are consistent between each other. Whereas if the input data changes rapidly, meaning the robot is rapidly in completely different location, we consider this situation to be a "kidnapped or teleported" situation. Our search algorithm though does not differentiate between these situation explicitly and is able to perform robust place recognition in both cases. We provide different experiments to support this claim.

The first experiment is designed to show how our approach can deal with situations, in which the robot is navigating outside the mapped area, i.e., reference sequence. This means that there are no corresponding images with respect to the reference sequence. An example for that can be seen in Figure 5.6 (left) marked by the large rectangles. Figure 5.6 (middle) illustrates that our approach localizes the robot in such a situation (as can be seen from the red matched and blue unmatched pixels). In contrast to that, our previous lazy DA approach finds the matches only partially as is searches in the wrong area of the graph, see Figure 5.6 (right).

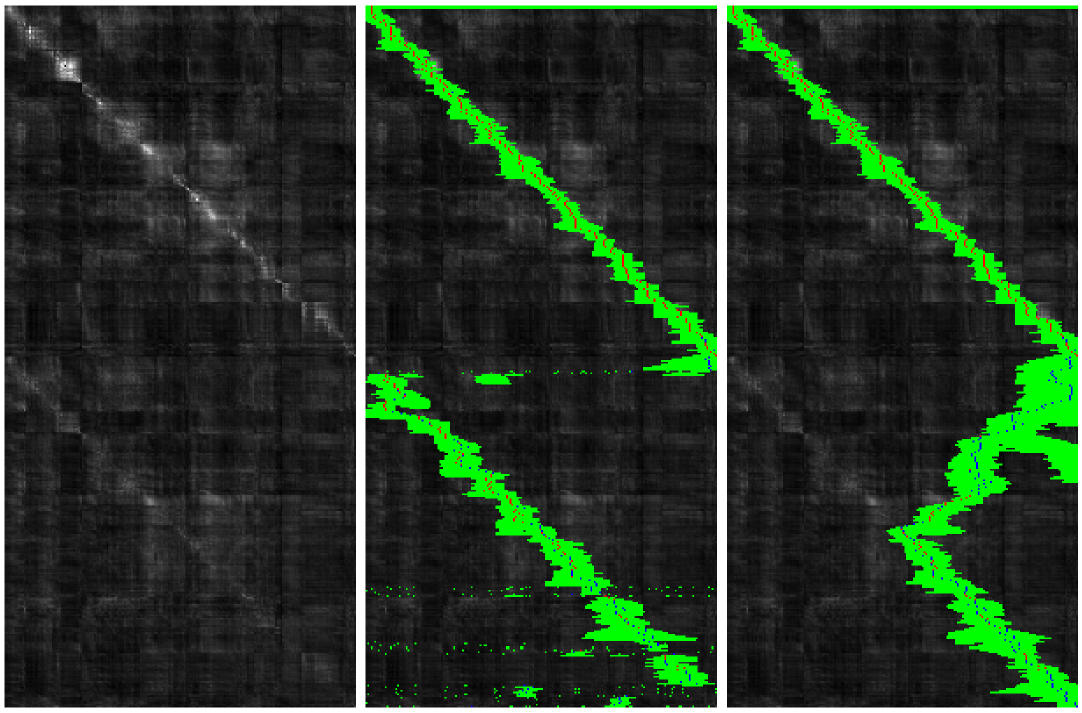


Figure 5.7: Example of the trajectory matching from the VPRICE datasets. Here the query trajectory follows the reference trajectory twice, once during the day time (upper matrix part) and once during the night time (lower matrix part). Left: cost matrix. Middle: result of proposed algorithm. Right: Result using previous approach.

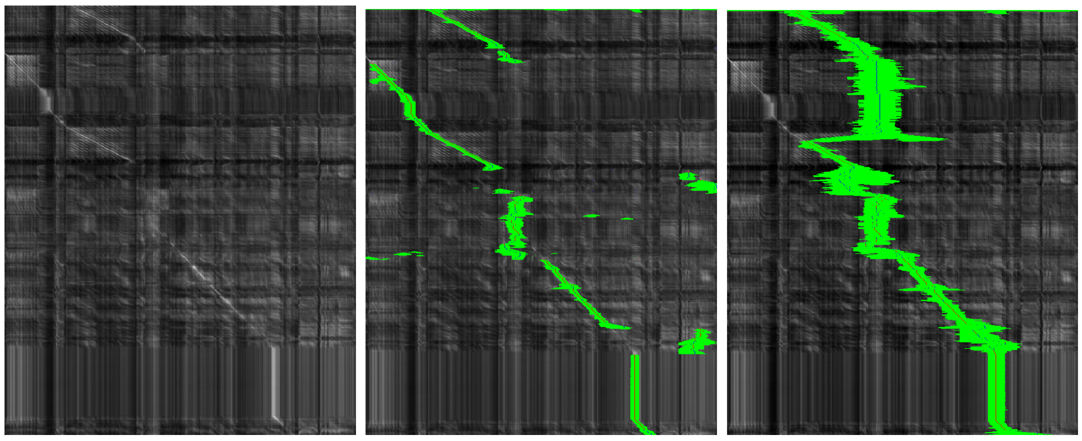


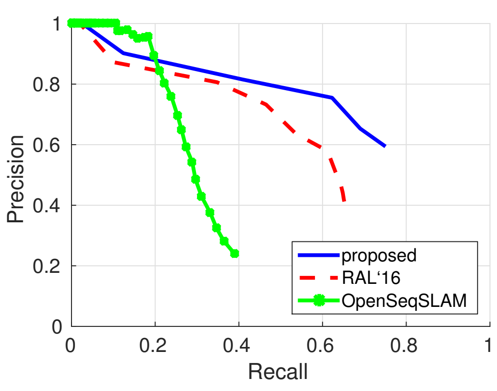
Figure 5.8: Matching example from the Freiburg dataset. The trajectory contains partial revisits of the reference sequence as well as detours in the query sequence. Left: cost matrix. Middle: result of proposed algorithm. Right: Result using previous approach.

We further tested our system on the publicly available VPRICE Challenge dataset to illustrate the handling of the kidnapped robot problem. We depict here the part that contains images where a person is moving with a hand-held camera during the day in the reference sequence and repeats the same path during the day and at night within the query sequence. Since the image sequences between the day and night runs are appended to each other, this corresponds to the kidnapped robot situation, i.e., the robot was teleported from the current location (end of the sequence) to another place (beginning of the sequence). As Figure 5.7 (middle) suggests, also in this case we can dramatically improve the matching quality with respect to our previous approach Figure 5.7 (right). Furthermore, we evaluated our approach on a more challenging dataset that has multiple revisits of the same places in query as well as reference sequence, see Figure 5.8.

We also ran multiple evaluations for the proposed approach. The results are visualized through the precision-recall plots in Figure 5.9. The left image depicts the precision-recall curve for the same dataset as in Figure 5.8 and as can be seen the quality of the result is also better than in our previous approach from Chapter 4 (here labeled as “RAL’16”) exactly due to the ability to detect loops. Figure 5.9 (right) visualizes the results for a dataset with a query loop and relocalization part, as in Figure 5.6 and it clearly outperforms the RAL’16 approach.

The next experiment is performed using more challenging shapes of trajectories, recorded in downtown Bonn. In Figure 5.10, the query sequence was collected in the morning with a slight rain, whereas the reference in the late evening around a week later. The trajectories overlap only partially, which results in the broken up bright patterns in the cost matrix. As can be seen, our approach (middle) is able to find the underlying pattern, i.e., find the matches, whereas our previous approach only performs reliably within the continuous pattern. Figure 5.11 shows the results for another pair of trajectories. The query sequence was collected in the early evening, whereas the reference in the late evening. As can be seen our proposed approach finds the underlying pattern as well as ignores the areas, where the query trajectory deviates from the reference one, and thus no matching images and minimal expansion are expected. Due to inability of our previous method to handle the loops in the trajectories without pose priors, it performs poorly on this dataset.

To evaluate the performance of our approach in a more quantitative way, we compute precision recall as well as F1-score statistics for the given datasets, see in the Table 5.1. As explained in Chapter 2, F1-score is a harmonic mean of the precision and recall. It reaches its maximum value at 1 for perfect precision and recall and worst at 0. We use F1 score as a way to summarize precision-recall with one value to make the comparison between two methods more intuitive. As it can



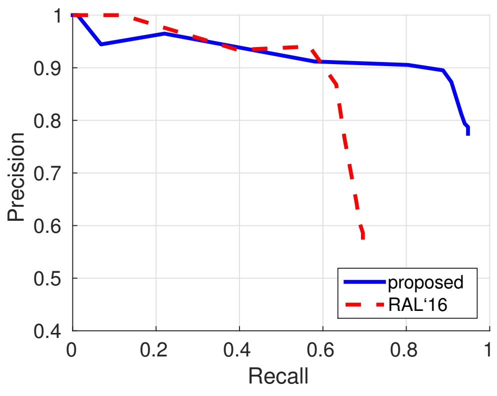


Figure 5.9: Precision recall plots for the dataset with multiple loops in query in references sequences (left) and the dataset with a query connected through the similar places in the reference sequence (right).

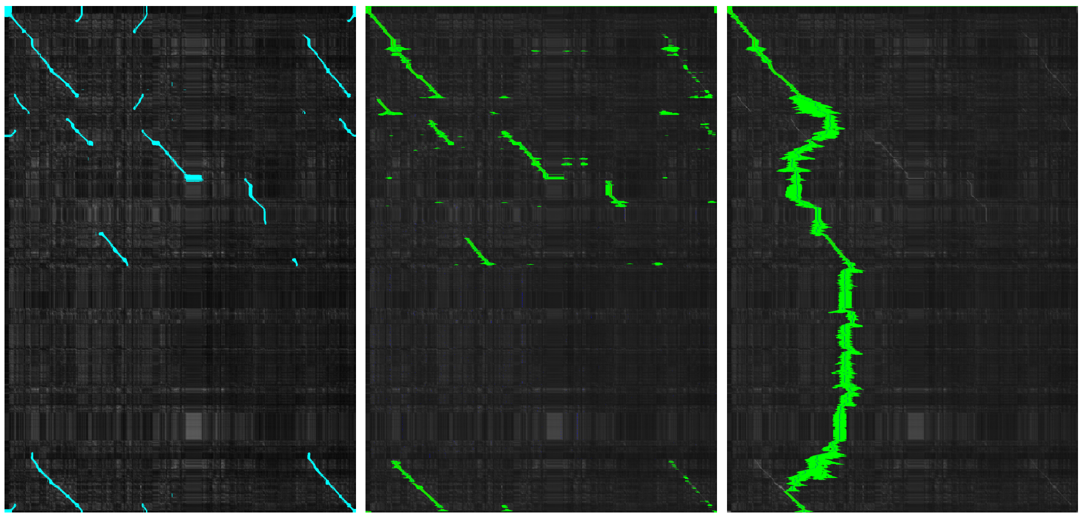


Figure 5.10: Matching example of trajectories from Bonn in which both trajectories contain loops. The query also deviates for reference trajectory for a significant amount of time. Left: cost matrix with ground truth overlayed. Middle: proposed approach; Right: Result using previous approach, see Chapter 4.

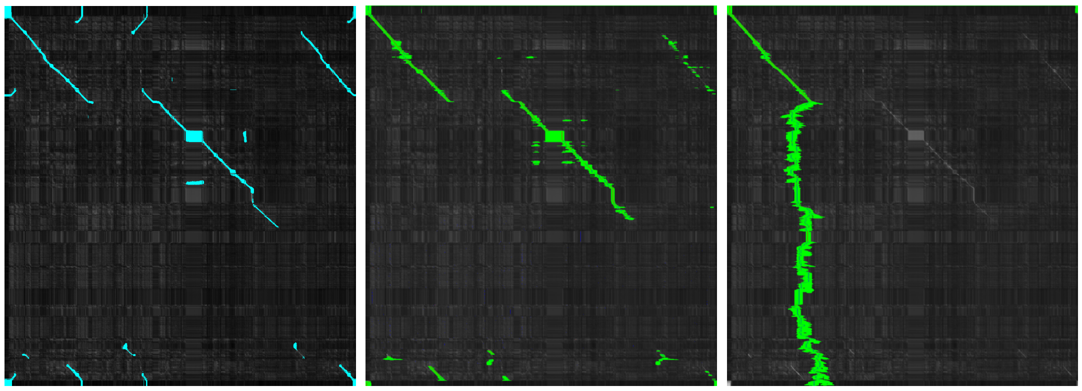


Figure 5.11: Additional matching example from Bonn. Left: cost matrix with ground truth overlayed. Middle: proposed approach; Right: Result using previous approach.

Table 5.1: Qualitative comparisons between the image sequence matching approach from [140] (RAL'16) and proposed method. Trajectories of various shapes. Percentage of expanded nodes with respect to the maximum possible expansion (exp), pr—precision; re—recall.

	RAL'16			Proposed		
	exp	pr; re	F1	exp	pr; re	F1
1	7,3%	0.98; 0.35	0.51	6%	0.95; 0.92	0.93
2	12,6%	0.89; 0.63	0.74	11%	0.89; 0.91	0.86
3	10,3%	0.55; 0.64	0.59	4.2%	0.7; 0.71	0.70
4	2.9%	0.99; 0.31	0.48	1.5%	0.95; 0.76	0.84
5	2.7%	0.72; 0.35	0.46	1.8%	0.8; 0.81	0.80

be seen, by introducing the additional constraints and an efficient relocalization strategy, we are able to increase the number of found image matches (recall) with almost the same precision rate as in our previous chapter. This naturally leads to an increase in accuracy in terms of F1-score.

5.4.2 Hashing comparison

The second experiments is designed to show that the performance of the proposed relocalization strategy and the multi-probe locality sensitive hashing is comparable. We also confirm that the proposed hashing algorithm runs faster for the data with very high dimensional features. We select the following parameters for all of the experiments: number of trees = 1, key size = 10, probe level = 2. From our experience selecting a higher number of trees or key size does not improve the performance of the algorithm, but dramatically increases the computation time. Figure 5.12 depicts only small deviations of the precision recall curves between the locality sensitive hashing (LSH) and the proposed dimension hashing (DH), which indicates that both hashing algorithm perform equally good on multiple datasets. On the other hand, the run time of the individual hashing strategies differ dramatically. For querying a set of candidates the LSH on average takes 120 ms, whereas DH retrieves the candidates in on average in 600 μs, which makes the candidates extraction time around 200 times faster. This speedup has a substantial impact on the overall timing.

Given the OverFeat feature vectors, we obtain the following timings. Processing a single image while being localized takes $2 - 3\,ms$. In contrast, processing a single image while being lost takes $30 - 80\,ms$ using our DH hashing and $150 - 200\,ms$ using LSH. Thus, our approach reduces the runtime for the relocalization by a factor of 2.5 to 5 in our experiments.

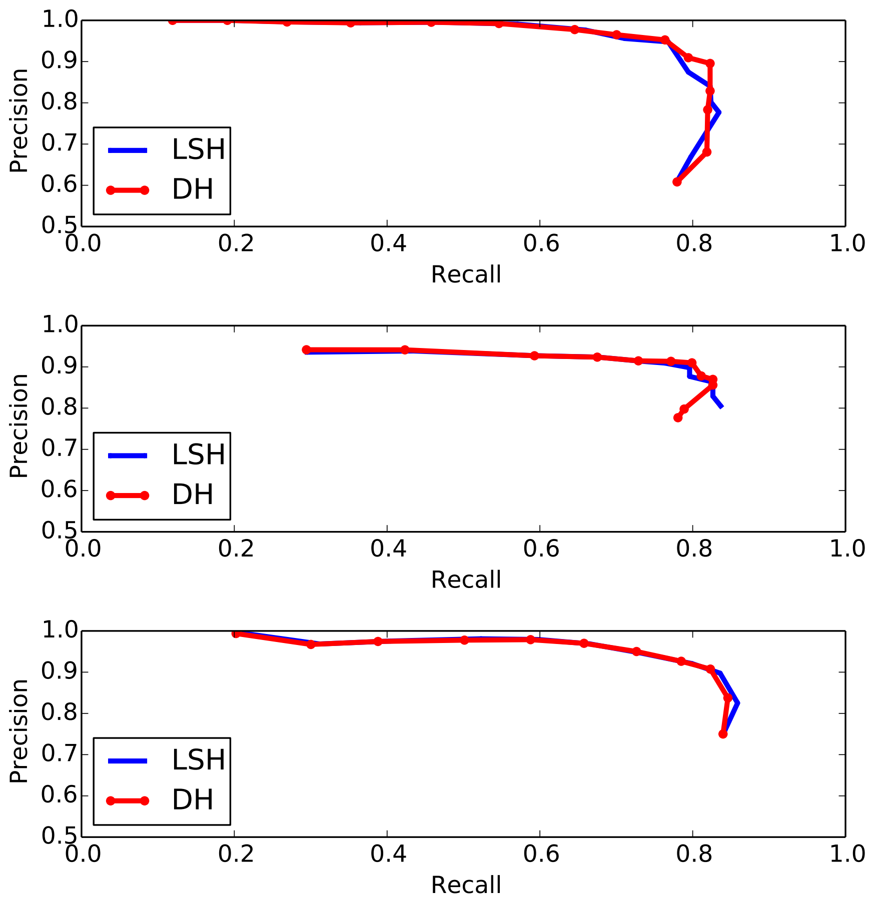


Figure 5.12: Precision recall plots for different pairs of trajectories from Bonn dataset. LSH - results for locality sensitive hashing, DH - proposed hashing algorithm (dimension hashing).

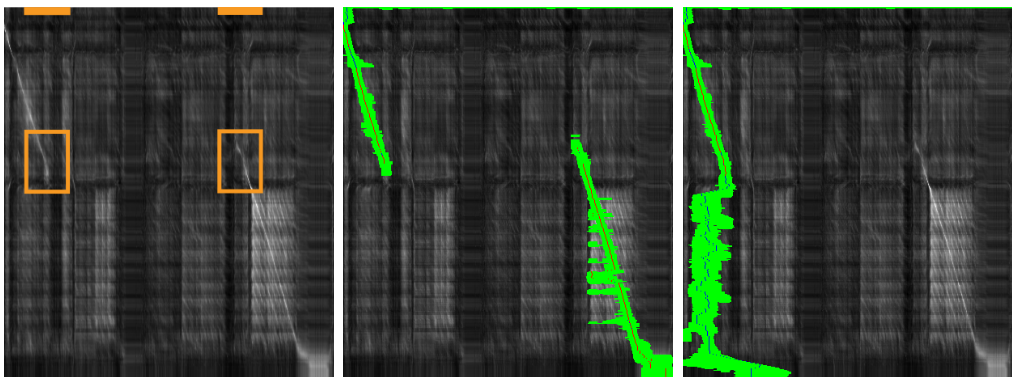


Figure 5.13: Example of trajectories, where the reference sequence traverses the same place in the environment twice (marked with orange squares) and deviates in two difference direction upon exiting similar area. The query trajectory then also passes the ”marked” area and follows one of the direction in the reference sequence. Middle: result of proposed algorithm. Right: Result using previous approach.

5.4.3 Loops in reference sequences

The last experiment is designed to show that taking into account the similarity of the places in the reference sequence leads to better localization results. As can be seen in Figure 5.13 (middle), our proposed approach finds the underlying pattern since the green area overlays the bright pattern depicted in the left part. On the contrary, our previous approach presented in Chapter 4 fails to detect a loop, i.e., the green area does not cover the right part of the bright pattern, see Figure 5.13 (right). Experiments with other datasets show similar results.

5.5 Conclusion

In this chapter, we presented a new approach for quickly finding correspondences between a currently observed image stream and a previously recorded image sequence under strong appearance changes. Related to previous chapters, we build a data association graph incrementally and search for a data association sequence using an effective search heuristic. The work proposed here overcomes two key limitations of our previous method. First, it provides an efficient way for relocalizing the robot in situations, in which it got lost. For example, after the robot has left the previously mapped areas and is reentering the known part of the environment or to solve the kidnapped robot problem. Second, our new approach can deal with loops in the reference sequences effectively without additional pose priors, like consumer GPS. We implemented and evaluated our approach on different publicly available datasets. Our evaluations and comparisons show that we can handle the above mentioned situations, which could not been solved with the approach in Chapter 4. We furthermore show through the experiments that the our approach runs online, provides an effective image matching, and supports all claims made in this chapter.

Chapter 6

Visual place recognition against multi-sequence maps

Our approach presented in previous chapters has proven to be a reliable solution for visual place recognition in changing environments for sequence-to-sequence matching. In most realistic scenarios, however, one reference trajectory is typically not sufficient to cover the operational environment of the vehicle. A more realistic scenario is when the vehicle operates traveling over multiple routes over extended periods of time. In this case, it is unlikely that the vehicle takes the same routes all the time. This results in having a collection of multiple unsynchronized image sequences as a reference dataset. All of those sequences exhibit potentially different representation of the places and we would like to use this information to improve the chances of successful visual place recognition. Since we had successfully applied our previous method to visual place recognition between two sequences of images, the next natural step is to reformulate the sequence-to-sequence matching to deal with multiple image sequences in parallel. In this chapter, we describe an extension our approach to deal with multiple reference trajectories and in this way allow our system to grow a map of places. This improves the coverage of the environment both, in terms of space and different appearances.

The particular challenge of the current setup of the reference dataset is that our previous graph construction strategy is not directly applicable to work with the map form of the reference dataset. We cannot use the same formulation for the edges, since now we potentially have multiple reference sequences. In this chapter, we describe how to adapt the construction of the graph structure so that it is able to operate in online fashion with the reference dataset consisting of the multiple sequences of images.

One of the key features that our search system should have is the ability to alternate the search between multiple reference sequences, if required. We achieve this ability by exploiting the relocalization strategy presented in Chapter 5. In the previous chapter, we proposed a relocalization technique based on the feature dimensionality analysis and inverted index structure. Inverted index is a data structure that stores a mapping from the content, in our case the dimension of the feature, to the location. i.e., the feature ID which has this dimension activated. This method works better and faster than Multi-Probe Locality Sensitive hashing (LSH) [84] for very high-dimensional feature vectors, for example features from OverFeat convolutional neural network [119] or VGG-16 [121] with around 200,000 and 25,000 dimensional feature vectors respectively. In this chapter, we opted for newer smaller feature vectors, namely the feature vectors obtained from NetVLAD convolutional neural network [8]. The advantages of these features are comparably small size (4092) and robustness against visual appearance changes, especially dramatic viewpoint changes. Due to the vector size, we selected Multi-Probe Locality Sensitive hashing as a fast alternative to perform relocalization. Whenever the robot is lost, defined by the fact that there are more than 80% hidden nodes in the sliding window around current best match, we pick the top candidates from all the images using Multi-Probe LSH and select the most promising one. This matching candidate becomes a real node if the respective matching cost is lower than non-matching threshold m and a hidden node otherwise. Afterwards, we connect it to the current best matching hypothesis. We consider the relocalization to be successful if no more than 80% of the nodes within the sliding window of 5 frames in path hypothesis are hidden nodes.

The novelty of this work is the fact that the reference dataset can now be a map of multiple sequences of the images that are collected in different points in time, recorded from different viewpoints and with different frame rates. Example of such a setup is depicted in the Figure 6.1, where the reference dataset consists of three trajectories collected in the morning, day, and early evening as can be seen from the images to the left. We color coded the trajectories with different colors: red, blue, and purple. The query is marked with black and was collected during the sunny day as can be seen from the top picture to the left. To the right, you can see the overlayed GPS coordinates (used only for visualization purposes here) of the collected images that have the corresponding color coding. Note that the reference trajectories only partially overlap between each other and at the same time the query trajectory passes a lot of places that were not covered by any reference trajectory, i.e., the query trajectory makes a detour. In subsequent sections, we describe how to adopt and reformulate the graph construction scheme to be able to work with a map of multiple reference image sequences as well as provide extensive qualitative and quantitative experiments that show that our approach is able to perform a robust visual place recognition for the changing outdoor environments by localizing against (i) multiple image sequences collected

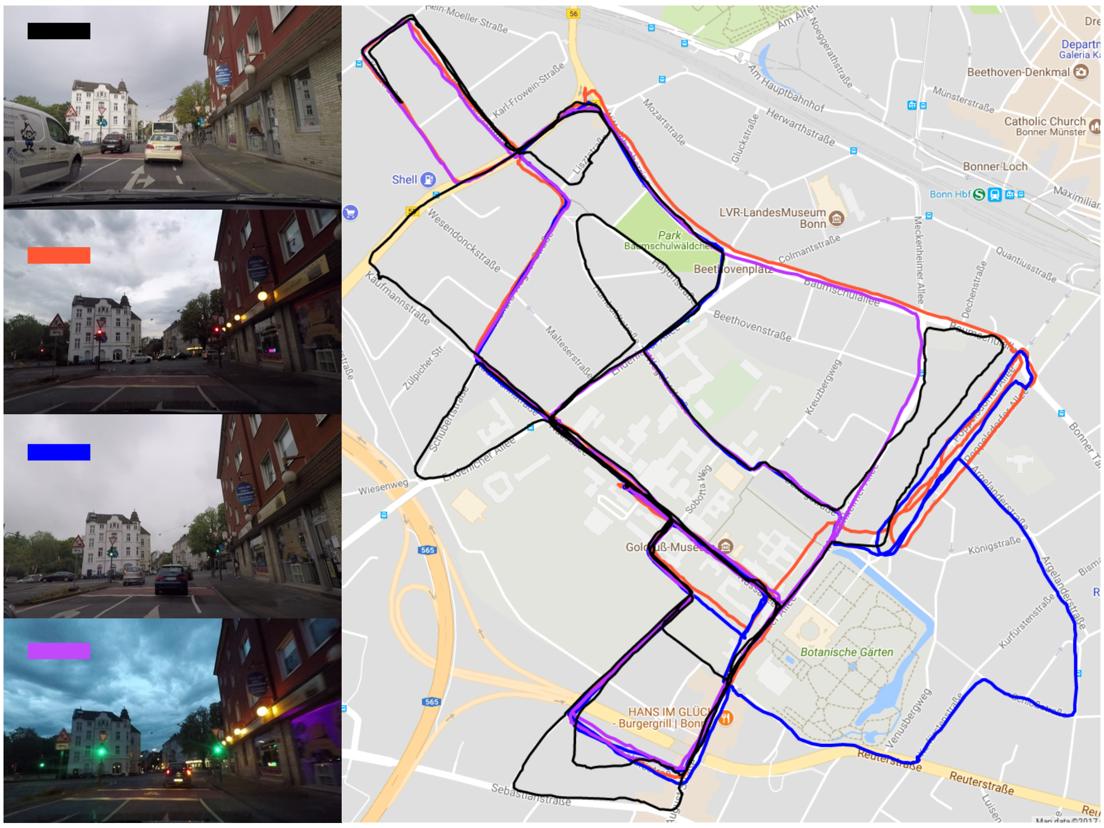


Figure 6.1: Example dataset, where the map consists of three trajectories collected in different points in time as well as in different weather conditions. The corresponding image sequences cover also partially different routes. The query trajectory depicted in black.

with similar camera setup and (ii) imagery coming from different modalities, sequences collected on bike as well as in the car.

6.1 Adapting the data association graph structure

In this section, we propose how to adapt the graph construction scheme to take into account multiple reference sequences. As we assume that the reference image sequences have been collected beforehand, we can execute a pre-processing step to synchronize the sequences by performing the pairwise sequence-to-sequence matching, i.e., using our approach described in previous chapters, for all reference sequences. From this matching information, we can define an in- reference matching function $M(j,t)$ that returns for the image j from reference trajectory t all images (image index and trajectory index) that match to image j from t among the other reference sequences. If there are no corresponding images, the function returns the empty set. To enhance the localizability capabilities of the system, we have changed the representation of the map in comparison to Chapter 4 to be able to match against multiple image sequences. To incorporate a map of multiple reference sequences into our search procedure, we need to redefine the edges of the data association graph. This also leads to a slight change into the notation. Here, every node in the "one-to-many" strategy is specified as x^i_{jt}, where i refers to the image id in the query sequence. The subscript jt refers to the image with ID j in the reference sequence t, see Figure 6.2 (right) for visualization.

The search starts with constructing the source node x^s. Since from the beginning the robot has no information about its location, we start with a relocalization action. This includes hashing the first query image q_0 and retrieving potential candidates $C(q_0)$ from a hash table , forming the first type of edges called E^s in the data association graph, given by:

$$E^s = \{(x^s, x^0_c)\}_{c \in C(q_0)}. \tag{6.1}$$

During the search, every node that is *worth expanding* given the heuristic proposed in Chapter 4, Equation (4.4) is connected to its children within the same sequence using the set of edges E^X:

$$E^X = \{(x^i_{jt}, x^{i+1}_{kt})\}_{k=j-K,\ldots,j+K}. \tag{6.2}$$

At the same time, we allow for transitions between reference sequences given the identified correspondences through the function $M()$. Thus, the set of edges E^m interconnects the images along the different reference trajectories, i.e.,

$$E^m = \{(x^i_{jt}, x^{i+1}_{\gamma})\}_{\gamma=M(j,t)-K,\ldots,M(j,t)+K}. \tag{6.3}$$

Finally, in case the vehicle looses track of its localization, either due to a failure or due to the fact that it had left the previously mapped area, the last node of the current best matching hypothesis x^i_* is connected to the set of the candidates obtained from our hashing scheme:

$$E^l = \{(x^i_*, x^{i+1}_c)\}_{c \in C(q_{i+1})}. \tag{6.4}$$

Each edge $e \in E$, independently of type, has a weight $w(e)$ assigned to it. The weight is inverse proportional to the similarity score between the images. If an edge connects two nodes $(x^i_{jt}, x^{i+1}_{j't'})$, the weight $w(e) = 1/z^{i+1}_{j't'}$, where $z^{i+1}_{j't'}$ is the cosine distance between query image feature $i+1$ and reference image feature (j', t'). As mentioned before, we use NetVLAD features that were designed to be compared with the Euclidean distance, so $w(e)$ is given by the Euclidean distance between two feature vectors within this chapter.

By introducing new edges and weights, we described how to adapt the graph construction scheme, namely the update step from the matching algorithm presented in Chapter 4, to be able to account for multiple sequences. All other steps

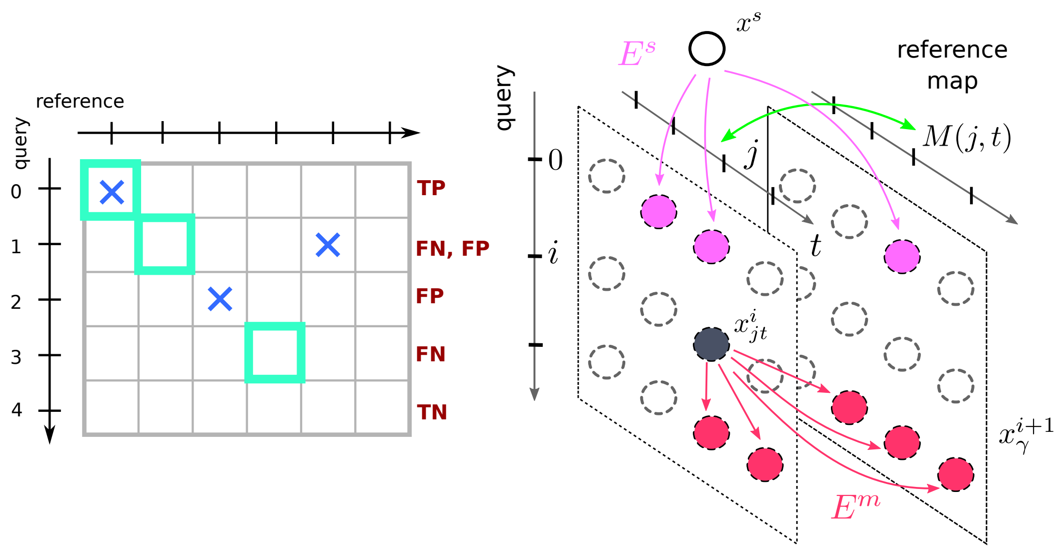


Figure 6.2: Left: Evaluating per image data associations between query and reference sequences. Blue crosses denote the matches found by our algorithm. Green squares denote the ground truth solutions. TP-*true positive*, TN - *true negative*, FP - *false positive*, and FN - *false negative*. Right: E^s (pink circles) relocalization edges; E^m - correspondence edges; $M(j,t)$ (green arrow) corresponds to the images that represent the same place from different image sequences.

remain the same, which shows how easy it is to extend our online graph based sequence matching algorithm to more general cases.

Furthermore, the proposed approach operates also on the reference sequences that are not synchronized. In this case, whenever the camera leaves the current sequence it was localized against the relocalization step will be triggered. This extended data association graph design is built up on the fly while new images are obtained and allows us to consider multiple reference trajectories jointly. Only the function $M()$, for synchronized case, is computed beforehand as it does not depend on the query image sequence.

In sum, this leads to a sequence-based visual place recognition approach that localizes against map consisting of multiple sequences. Our approach allows for better localization in challenging environments, since we have more information available through multiple sequences, while not imposing any restrictions on the map data collections process.

6.2 Experimental evaluation

We designed the experimental evaluations to confirm the following properties: Our approach is able to efficiently relocalize against

(i) multiple image sequences collected with similar camera setup, and

(ii) imagery coming from different modalities, sequences collected on bike as well as in the car.

Note, there are no constraints on shape, length, or visual change of the trajectories.

6.2.1 Evaluation setup

To describe our evaluation setup, we first analyze the output of the matching algorithm. Our place recognition system reports for every query image if there is a matching image in the reference dataset as well as what exactly that image is. Additionally, we have an estimate about the ground truth matches. We consider two images to match if their GPS coordinates lie within the 30 meters range. There are 5 types of situations that can happen while evaluating a match for a query image, see Figure 6.2.

- Case 1. *True positive* (TP) occurs when the algorithm found a match that is in the set of ground truth matches as for the query image 0 in Figure 6.2.
- Case 2. *False negative* (FN) there is a match for a query image 3 in the dataset, but the algorithm failed to detect it.
- Case 3. *False positive* (FP) when the algorithm has detected a match but there should be no match for a query image, as for image 2. This typically happens when the query trajectory makes a detour from the reference ones.
- Case 4. *True negative* (TN) there is no match in the ground truth set and the algorithm has correctly not found one as it is the case for image 4.
- Case 5. A fifth possible situation is that there exists at least one ground truth image correspondence for a query image but the algorithm failed to detect it and found a wrong match instead as for image 1. Then, by our definition this match is a false positive as well as false negative. To not penalize this situation twice, we increment the set of FP and FN not by 1 as for the other cases but by 0.5.

Afterwards, we compute the accuracy for individual dataset as

$$\text{acc} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.5}$$

Since the performance of our search algorithm depends on the non-matching parameter m, we vary this parameter to evaluate the behavior of the search. This allows for obtaining the accuracy curve.

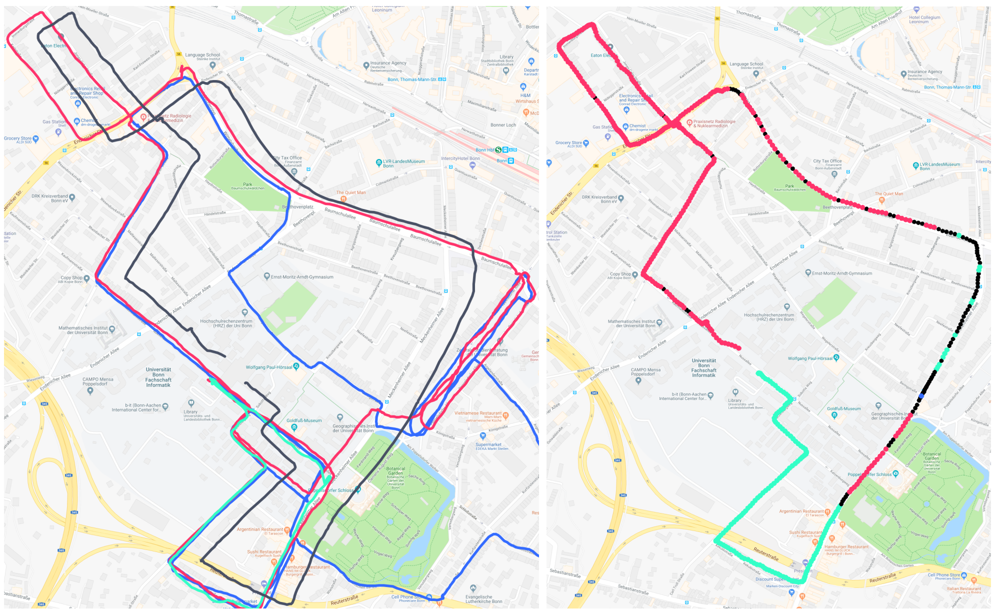


Figure 6.3: Left: Query trajectory drawn in black and shifted artificially for better visibility, others are reference trajectory. Right: Query trajectory painted in the colors of the reference trajectories it was matched to.

To provide comparative evaluations, we deploy an open-source version of FABMAP [54] algorithm as well as open-source version of DBow2 [49]. To adjust it to our setup, we trained the vocabulary for both approaches on several extra datasets that exhibit similar visual conditions, like viewpoint changes, changes in environmental appearance etc. We used the default provided parameters for both approaches. Since FABMAP and DBow2 do not explicitly work with reference data represented with multiple trajectories, throughout all our experiments we stacked reference trajectories into a single big trajectory. For FABMAP, we select for every query image a matching image if it has the highest probability. Whenever the probability exceeds the predefined matching threshold, then the match is considered valid, otherwise the query match does not have a matching image in the reference dataset. To obtain the accuracy curve, we vary the matching probability threshold from 0 to 1. The same evaluation strategy holds for the DBow2 but there we threshold by the score and not probability.

Additionally, we compare our search strategy against the algorithm that compares every query image to every image in the reference dataset—a property that an online approach cannot have. This algorithm operates with the same features as ours, but selects the match with the smallest cost from all the reference sequences, making it a “fully informed search”, labeled as FI in the plots, also known as exhaustive search. As also for FABMAP, a match for a query image is accepted if the matching cost is smaller than a non-matching cost parameter m. The curve is generated by varying the non-match parameter.

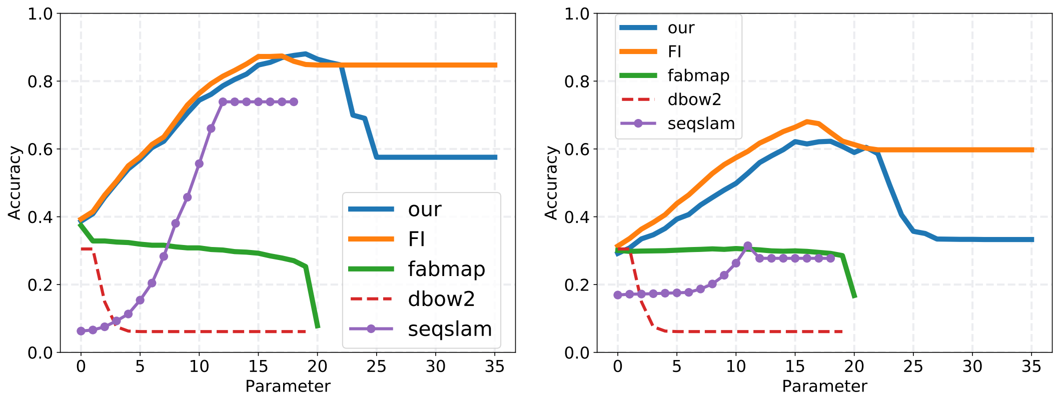


Figure 6.4: Left: Accuracy plot for the dataset in Figure 6.3. Right: accuracy for a larger query sequence against three reference trajectories, depicted in Figure 6.1.

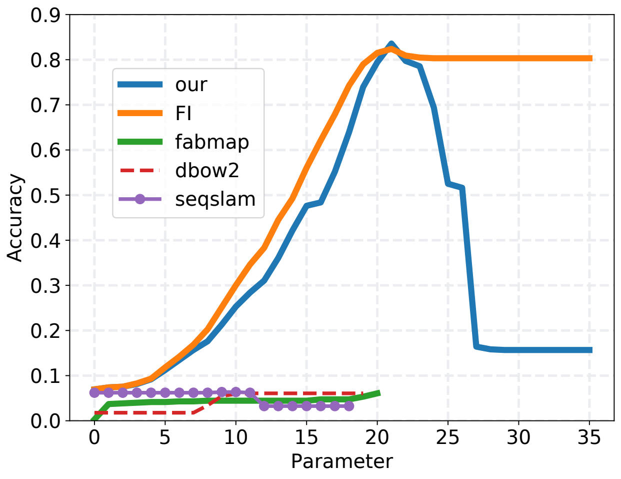
Furthermore, we have compared our approach to the state of the art approach in visual place recognition under dramatic visual appearance changes, SeqSLAM [92]. Since SeqSLAM is designed to match two sequences of images, we applied the same strategy as for FABMAP and DBow2 to convert our reference map of multiple sequences into one reference sequence. For clarification, the x-axis ”Parameter” on all accuracy plots correspond to the non-matching cost m for our algorithm and FI search and to the probability threshold for FABMAP. The scale only shows how many parameters are used.

6.2.2 Datasets

To evaluate our approach we have collected several types of datasets. We used goPro Hero 6 camera with additional GPS used for evaluations. We collected sequences using a car as well as a bike. For the sequences collected from within a car, the camera was mounted on the front window, and for the dataset obtained with a bike, the camera was mounted on the helmet. Throughout our experiments, we use the sequences of different length as well as sequences that exhibit different environmental changes, for example rain, overcast, morning, evening, different viewing direction etc. Throughout our experiments, the images were extracted at 1 fps.

6.2.3 Experimental results

In the first experiment, we show that our system is able to recognize previously visited places within multiple reference trajectories. The query trajectory consists of 636 images and was collected during the evening. There are three reference trajectories (around 3k images in total) of different shapes that were collected during the rainy morning, early and late evening respectfully. Figure 6.3 shows the GPS



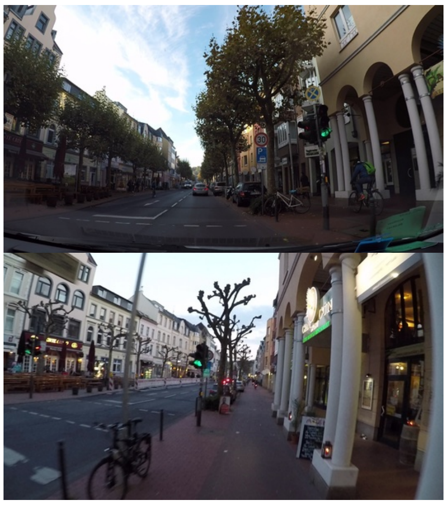


Figure 6.5: Left: Accuracy curve. Right: Matching image pair from query (bike) down and reference (car) up.

trajectories of the reference sequences (pink, blue, cyan) as well as query (black) sequence. Figure 6.3 (right) shows the trajectory of a query sequence drawn with the color of reference trajectory it was localized against. Black corresponds to the fact that no reference image was found. As can be seen, most of the time the query sequence is localized successfully against reference trajectories (pink or cyan) as well as almost no correct place associations made for the cases, where query trajectory deviates from any reference trajectories, the part where black trajectory deviates from all reference trajectories. Quantitative evaluations are shown in Figure 6.4 (left). As can be seen, our approach shows similar accuracy to the fully informed matching (FI) and outperforms the FAB-MAP and DBoW2 approaches, because the accuracy curve for our approach reaches 88%, whereas accuracy curve for FABMAP reaches 40% at maximum and 35% for DBoW2 respectively. Figure 6.4 (right) depicts accuracy results for another dataset depicted in Figure 6.1 with query trajectory of 2,022 images and shows similar performance of our algorithm.

The second experiment is designed to show that our search approach is able to perform reliable visual place recognition for the cases when trajectories have been collected using camera on the dashboard of a car and on a helmet of the bicyclist. This setup imposes a particular viewpoint challenge, see Figure 6.5 (right) for an example of a matching pair between a query (bike) image at the bottom and a reference (car) image on the top, successfully found by our algorithm. Figure 6.5 (left) shows that our approach has a comparable performance to the FI with around 80% accuracy and they both outperform FAB-MAP, DBoW2, and SeqSLAM for this dataset.

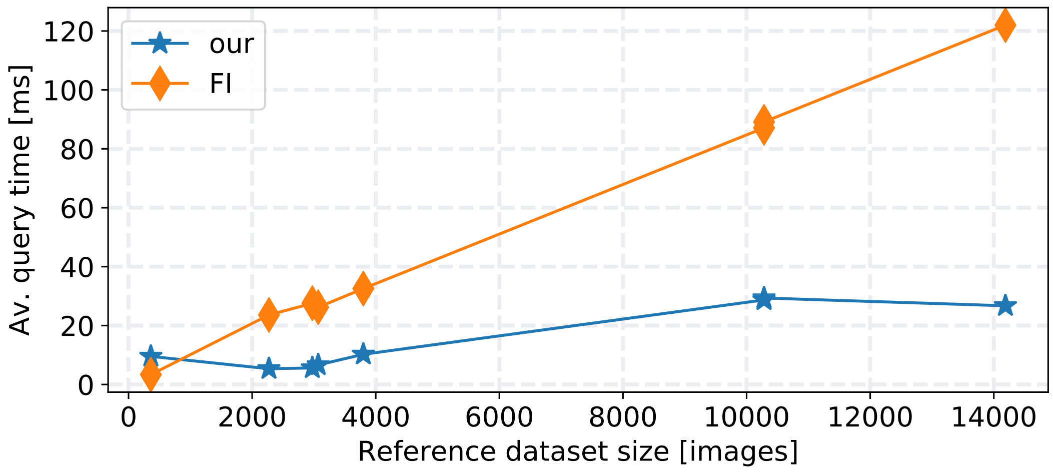


Figure 6.6: Comparison of the running time for the proposed algorithm (our) and for the fully informed search (FI). Every point depicts the average time to find a match for a query image for the reference datasets of various sizes.

6.2.4 Timings

In this experiment, we confirm that the proposed algorithm allows for faster image matching than fully informed search. We performed the runtime measuring on the mentioned datasets by averaging the performance of individual datasets within the 10 runs and selecting the non-matching parameter that leads to the highest accuracy. Figure 6.6 shows average matching time for a query image with respect to the reference datasets of the different size, e.g., total number of images in reference sequences. Since FI algorithm matches a query image to every image in the reference dataset, the time needed for finding a match grows with increasing dataset size, whereas our approach experiences only slight increase in running time. In general, the performance of our algorithm is independent from the size of the reference dataset. To show this, we augmented the dataset that had 10,000 images with 4,000 additional ones and leaving the query trajectory the same. As can be seen, the runtime of the FI algorithm for the second dataset is increased whereas for our algorithm stayed almost the same. The relocalization step in our approach, however, may lead to an increase in runtime. Whenever the robot is lost, querying the candidate locations in performed via a variant of hashing, whose performance is directly influenced by the size of the dataset. Also matching capability of the features influence the search speed. The more distinctive the matching scores are, e.g., the bigger the score difference between the matching pairs and non-matching pairs is, the faster the search will reject unpromising candidates and thus runtime will decrease.

6.2.5 Limitations

Since the matching performance of our algorithm depends on the non-matching parameter m, selecting it correctly may be not an obvious thing to do. Also we observe a performance degradation whenever the visual appearance changes within the query sequence. For example, if the sequence starts at the evening and matching continues for a long time, so that it gets dark outside, the same non-matching parameter that reasonably described the non-matchiness of the sequence is no longer valid.

6.3 Conclusion

This chapter presented a novel approach for quickly finding correspondences between a currently observed image stream and a map of several previously recorded image sequences given substantial appearance changes. Matching is performed through an informed search in a data association graph that is built incrementally. By deploying hashing technique, we are able to relocalize the robot if it is lost as well as between multiple image sequences. Our evaluations show that we can perform place recognition faster than offline, fully informed search with the comparable or better matching performance.

6.4 Conclusion for Part I of this thesis

In this first part, we have presented a novel approach for visual place recognition in changing environments. We approach the visual place recognition problem as an image sequence matching problem where the sequences can be collected at different points in time, under changing weather conditions, or other drastic visual appearance changes induced by seasonal variations. One novelty lies in formulating and solving this sequence matching problem as a graph search problem. In brief, the nodes in the graph represent potential image associations between two sequences of images. The shortest path through such a data association graph provides the image correspondences between two sequences, and thus automatically reports which images in the query sequence correspond to which image in the reference sequence. Thus, this solves the task of visual place recognition systems. In Chapter 3, we provided the required details about the construction of the graph for the cases in which two complete image sequences are given. Moreover, we showed how incorporating the information about the robot's pose prior allows our graph-based place recognition system to efficiently handle loops in the trajectories, which other state-of-the-art methods like SeqSLAM are not able to do. By using either complete or sparse cost matrices, we are able to achieve $70 - 95\%$ precision over $80 - 95\%$ recall on various challenging datasets as well as outperform state-of-the-art methods like SeqSlam and FABMAP.

In Chapter 4, the novelty is the extension of our graph-based search algorithm to operate in an online fashion. This brings our approach closer to real world applications as it enables us to make image association decisions for every incoming query image online as opposed to the previous approach that requires the complete query sequence to be given. The previous approach computes the full matching matrix and only afterwards finds image correspondences between the sequences. The main contribution of this chapter is the reformulation of the graph construction process by using a lazy data association principle that allows for simultaneously constructing and searching in the graph. Furthermore, we propose an efficient search heuristic that results in dramatic computational savings and enables the online operation of our place recognition approach. As our results suggest, we are able to reduce the image comparisons by up to 99.5% and at the same time can perform successful place recognition.

In Chapter 5, we relaxed the constraint that image sequences should be weakly synchronized and allow for place recognition with flexible trajectories. The main novelty of this chapter lies in proposing an efficient hashing algorithm for fast and robust feature retrieval for very high-dimensional feature vectors. Deploying our hashing technique for the relocalization actions allows us to successfully determine when the robot re-enters previously mapped areas after a period of a detour as well as handle the kidnapped robot problem. The ability to relocalize in arbitrary situations enabled for performing robust place recognition with loopy trajectories without additional information about pose priors.

In Chapter 6, we focused on place recognition in cases where there is more than one reference trajectory provided. It frequently happens in the real world that a vehicle travels not just along one route but through multiple different ones with a various amount of overlap between them. The main contribution of this chapter is the formulation of the graph definition to enable searching against the map of reference sequences as opposed to a single reference sequence. This allows for better map coverage and thus for better localization. The introduction of the hashing-based relocalization strategy plays a major role in the search by providing a smooth transitioning between the reference sequences also for the cases when the reference sequences were not previously synchronized. As our experiments suggest, by incorporating more reference sequences, we are able to achieve a performance accuracy of $60 - 80\%$ depending on the datasets and outperform other state-of-the-art methods as SeqSLAM, FABMAP, and DBow.

To share our method with the community, to support open and reproducible research, and to boost further research in the area of visual place recognition, we have open sourced our software developments. The code can be found on GitHub under:

- `https://github.com/ovysotska/image_sequence_matcher`
- `https://github.com/PRBonn/online_place_recognition`
- `https://github.com/PRBonn/vpr_relocalization`

Furthermore, we released our datasets at:
`http://www.ipb.uni-bonn.de/data/visual-place-recognition-datasets/`

Part II

Exploiting publicly available information

Chapter 7

Visual place recognition against Street View data

In part Part I of this thesis, we have shown how to approach the problem of visual place recognition in changing environment by taking sequence information into account. In Chapter 6 we have shown that having a map of reference sequences as oppose to a single sequence improves the place recognition performance. By incorporating multiple sequences, we are able to enhance our map by an additional representation of places, in case trajectories overlap, or simply add more places to the map, in case sequences pass through previously unseen places. In both cases, we grow the map and thus the coverage of the environment in terms of space and visual appearance. Basically, the more information we add the better should be the performance of our matching algorithm. However, creating large maps of the environment comes at the cost of explicitly collecting the sequences. This may become a time and resource consuming operation.

Nowadays, there already exists a vast majority of maps, like Google Maps, OpenStreetMap, Bing Maps, etc. Unfortunately, these maps are mainly created for people and thus need adaptation to be suitable for robot operation. In this part of the thesis, we focus on how to use the already available maps potentially created with different sensors and a different purpose in mind within robotics.

In this chapter, we show how to use road information from OpenStreetMap [67] in combination with Google Street View (GSV) [66] imagery to extend our map of image sequences. This map can then successfully be deployed as a reference dataset within visual place recognition approach using multi-sequence map presented in Chapter 6. Our approach is not limited to work with GSV imagery only, it is generic enough to account for image sequences that come from different publicly or commercially available sources.

7.1 Leveraging Google Street View for multi-trajectory visual place recognition

In this section, we provide a detailed overview how to adapt the information available form OpenStreetMap and GSV to form a map of image sequences suitable for our place recognition approach.

Google Street View is a technology within Google Maps that stores panoramic imagery from positions along streets almost everywhere on the planet. Using the Google Street View API, users can obtain a certain image of a street by providing several parameters. We query images from GSV given a GPS coordinate as well as a selected heading. With these parameters, one can obtain an image for a given position and orientation in the world. The panoramas stored in GSV are only partially ordered, since every queried image stores the identifier to the previous and next panorama. This ordering enables the visually appealing warping from one panorama image to the next one in the street view, which can be visualized in the GSV browser, however, it does not explicitly form sequences. To exploit GSV imagery within our place recognition framework, we perform a sequence of transformations that turn a set of unordered images into the map of image sequences. We form image sequences by combining the Street View Images along streets. Then each individual street turns into a reference image sequence. The synchronization of the reference sequences comes naturally from incorporating the information about street crossings. Further, we will describe a way to arrange a set of images into sequences.

As a first step, we extract the GPS coordinates of the streets from OpenStreetMap. Obtaining the street coordinates from OpenStreetMap requires only parsing the provided xml file. The OpenStreetMap API provides for every street a set of GPS coordinates in form of street segments. The size of the line segment, e.g., the distance between the GPS coordinates, depends on the shape and curvature of the street. If the street is long and straight, we typically get a small amount of GPS points with large distance between them and vice versa if the street is curvy, we get a lot of small segments that describe the physical shape of this street. For every segment, we compute the heading of this street with respect to the North, since this is one of the parameters from Google Street View API. The heading basically defines which way the car is facing the street. Since the road segments can be quite long, we interpolate the points in between the segment endpoints to get more locations to query a panorama image from, see Figure 7.1 for details. Having GPS coordinates with associated headings for every street allows us to directly query images into sequences.

Performing visual place recognition against Street View imagery imposes several further challenges. In addition, to being collected at different point in time,

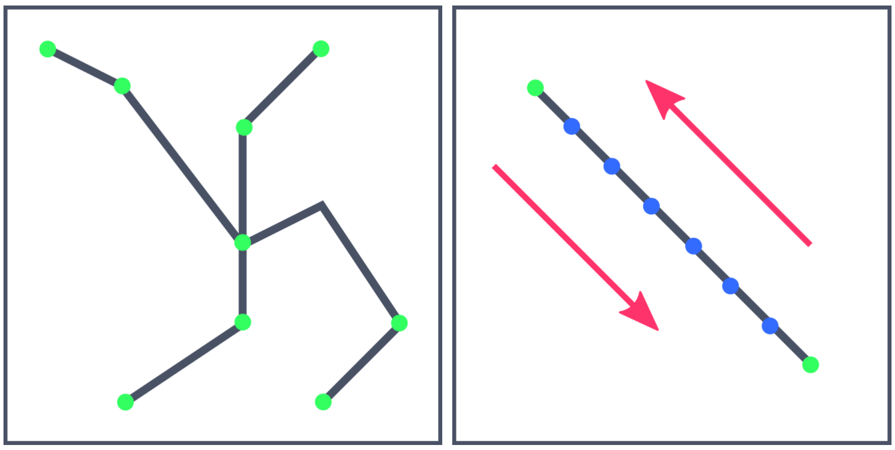


Figure 7.1: When extracting the street information from OSM, we get individual GPS points (green) that describe the road curvature as depicted in the left figure. For every segment, we then increase the number of points to query (blue) by interpolating between the boundary points (green) and estimate the heading (pink arrow) of the street with respect to the North.

with respect to query sequence as well as within the panoramas sequences itself, the frame rate of the panoramas is not constant. There are parts of a street where the density of panorama images is higher, which gives better place coverage in comparison to the places where the density is lower. Furthermore, the viewpoint change can get severe, firstly because the camera on the Google car was mounted on the poll on the rooftop, whereas the camera in our experiments is mounted inside of the car. Secondly, it is not guaranteed that the cars have taken the same lanes or have changed lanes. This becomes particularly challenging for carrying out recognition tasks in the cities with wide streets (6 lanes), since the same place may look substantially different from different sides of the street.

7.2 Extracting streets from OpenStreetMap

To organize the images into the sequences, we use the information from OpenStreetMap about the GPS coordinates that define every individual street. Since OpenStreetMap is an open source community driven project, everyone can add information to it and it is hard to enforce particular rules to be followed for the data collection. Due to this fact, it happens that the streets are not uniquely and completely defined. Typically, a street is defined within an xml file as a node that has a particular property, e.g., a tag “highway”. However, there are instances when a street with the same name is stored in multiple nodes. As a result of that, we obtain from the API way more image sequences than there is exists streets and we end up having a lot of very small image sequences. This becomes critical for our place recognition approach, since the search needs to frequently change between the reference sequences. The change between the sequences is

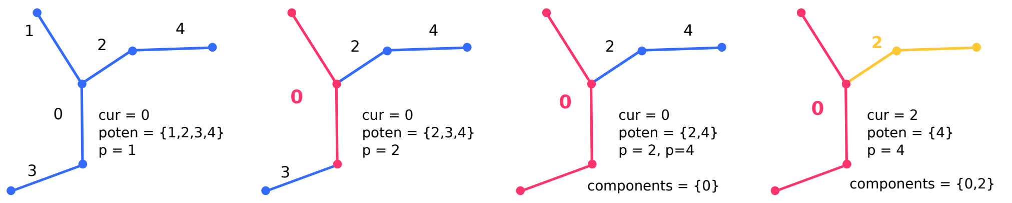


Figure 7.2: From left to right. Steps to combine individual street segments. In this case, due to geometrical structure of the street, there is no unique way to combine segments in one street. So the merging algorithm return two components 0 and 2 representing 3 and 2 segments respectively.

typically triggered by a relocalization, hence hashing action. We showed in the previous chapter that a relocalization action takes more time than a regular graph expansion and thus should be used in case the robot is actually lost, whereas the search that follows an image sequence is preferable and faster. In this section, we describe how to overcome the problem in situations when a street gets represented by multiple image sequences. We propose an algorithm that merges different street segments and allows for computing a smaller number of longer image sequences.

To merge street segments, we first parse the xml file and search for all the nodes that represent a particular street. We store the respective segment coordinates in a list and afterwards work with the list of segments. To make the explanation more intuitive, let us consider the small example shown in Figure 7.2. In this example, we have a street of a certain shape that is represented by 5 segments, where $\{0...4\}$ are the segments IDs. Our segment matching algorithm iteratively goes over the segments and checks if a pair of segments can be merged or connected. We can make the decision on whether two segments can be merged by checking the proximity of GPS coordinates of their boundary points. Two segments can be merged if a starting coordinate of a one segment is located in a close proximity to the end coordinate of the second segments or another way around, if the starting coordinate of second segment is located in close proximity to the end point of first segment. In our example in Figure 7.2, segment 1 can be merged with segment 2 or 0, but not with segment 3. The full algorithm is summarized in the Algorithm 2. To provide a more intuitive explanation, we describe the flow of our algorithm given a small example. Given a street defined by segments in Figure 7.2, the algorithm starts with a segment with ID `cur` $= 0$ and checks if other segments stored in a set `poten` can be merged with current segment, see line 12. On first iteration, segment 1 gets connected to segment 0. This changes the boundary points for the merged segment. Now we have a longer segment with starting point from segment 1 and ending point from segment 0. The new longer segment keeps the id of the current segment 0. On the next iteration, the segment 0 is considered to be matched to segment 2 and since neither start nor

Algorithm 2 Merging street segments(`lines`)

```
1: lines // list of street segments
2: $n$ = len(lines) // number of segments
3: if $n$ == 1 then
4:     // only one segment
5:     return lines
6: cur = 0 // id of segment under consideration
7: poten = {1:n} // set of potential segment ids to match
8: components = []
9: process = True
10: while process do
11:     for p in poten do
12:         if canMerge(lines[cur], lines[p]) then
13:             merge p to cur
14:             poten.remove(p)
15:     if len(poten) == 0 then
16:         components.add(lines[cur])
17:         process = False
18:         break
19:     // cur segment cannot be merged to anything
20:     components.add(lines[cur])
21:     cur = poten[0]
22:     poten.pop(0)
23: return components
```

end lies close to the boundaries of segment 0, the segments can not be merged. The algorithm proceeds with merging segment 3 to segment 0. Afterwards, since the `poten` set is not empty, new subroad consisting of segment 2 and 4 will be created, see line 20. In the end, our matching algorithm will either construct one street or at least reduce the number of segments if unique streets can not be constructed based on street topology.

Figure 7.3 depicts a small part of the city where several long streets were stored correctly. i.e., as one street segment. However, street 1 and 2 are wrongly subdivided in multiple small segments. This generates for our map obtaining approach a lot of small image sequences that have one or two images. By applying the above proposed segment merging strategy, we obtain the right image in Figure 7.3, where the street segments were merged into longer streets, particularly street 1 and 2. The colors of the streets are picked at random and do not correspond in between the images. To further confirm the benefits of our segment

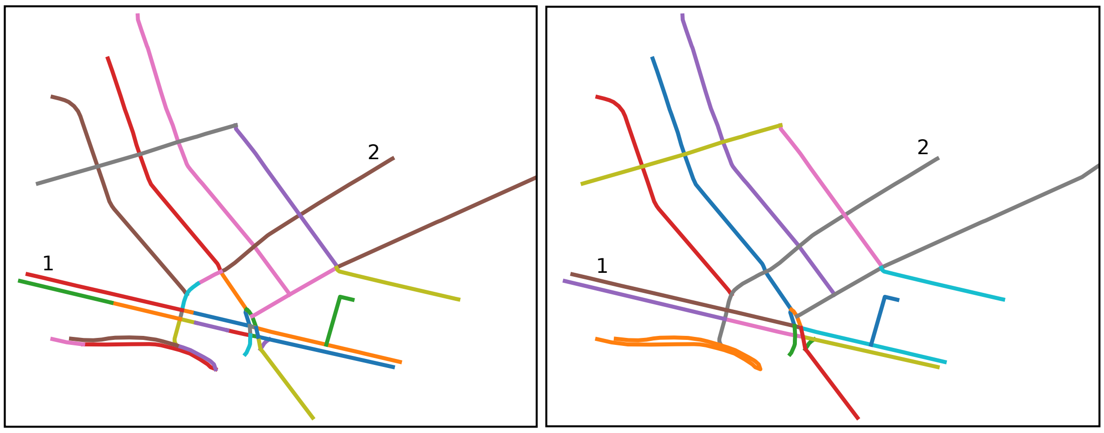


Figure 7.3: Left: Street segments from OpenStreetMap data. Right: Merged street segments. The colors are randomly selected for visualization and do not correspond between the images.

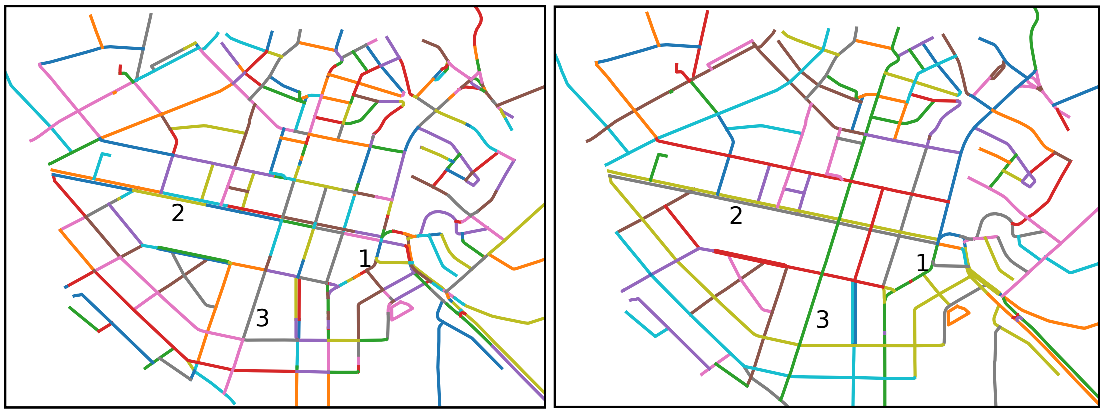


Figure 7.4: Left: Street segments from OpenStreetMap data. Right: Merged street segments. The colors are randomly selected for visualization and do not correspond between the images. Our algorithm reduces the number of street segments which is particularly visible for streets 1, 2, 3.

merging scheme, we selected a bigger area with streets with big variety of shapes. As can be seen from Figure 7.4 the merged streets (right) are more homogeneous in color than the street segments obtained originally from OpenStreetMap (left). This indicates that we are able to reduce the number of street segments and thus make the reference image sequences longer, which is beneficial for our visual place recognition approach. The effect is particularly visible for the street 1, where we reduce from 14 to 3 street segments. Similar result can be seen for street 2 and 3.

7.3 Experimental evaluations

Our experimental evaluations are designed to show that publicly available information can be successfully used with our multi-sequence visual place recognition approach. We demonstrate this with several experiments.

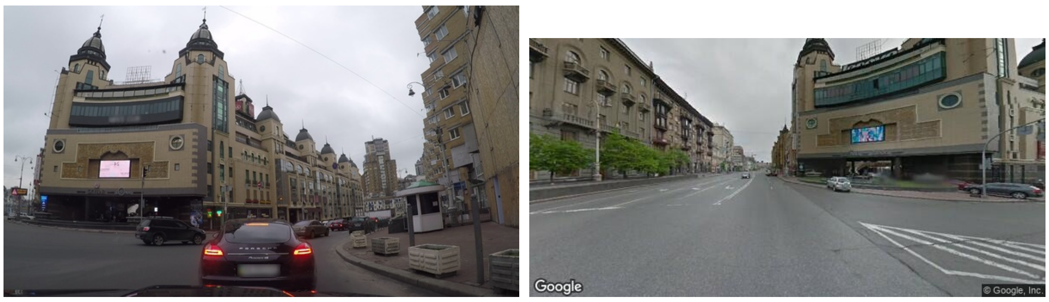


Figure 7.5: Experiment 1. An example of a matching image pair from the car perspective (left) and from Google Street View (right). The images are subject to strong view point change as well as seasonal changes.

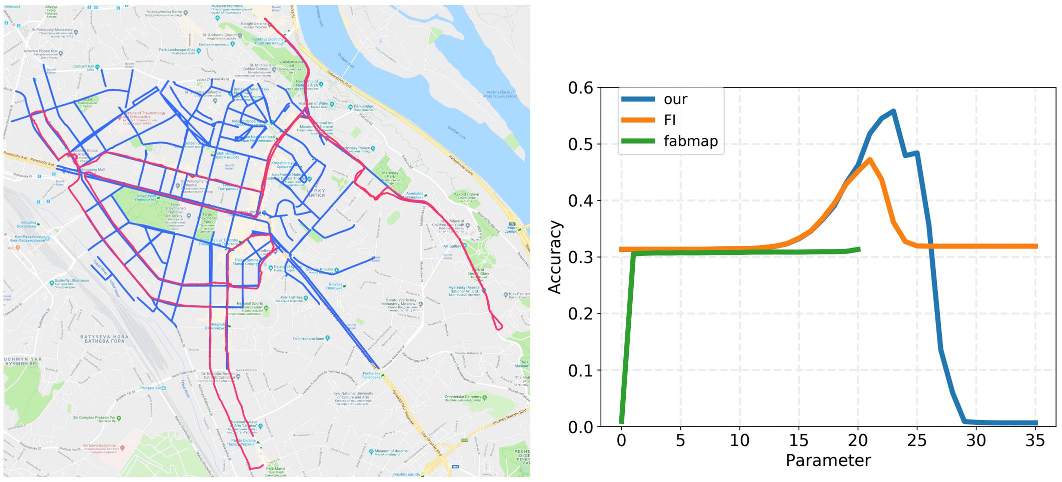


Figure 7.6: Experiment 1. Left: City streets for which panorama images were extracted (blue), query trajectory driven by a car (pink). Right: Corresponding accuracy plot.

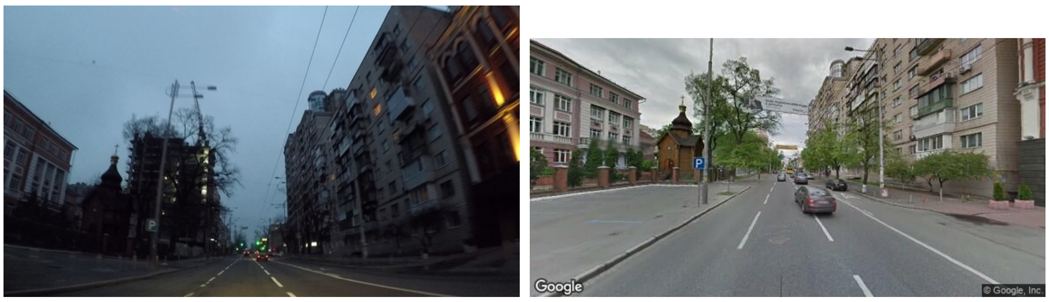


Figure 7.7: Experiment 2. A matching example of a particularly challenging sequences to match. The visual appearance of the image from the car (left) is substantially different to the one from the street View (right).

The first experiment is designed to show that the ideas of place recognition against multiple trajectories can be successfully applied to relocalize against Google Street View. As was noted before, place recognition against street view is more challenging than recognition against multiple sequences due to irregular frame rate of panorama images, partially drastic viewpoint changes on top of environmental visual appearance changes. In this experiment, we show that despite the additional challenges our method is able to achieve accuracy from 34% to 55% on various sequences. The first query image sequence consists of 3,800 images and was collected inside the car in Kyiv city center, see Figure 7.6 (left) for visualizations of the trajectories. The total amount of extracted panorama images for the reference trajectories is 10,272 and they form 247 streets or reference sequences. Figure 7.5 shows a typical matching example successfully found by our approach from query and Street View. As to qualitative evaluations, Figure 7.6 (right) shows that taking sequence information into account (our approach) outperforms the pure FI search and results at best with 58% accuracy versus 48% for informed search. The left picture of the same figure shows the overlay of the trajectories over Google Street Maps. Blue refers to the region extracted from Street View, whereas pink depicts the trajectory of the query image sequence.

The second experiment consists of a query trajectory with 3107 images that was recorded in the early till late winter evening, whereas the images from Street View are typically recorded throughout the day. We use the same reference dataset as in the previous experiment. The visual changes goes as extreme as the one depicted in Figure 7.7. Let us look more closely what makes this pair of images particularly challenging to match even for people. Firstly, the tilt between the camera positions changes the way parts of the environment are getting projected into the images. For example, for the car image (left) the road occupies a relatively small part of the image whereas in the street view image (right), the road covers almost the third of the image. This results in the fact that parts of the same object are clearly visible in one image and completely hidden in

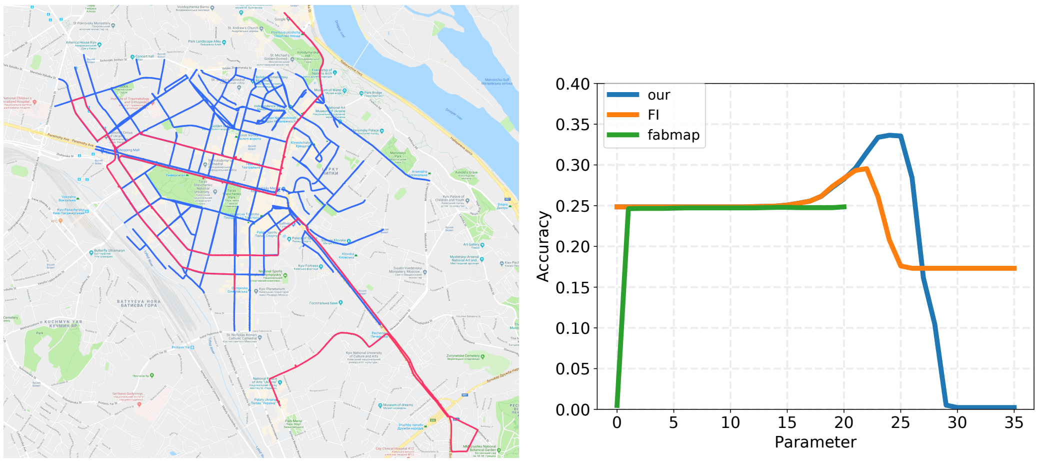


Figure 7.8: Experiment 2. Right: accuracy evaluation. Out approach gives higher accuracy than fully informed search (FI) and FABMAP. Left: The query trajectory depicted in pink, with a particular challenge of visual appearance changes within the query trajectory from evening to night.

the other one. The same holds for the buildings where we can clearly infer the ninefloor building in the car image but only five floors are visible in the Street View one. Additionally, to seeing parts of the environment that are not exhibit in both images, the day- night illumination difference plays a role. The natural and artificial lights highlight different textures and edges on the buildings, which are typical robust features for matching. Furthermore, we can spot the effect of seasonal changes, where the trees in the car images lack foliage and thus reveal structured parts of the environment not visible in the street view image. However, these challenges fade away when considering the fact that not just the visual appearance of the environment has changed but also the the structure of the environment itself. A ten floor building is presented in one image and is missing in another one.

All these factors make the image matching process hard, but unfortunately are part of our every day life. As can be seen from Figure 7.8 (right), our approach outperforms the FI search and FABMAP with maximum achieved accuracy of 34%. The overall performance of all the algorithms though is lower than in previous experiment due to combination of seasonal, view point, and stronger natural illumination changes.

The last experiment is designed to illustrate that our approach can recognize places from a random street drive footage taken from YouTube. The particular challenge of this experiment lies in the fact that both query and reference sequences were collected with different cameras as well as a different and unknown to us positioning setup. Figure 7.9 shows an example of a matching pair that was successfully found by our approach. More examples of successfully found pairs are depicted in Figure 7.10 and Figure 7.11. This experiment and the video

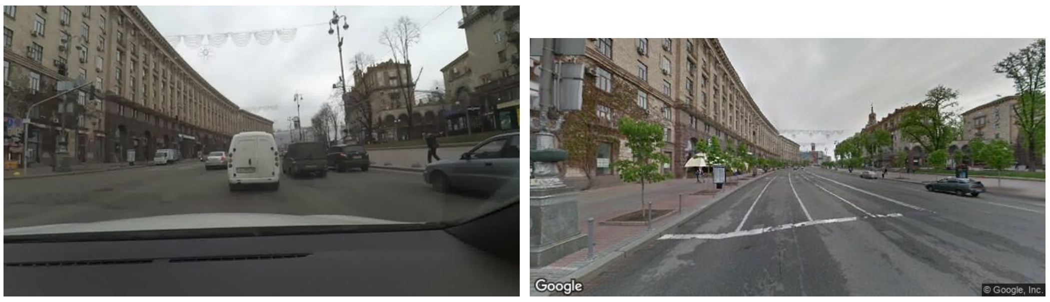


Figure 7.9: Experiment 3. A matching image pair found by our approach from a YouTube video (left) and from Google Street View (right).

suggest that our algorithm is able to recognize places using images only from unknown camera setups, although we do not provide the accuracy evaluations for this experiment due to the lack of exact positioning information from the YouTube video.

7.4 Conclusion

In this chapter, we have shown how our visual place recognition approach against sequence maps can be deployed when having publicly available information, like Google Street View or other image map sources. This is a step forward toward deploying vehicles in new places without place-dependent setup or installation procedures. Our system is also able to relocalize simultaneously in manually collected sequences as well as image sequences from other sources. Moreover, we have shown that our algorithm does not require knowledge about camera calibration parameters and is able to robustly recognize the place from image footages taken from a random video.

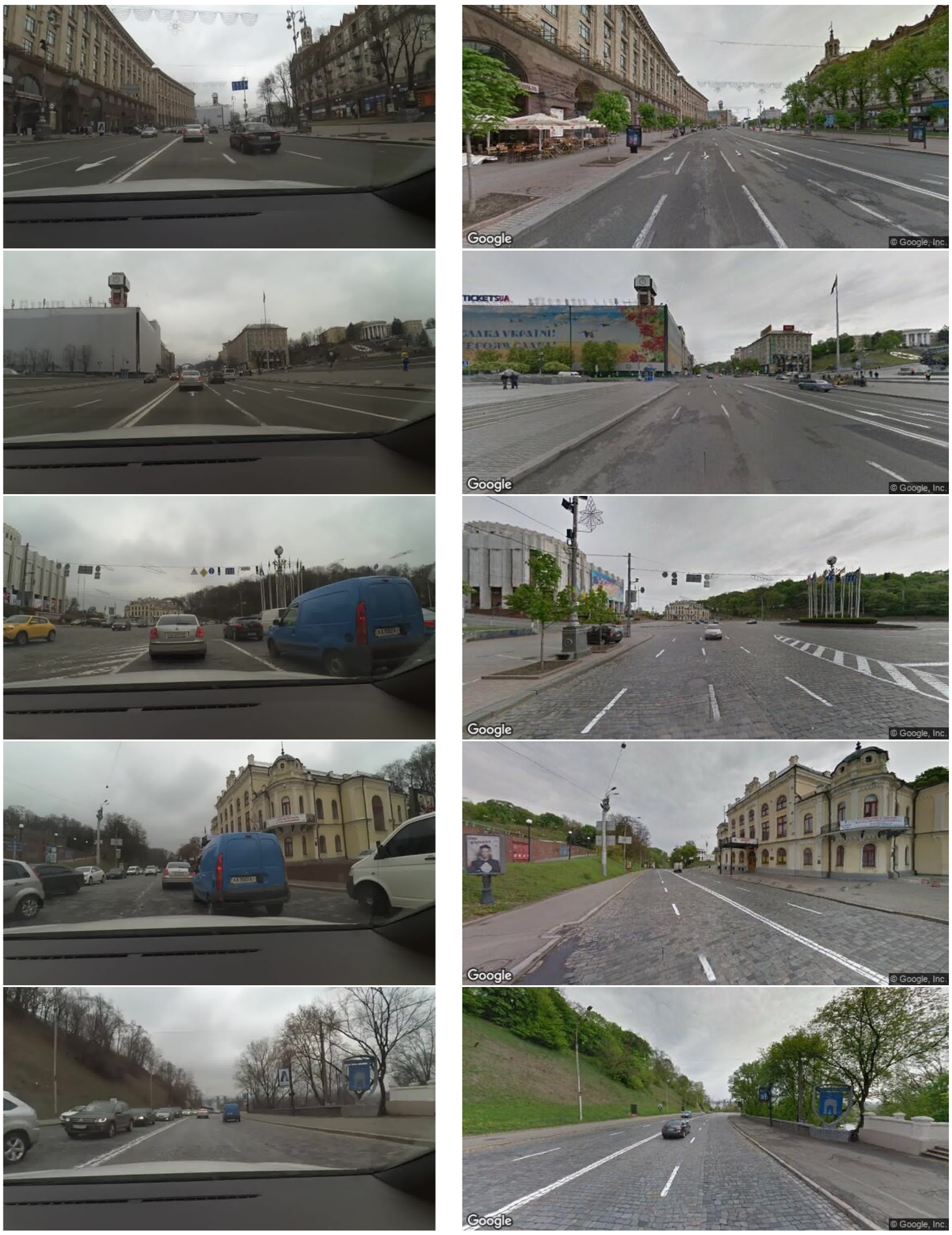


Figure 7.10: Example matches found by our visual place recognition approach. Left column: images from a random YouTube video; right column: images from Google Street View.

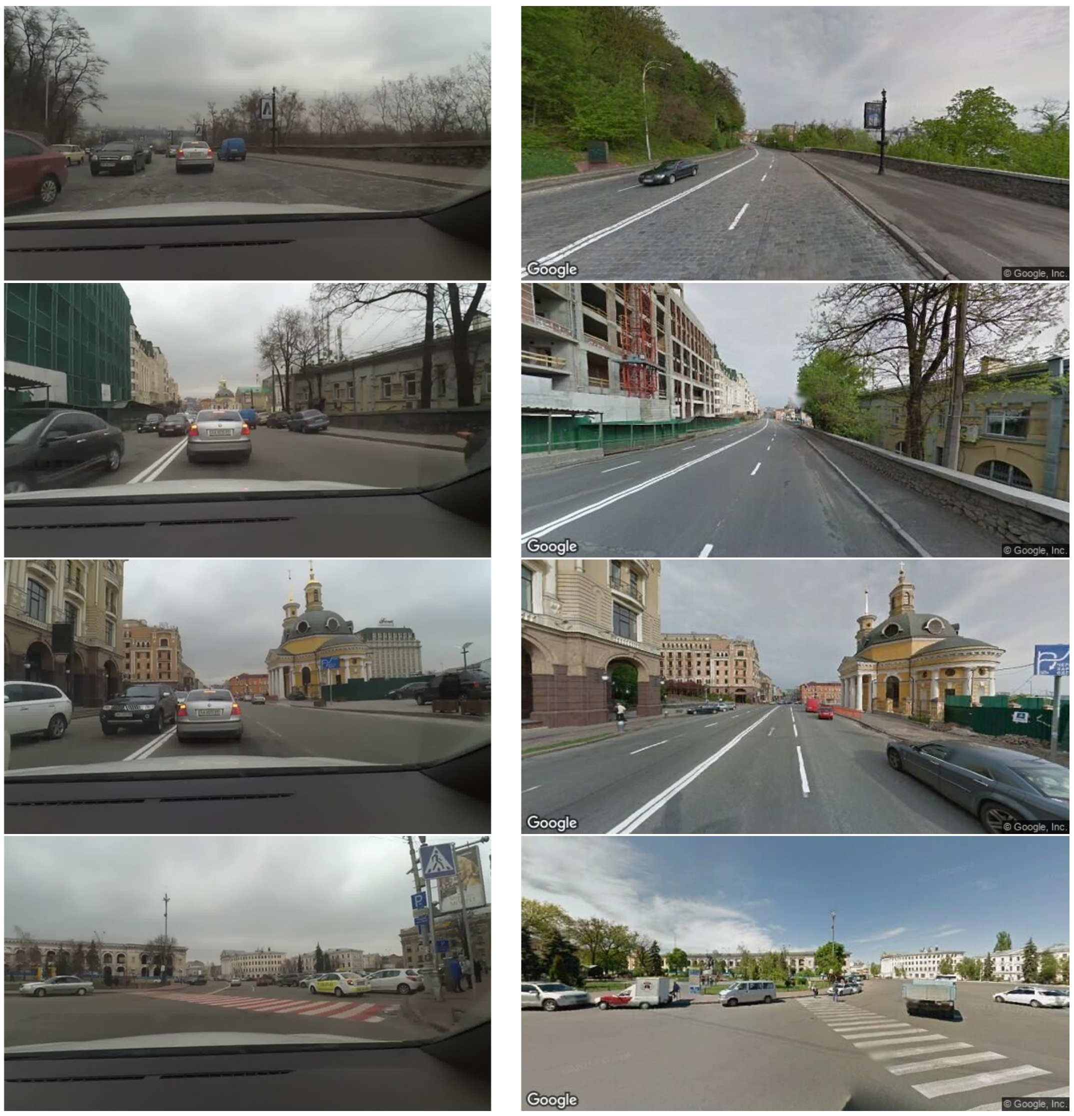


Figure 7.11: More example matches found by our visual place recognition approach. Left column: images from a random YouTube video; right column: images from Google Street View.

Chapter 8

Improving robot localization using publicly available maps

Maps are needed for a wide range of applications. In the context of mobile robotics, the map learning problem under uncertainty is often referred to as the simultaneous localization and mapping or SLAM problem. These maps are often used for path planning, navigation or localization. In previous chapter, we showed how to use publicly available data to perform robust place recognition and avoid costly mapping phase. In this chapter, we show another way how publicly available maps can contribute to not just spare computationally expensive mapping phase, but also enhance the localization capabilities of the robot for graph-based SLAM approaches. Additionally, we select a 2D laser scanner as the robot's main sensor for localization and mapping, to provide means for robots that are not equipped with cameras, as well as using OpenStreetMap as a source for publicly available information. We achieve localization on OpenStreetMap by relating the information about buildings with the perceptions of the robot and generate constraints for the pose-graph-based formulation of the SLAM problem. In addition to that, we present a way to select target locations for the robot so that by going there, the robot can expect to reduce it's own pose uncertainty. This localizability information is generated directly from OpenStreetMap data to support active localization. We implemented and evaluated our approach using real world data taken in urban environments. Our experiments suggest that we are able to relate the newly built maps with information from OpenStreetMap with the laser range finder data from the robot and in this way improve the map quality. The extension to graph-based SLAM provides better aligned maps and adds only a marginal computational overhead. Furthermore, we illustrate that the localizability information is useful to evaluate the ability to localize the robot given a trajectory.

Most robot navigation systems require a map of the environment as well as

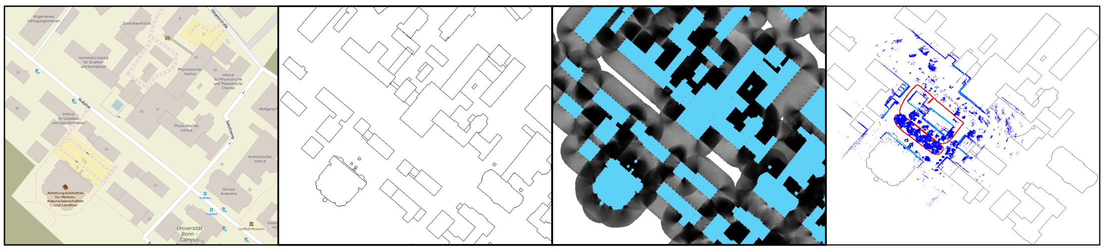


Figure 8.1: From left to right: Screenshot from OpenStreetMap; map that we render for alignment; computed localizability map; resulting robot map.

the current pose of the vehicle in this map. Thus, having an online building procedure for a consistent map of the robot's surroundings is one of the essential prerequisites for the reliable autonomous robot operation. As errors in the robot's pose accumulate over time, building large scale maps from odometry and laser range information often leads to a drift in the trajectory estimate. GPS information can be used to compensate for that drift. This works well if a sufficient number of satellites is visible. In urban environments, however, narrow streets, high buildings and trees can hinder the capabilities of the receiver and disturb the GPS signals. This may result in a poor positioning performance. In addition to that, performing loop closures reduces the drift, but it forces the robot to re-visit places in the environment, e.g., re-entering the same street.

Recently, exploiting alternative information sources for enhancing outdoor mapping are gaining attention in the various communities, such as coupling the information from publicly available maps, like aerial photographs [77] or OpenStreetMap (OSM) data with standard localization [64] or SLAM approaches. We see exploiting such background information as orthogonal to using GPS information as aligning sensor observations with publicly available maps works typically well in GPS-denied environments. The ideas of incorporating additional prior information about the environment are also explored in context of unmanned aerial vehicles, see [51] and [135].

The main contribution of this chapter is a novel approach to align the robot's trajectory to OpenStreetMap data and to estimate vantage points that are likely to reduce the pose uncertainty of the robot. We integrate the ability to relate the obtained laser range measurements with the OSM data into a SLAM framework using pose graphs. This approach uses buildings as landmarks for SLAM to align the robots trajectory with existing maps. Not all places in an environment allow for matching the onboard perceptions with the OSM data. In some areas, the building structure is not distinctive enough for the robot to find an alignment. For example the result of the localization is ambiguous when the robot is located in a corridor-like environment, e.g., between two long buildings, see Figure 8.2 (left). In this case, even obtaining ideal measurements, e.g., endpoints located on the

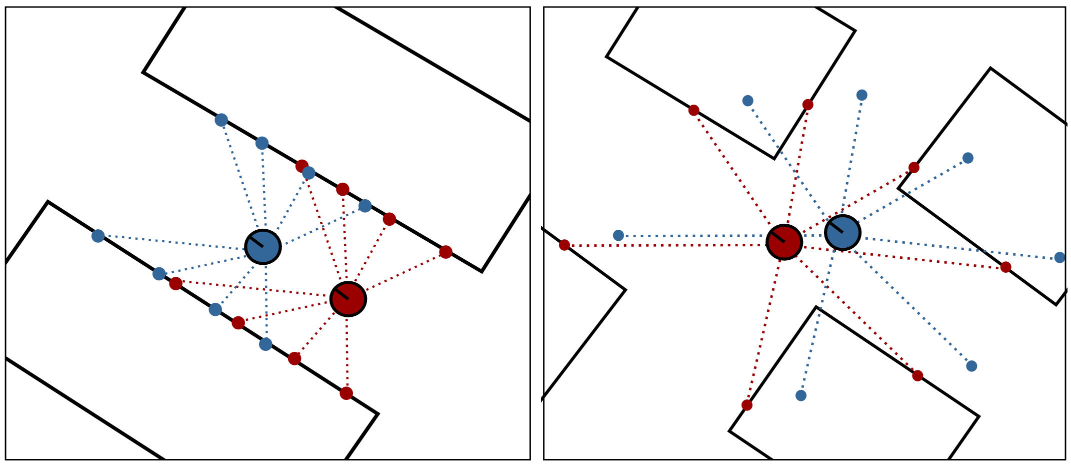


Figure 8.2: Uninformative (left) vs. informative (right) pose. In the left image the pose of the robot (red) is not informative, because by applying the small transformation to the robot's pose (blue) the virtual measurement explains the surrounding as well as from the previous pose, whereas in the second image the pose of the robot (red) is informative, since the transformations of the pose (blue) decreases the likelihood of measurement.

walls, will not improve the localization, since the nearby poses explain the environment as well as the actual location. To perform the alignment in a better way, we would like to avoid navigating through the ambiguous regions and prefer the regions with more distinctive structure like the one in Figure 8.2 (right), where the ideal measurement originated from the query pose has a low likelihood of being observed from nearby poses. Thus, we also propose a technique that turns the information from publicly available maps into so-called localizability maps, i.e., maps that indicate how well the robot is expected to establish the data associations between its own sensor readings and the OSM information and thus can localize itself.

8.1 Graph-based SLAM exploiting existing maps as background knowledge

We start by introducing a general formulation of graph-based SLAM as defined by Grisetti *et al.* [57]. In a pose graph, every node corresponds to the a robot pose. Consequent nodes are connected through the edges that model spatial constraints and arise from odometry measurements. Non consecutive nodes are typically connected through the edges that arise from multiple observations from the same place in the environment.

Considering X as the set of robot poses where x_i describes a pose of the node i. If the nodes x_i and x_j observe the same part of the environment, the measurement z_{ij} can be obtained by performing a scan aligning between the corresponding sensor measurements obtained from node x_i and x_j. The expected measurement $f(x_i, x_j)$ is computed from the relative position of the node x_i and node x_j and is expressed in the frame of reference on the node x_i. Then,the error $e_{ij}(x_i, x_j)$ depends on the displacement between the expected $\hat{z}_{ij}$ and real measurement z_{ij}:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij} \tag{8.1}$$

After defining the individual errors for the edges of the pose graph, our goal is to find a configuration of nodes X that minimize the total error of system defined as:

$$F(X) = \sum_{(i,j)\in C} e_{ij}^T \Lambda_{ij} e_{ij}, \tag{8.2}$$

where Λ_{ij} is the information matrix of the constraint.

The optimization step in graph-based SLAM systems aims at finding the configuration of the nodes that minimizes the error induced by observations. In general the pose of the robot consists of the location of the robot and its orientation. In this chapter, the pose representation of the robot's pose is a 3 dimensional vector, consisting of two translational and one rotational components. This yields a state vector $X = (x_1, \ldots, x_n)^\top$ where x_i is the pose of node i. The error function $e_{ij}(X)$ for a single constraint between the nodes i and j is often the difference between an expected measurement $f(x_i, x_j)$ (relative pose between nodes i and j) and the obtained measurement z_{ij}:

$$e_{ij}(X) \;=\; e_{ij}(x_i, x_j) \;=\; f(x_i, x_j) - z_{ij}. \tag{8.3}$$

Note that alternative representations can be used to avoid problems resulting from singularities in the angular components, see [58] for details. As the error functions are typically non-linear, we linearize $e_{ij}(X)$ around the current best estimate

$$e_{ij}(X + \Delta X) \;\simeq\; e_{ij}(X) + J_{ij}\Delta X. \tag{8.4}$$

Here, J_{ij} is the Jacobian of the non-linear error function computed in the current state. Thus, the resulting minimization problem turns into

$$X^* \;=\; \operatorname*{argmin}_X \sum_{ij} e_{ij}(X)^\top \Lambda_{ij} e_{ij}(X), \tag{8.5}$$

where Λ_{ij} is the information matrix, also referred to as weight matrix associated to a constraint. Up to this point, this is the standard formulation of pose-graph SLAM and Equation (8.5) can be solved as a least squares problem. To reduce the impact of the outliers, we use a robust kernel function, namely dynamic covariance scaling as proposed by Agarwal *et al.* [2]. This approach rescales the error function $e_{ij}(X)$ depending on its magnitude to reweight the impact of potential outliers. This is, up to a parameter, equivalent to using a Geman-McClure kernel. This scheme for down weighting the impact of outliers is also often referred to as robust estimation, see [45] for an overview.

8.2 Error function exploiting existing maps

In order to incorporate additional knowledge into the optimization process and relate the pose-graph to existing data, we extend the error function to

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{ij} e_{ij}(X)^\top \Lambda_{ij} e_{ij}(X) + F^{map}(X), \tag{8.6}$$

where $F^{map}(X)$ is the error introduced by the mismatch between the robot's observation and the map information. Analogous to pose-pose constraints, we split up the component $F^{map}(X)$ into individual constraints between robot poses and the OpenStreetMap information:

$$F^{map}(X) \quad = \quad \sum_i e_i^{map}(X)^\top \Lambda_i e_i^{map}(X). \tag{8.7}$$

The key elements in Equation (8.7) are the error function $e_i^{map}(X)$ and corresponding information or weight matrix Λ_i. The remainder of this section describes how to define such an error function and respective information matrix. Intuitively, the error function adds an additional constraint to the graph that anchors a pose of the robot to a specific location in the map. The key challenge here is to make the correct data association between the map and the robot's own sensor readings, obtained from the pose stored in the node of the pose-graph. Once this data association is solved and the correct coordinate transformations between the robot's poses and the map are computed, least squares error minimization will provide us with the global alignment.

To make the data association between the map and the robot's poses, we use the building information in the map and the data from a 2D or 3D laser range finder installed on the robot. When aligning laser range data with the building information from OSM, a central challenge is that the laser scanner observes a large number of objects in the scene that are not stored in the map. Examples for such objects, which are not present in the publicly available map, are trees, cars, or pedestrians. In contrast to most other approaches that perform localization on OSM data, we choose buildings as our features to make the data association and to compute the alignment. The most approaches rely on the road network to localize the robot [111, 64]. This is perfectly fine for cars or robots moving on the roads, but often limits the application to robots that operate on sidewalks,

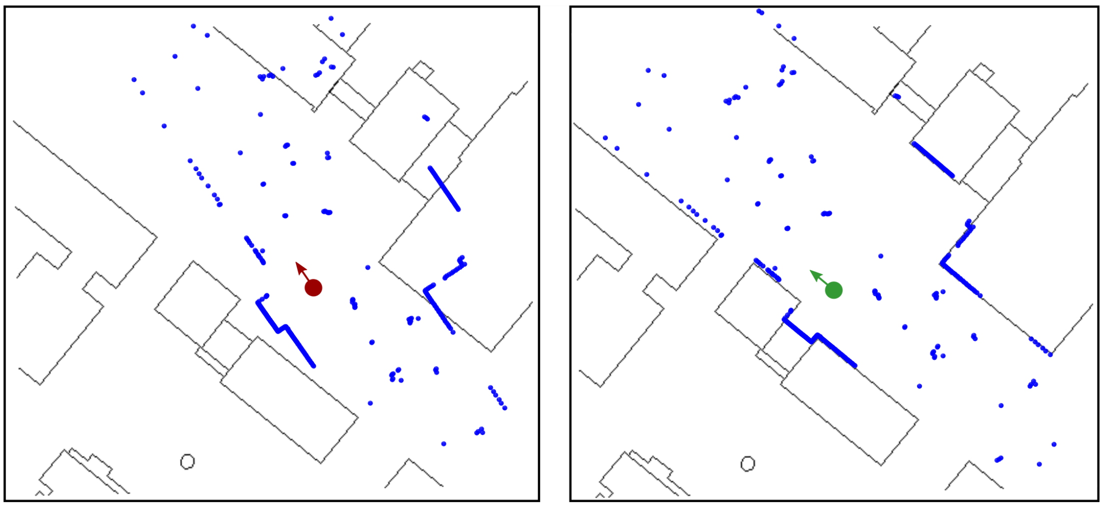


Figure 8.3: An example correction of the robot's pose based on the aligning of the scan (blue) to the buildings in the map (black); left image shows the robot pose before correction and right afterwards.

foot paths, or in pedestrian zones and do not follow the road network, as no good data association between the trajectory and the road network can be found in such cases.

8.3 Error function exploiting building information for robots equipped with laser range scanners

In our work, we use the information about the buildings' geo-locations rather than a road network as for example done by Ruchti *et al.* [111] to enable the robot to take paths independently from the road network. We obtain the building information directly from OpenStreetMap, which can be downloaded in form of an XML-file. Inside this file, the individual buildings are stored as separate nodes. Each node is a closed polygon describing the geo-referenced walls of the building, which directly yields a map of lines that shows the walls of the buildings in the environment, for example see the black polygons in Figure 8.3. A laser scanner typically provides the scan of the environment covering a large number of objects that are not buildings and does so at a comparably high level of detail. This may hinder the matching procedure to make the correct data association between map and laser scan. Therefore, we filter the range scans so that most of the non-building objects are removed. We investigated several techniques and in the end opted for an unsupervised approach that performs filtering based on line extraction. It does not require manually labeled training data and can be executed efficiently. We employ the Douglas-Peuker algorithm [38] for converting the raw 2D range scan into a polyline. We convert the polyline into a set of potentially disconnected lines based on two parameters: the length of a line and the number of laser end points assigned to each line. The problem of detecting building structures has also been investigated in the context of reconstructing the 3D structure. For example, Fischer *et al.* [41] use generic models to extract 3D buildings from the aerial images. Huber *et al.* [68] fuse the LIDAR data with aerial imagery and apply polyhedral models to reconstruct the buildings. We, on the other hand, detect buildings in the single 2D laser scan, not taking into account the information from the maps.

Since our aim is to incorporate the knowledge about the environment from the map into the graph optimization procedure to refine the robot's trajectory, the error function for this constraint should reflect the misalignment between the current robot's pose and the map. Intuitively, the bigger is the misalignment between the scan and the buildings in the map, the bigger the error should be. To estimate the (mis-)alignment, we use the Iterative Closest Point (ICP) algorithm [23] to match the current laser scan and the map of building. For finding the correspondences in ICP, standard nearest neighbor data association is applied.

More precisely, the error term $e_i^{map}(x_i)$ is defined based on the difference between the current robot's pose x_i and the pose $\hat{x}_i$, computed by aligning the scan in the building map. The 2D state of the robot x_i consists of translational t_i and rotational θ_i components, i.e., forms an element in $SE(2)$, the special Euclidean group for two dimensions. The same holds for the state $\hat{x}_i$. Thus, the error function and its Jacobian turns into:

$$e_i^{map}(x_i) = \begin{pmatrix} \hat{R}_i^\top (t_i - \hat{t}_i) \\ \theta_i - \hat{\theta}_i \end{pmatrix} \quad \text{and} \quad J_i = \frac{\partial e^{map}(x_i)}{\partial x_i} = \begin{pmatrix} \hat{R}_i^\top & 0 \\ 0 & 1 \end{pmatrix}, \tag{8.8}$$

with $\hat{R}_i$ being the standard 2D rotation matrix corresponding to the angle $\hat{\theta}_i$. For the ICP algorithm to operate reliably, we need a good initial guess. In our setup, the initial guess is achieved by either manually specifying the first pose of the robot on the map or by using an initial guess from a consumer GPS with an accuracy of a few meters. The initial guess of all successive poses is then automatically obtained from the odometry constraints of the graph or by incremental scan-to-scan alignment typically used in graph-based SLAM with laser range finders.

Finally, we have to compute the weight matrix Λ_i of a map constraint, which is the inverse of the covariance matrix of the ICP alignment, i.e., $\Lambda_i = (\Sigma_i^{ICP})^{-1}$. We compute the covariance matrix Σ_i^{ICP} from the ICP result by using the Hessian as described by Bengtsson *et al.* [21]. This assumes the error function e^{ICP} used in the ICP algorithm to be quadratic near the optimal solution, i.e.,

$$e^{icp} = \sum_k \|Tp_k - q_k\|^2 \,, \tag{8.9}$$

where p_k is a point from a laser scanner that belongs to the detected buildings and q_k is a corresponding closest point in the buildings taken from the publicly available map. The optimal transformation T that the ICP algorithm reports is found by minimizing the function e^{icp} with the covariance matrix of T as

$$\Sigma_i^{icp} = \text{cov}(T) = 2\sigma^2 \left(\frac{\partial^2}{\partial T^2} e^{icp} \right)^{-1} = 2\sigma^2 H_{\text{icp}}^{-1}, \tag{8.10}$$

where H_{icp} is the Hessian matrix of e^{icp} and σ^2 is the variance factor.

So far, we described how to obtain the error function for 2D range data such as a horizontally mounted 3D range scanner, but it works analoguously on data from a 3D laser scanner. In this work, we use 2D and 3D range data. In case of 3D data such as the one coming from a Velodyne scanner, for every individual scan we construct a 3D point cloud and generate new virtual 2D laser scans given the planar surfaces in the cloud. We basically follow the approach proposed by Wulf *et al.* [145], which is also used by Bogoslavskyi *et al.* [25] and Hentschel *et al.* [65].

8.4 Error minimization

Given the error function e_i^{map} with corresponding information matrix Λ_i^{map} and Jacobian J_i, we use Levenberg-Marquardt optimization to solve the problem given in Equation (8.5). This leads to iteratively solving a linear system of the form

$$(H + \lambda I)\Delta X^* \;\; = \;\; -b, \tag{8.11}$$

with

$$H = \sum_{ij} J_{ij}^\top \Lambda_{ij} J_{ij} + \sum_i J_i^\top \Lambda_i J_i \quad \text{and} \quad b = \sum_{ij} J_{ij}^\top \Lambda_{ij} e_{ij} + \sum_i J_i^\top \Lambda_i e_i^{map}. \tag{8.12}$$

H and b are the key elements and are computed from the linearized error functions and λ is the damping factor used in the Levenberg-Marquardt. The term ΔX^* refers to the increments that are added to the graph configuration in order to minimize our error function in the current iteration. In our implementation, we use the g2o framework by Kümmerle *et al.* [76] to conduct the minimization with dynamic covariance scaling by Agarwal *et al.* [2]. This yields an update of the graph configuration in every iteration of the form:

$$X \;\leftarrow\; X + \Delta X^{*}. \tag{8.13}$$

We do not execute this procedure in a batch fashion but selected the incremental option of the g2o optimizer, which allows us to optimize the trajectory in chunks. In our current implementation, we trigger an update whenever the robot drove for 25 m. This has two advantages. First, the data are available already online during mapping. Second, the correction of the trajectory up to a point in time t_1 will simplify the data association for the ICP step for subsequent matching with $t > t_1$ and, thus, has the potential to provide a better alignment. As a result of that, we obtain an optimized pose-graph that is aligned with the provided map.

8.5 Estimating localizability for actively reducing pose uncertainty

By using publicly available maps such as OpenStreetMap, we are able to better align the robot's trajectory and, hence, the robot's own map to the surrounding environment. This approach, however, relies on the observations that the robot obtains, which in turn depend on the local surroundings of the robot. The ability to match the local perceptions to the OSM data depends on the visibility of buildings and the local geometry or arrangement of the buildings. The goal of this section is to describe an approach to estimate the ability of the robot to align its perception with the OSM data *before* moving there. Thus, we aim at estimating a visibility map from the OSM data and selecting the regions that are expected to provide good vantage points that support the alignment. A measurement will be informative for the robot if (i) the buildings can be detected in the individual scans and (ii) the observed structure allows for reducing robot's pose uncertainty. Thus, we present a method that estimates the regions in the environment, where the informative laser scans are likely to be obtained, given the information from publicly available maps.

To reason about the informativeness of a particular pose on the map, we need to specify the function, which measures the likelihood of obtaining a certain laser scan given a pose. For our sensor model, we assume that individual beams z_j in the laser scan are independent and that the measurement noise for each beam is Gaussian, i.e., $p(z_j) \sim \mathcal{N}(c_j, \sigma_l)$, where c_j are the closest points in the buildings that correspond to the individual laser beam endpoints z_j, for visualization see Figure 8.4 (right).

The estimation begins by rendering the map M, as in Figure 8.1 (middle), from the publicly available OSM xml file. Here we assume a simplistic representation of the world, where black pixels correspond to the boundaries of the existing

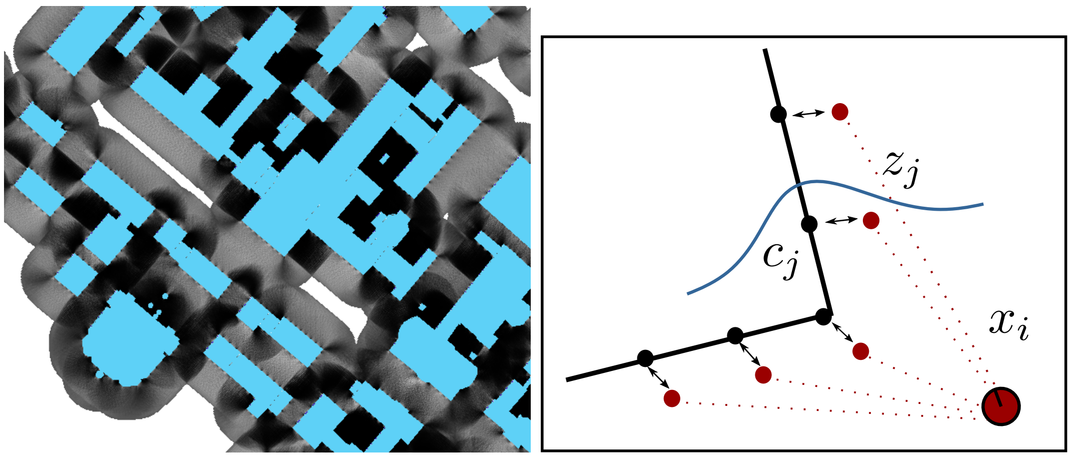


Figure 8.4: Left: An example of a localizability map. The darker the regions the bigger the likelihood to obtain in informative measurement. The buildings are marked in blue. Right: The total error of transforming virtual scan w.r.t the pose x_i depends on the distances from the measurement endpoints z_j to corresponding closest points in the buildings c_j.

buildings and every white pixel outside the buildings corresponds to the potential robot's pose. These assumptions are easily violated in real world, since not every pixel outside the building can be reached by a robot due to fences, parked cars or other unmarked structures. The assumptions are, however, sufficient for our purposes, i.e., for coarsely estimating the regions for the informative measurements. Reachability can later on be easily verified using Dijkstra algorithm or a path planner such as A^*.

Our goal is to estimate for every potential robot pose the associated uncertainty of the pose estimate based on the visibility of the building structures in the scene. We start with simulating an ideal (virtual) laser scan at every potential pose x_i. This measurement assumes that only the buildings from OSM data exist in the environment. This simulated scan is generated by performing a ray-cast operation in the maps from OSM. We then estimate how well this ideal measurement matches to the OSM maps under pose uncertainty. We estimate this by applying small perturbations to the robot's pose in x, y and θ. By doing so, we form a set S of potentially similar pose configurations x_j and estimate the corresponding errors which arise by comparing the a virtual scan to the map in the new robot configuration. We estimate an error for a pose configuration as a sum of squared errors $e(z_j)$ of individual beams of the scan. We then perform an approximation taking into account our assumption about the probability distribution of an individual beam. We approximate the probability of taking a virtual measurement at the pose x_j as $p(x_j) \sim \exp\left(-\frac{\sum e(z_j)}{2N\sigma_l^2}\right)$, where N is size of the scan and $e(z_j)$ is a squared distance between measurement endpoint z_j and closest building in the map. In other words, by applying these actions, we approximate the unknown probability distribution about the robot's pose. Having the samples

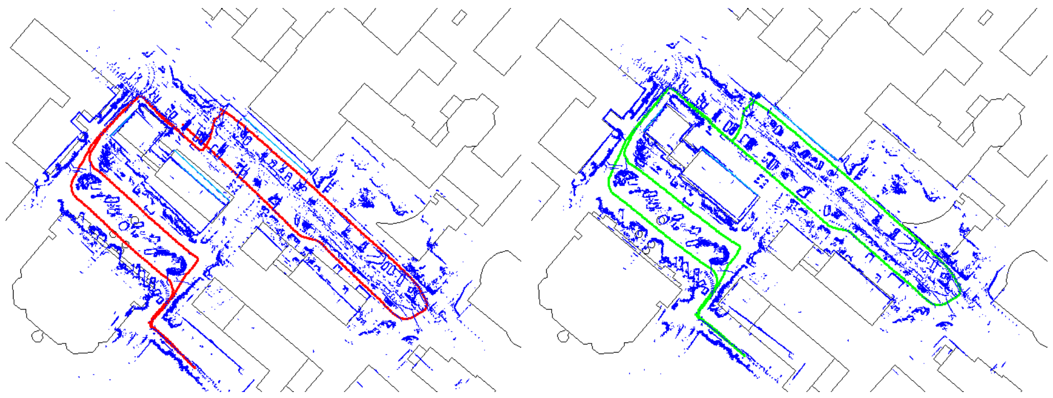


Figure 8.5: Example of aligning the robot's trajectory with the buildings on the map and as a result of it improved loop closure, which also leads to more consistent robot map. The OSM map is rendered with 0.3 meters per pixel resolution.

from a probability distribution, we obtain an estimated covariance matrix, i.e., the uncertainty of the pose, as follows:

$$\mathrm{cov}(x_i) = \sum_{x_j \in S} p(x_j)\,(x_j - q_{x_i})(x_j - q_{x_i})^\top \tag{8.14}$$

where q_{x_i} corresponds to the coordinates of the query pose.

To be able to reason about the informativeness of the different regions in a more quantitative way, we compute the Eigenvalues/Eigenvectors of the respective covariance matrices and therefore obtain the information in which direction we are most uncertain about the pose. Afterwards, we update our localizability maps with the biggest Eigenvalue for every pose. This ensures that we take into account the value of the biggest uncertainty across all dimensions. As a result, the regions in the map with smaller values correspond to the places where the largest uncertainty over individual dimensions is smaller in comparison to other regions or, in other words, the regions where the informative measurements are more likely, see Figure 8.4 for visualization.

8.6 Experiments

The evaluation is designed to illustrate the performance of our approach and to support the following claims. These key claims are that (i) we improve the map alignment with our approach, (ii) can handle situations in which the map data are partially outdated, for example if buildings have been demolished or new buildings have been built, (iii) all operations yield only a small computational overhead compared to a standard graph-based SLAM system, and (iv) the localizability maps provide information about the ability of the platform to localize along a given trajectory. We performed our experiments in outdoor urban environments

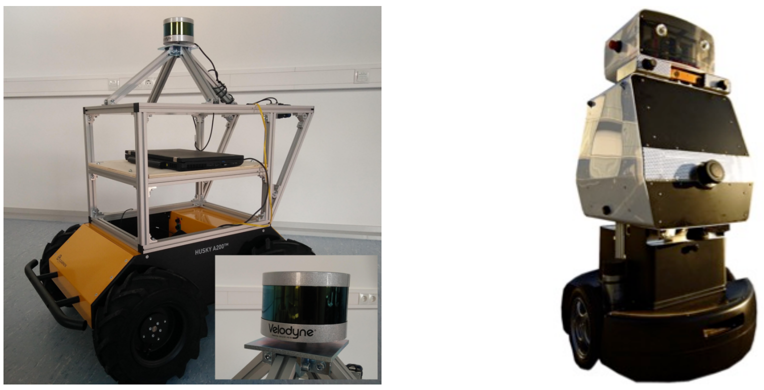


Figure 8.6: Robots used in our experiments: robot equipped with Velodyne VLP-16 laser scanner mounted parallel to the ground (left) and a Velodyne HDL-32E mounted on the head of the Obelix robot from the AIS lab of the University of Freiburg (right).

using the odometry from two robots, Husky A200 and Obelix, both are depicted in Figure 8.6 and are equipped with Velodyne VLP-16 and Velodyne HDL-32E laser scanners respectively. For detecting lines in laser scans, we used Douglas-Peuker algorithm and maintain only lines with a length of at least 5 m containing at least 100 laser end points (for a scanner with a 0.25° resolution). This clearly eliminates also end-points belonging to walls, but overall, it keeps the number of false-positives small — which is more important for us in order to obtain a robust alignment between laser scan and map.

8.6.1 SLAM exploiting OpenStreetMap data

The first set of experiments is designed to illustrate that we use the information from publicly available maps to locate the robot within these maps. By considering the individual laser scans obtained by the robot within the alignment procedure, we even have the possibility to find loop closures that are partially missed by the pose-graph SLAM otherwise. Figure 8.5 depicts a trajectory of the robot overlayed on the map when using traditional 2D graph-based optimization (red) without considering map information and when incorporating the map structure into the optimization process (green). The map in this case is rendered with $0.3\,m$ per pixel resolution and covers the area of $250\,m$ by $300\,m$. As can be seen, not only the robot's own map is better aligned with the structure of the environment, but also the loop closure was correctly detected due to the aligning laser scans to the buildings. To provide a more quantitative evaluation, we compute the error for the final pose of the robot, as a distance between the optimized pose and manually specified ground truth position. The error of the final pose without map information is about $5\,m$, whereas using the map priors lead to an error for the final pose of about $1\,m$. Figure 8.7 represents another example of the robot's trajectory, here using 3D Velodyne data, which spans over

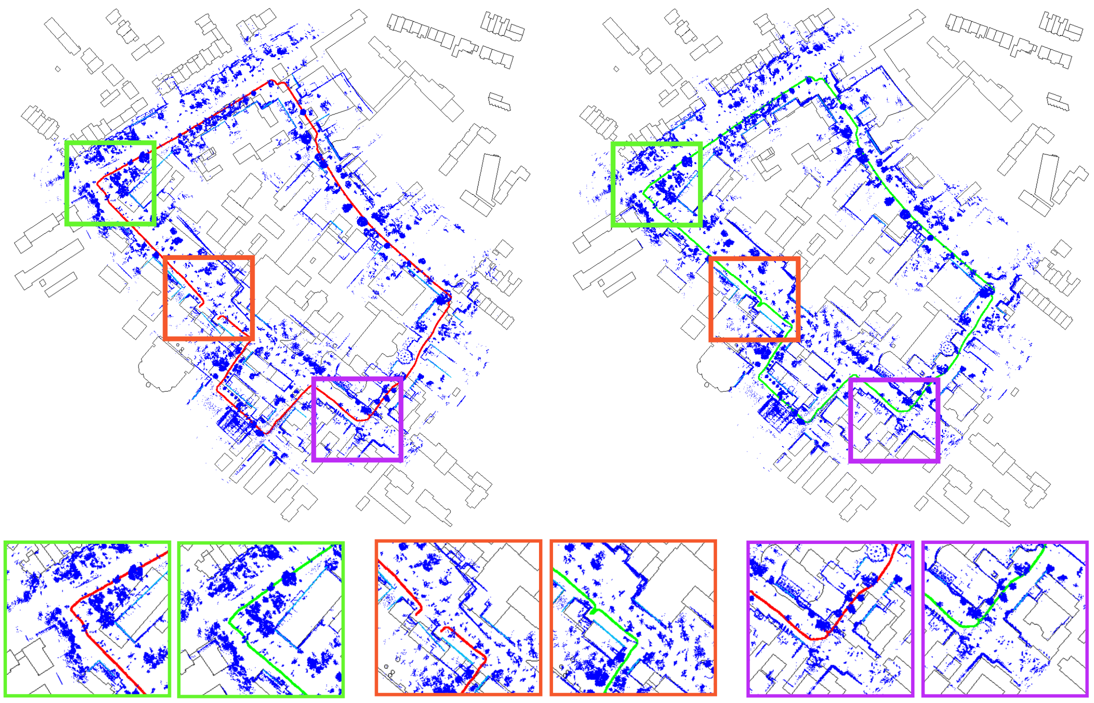


Figure 8.7: Left: overlayed trajectory before the optimization. Right: trajectory after optimization. Bottom: Zoomed-in parts of the trajectory.

a significantly larger area than the previous example. For this experiment the map represents the area of size 500 m by 500 m and thus is rendered with 0.5 m per pixel resolution. In Figure 8.7, only the laser end points that do not belong to the ground plane are plotted. As can be seen, the map produced by the robot is filled with substantial clutter in the environment, which makes the aligning procedure more challenging. Nevertheless, our approach is able to fix the misalignments that come from the inaccuracy of the initial position, see Figure 8.7 bottom row left and right image, as well as to find loop closures missed by the pure pose-graph approach, see Figure 8.7 center. Using the same definition of the localization error as for the previous experiment, the error of the last pose using pure pose-graph formulation is 22 m, whereas with OSM information it is 0.5 m.

8.6.2 Map inconsistencies

The second experiment is designed to show that our approach is able to deal with a certain amount of map inconsistencies. Such inconsistencies result from different sources, for example wrongly mapped buildings, a demolished building that is still present in the map or a building that was built after the time of the map creation. The two examples for inconsistencies shown in Figure 8.8 are real inconsistencies in OSM data and not artificially simulated. As we take the matching-dependent uncertainty into account, the information about the inconsistencies is incorporated into the optimization process as well. For this exper-

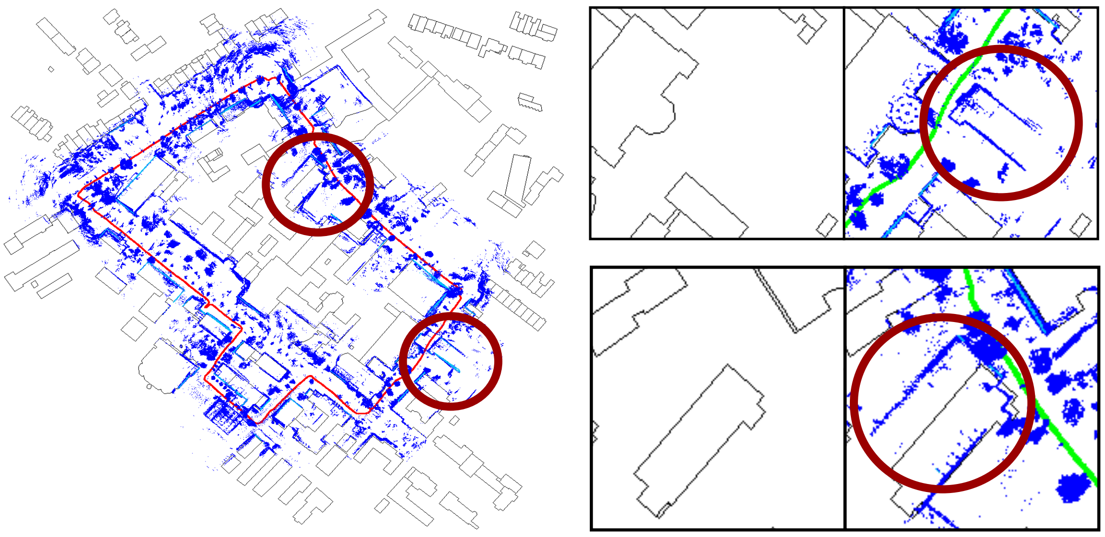


Figure 8.8: To the right zoomed views of the map inconsistencies our system can deal with. Detected buildings are marked with light blue.

iment we used the same trajectory as for the previous experiment. Figure 8.8 (second column) depicts two examples of the map inconsistencies that are successfully handled by our approach. The upper image depicts a situation in which the building is visible in the scan and not present in the map and the image in the bottom shows a case where the building is wrongly mapped (building in the map is too small). Our system deals with inconsistencies through the use of a robust kernel function. Figure 8.9 shows the effect of disabling the robust kernel function. As can be seen, the robot map gets distorted near the wrongly mapped buildings, corresponding places are marked with circles.

8.6.3 Execution time

In this experiment, we show that our approach adds only a small computational overhead to the simultaneous localization and mapping process. We ran our algorithm on different datasets with various size and complexity and summarize the runtime results in the Table 8.1. The datasets are obtained with the robot setup specified before, namely using Husky A200 with Velodyne VLP-16 or Hokuyo laser scanners. As can be seen, the time needed to process additional map knowledge (OSM, 4th and 6th column) is almost negligible in comparison to the time needed for the pose graph SLAM (pose-graph, 2nd and 5th column). This means that we integrated our extension into the optimization procedure without adding a significant computational overhead.

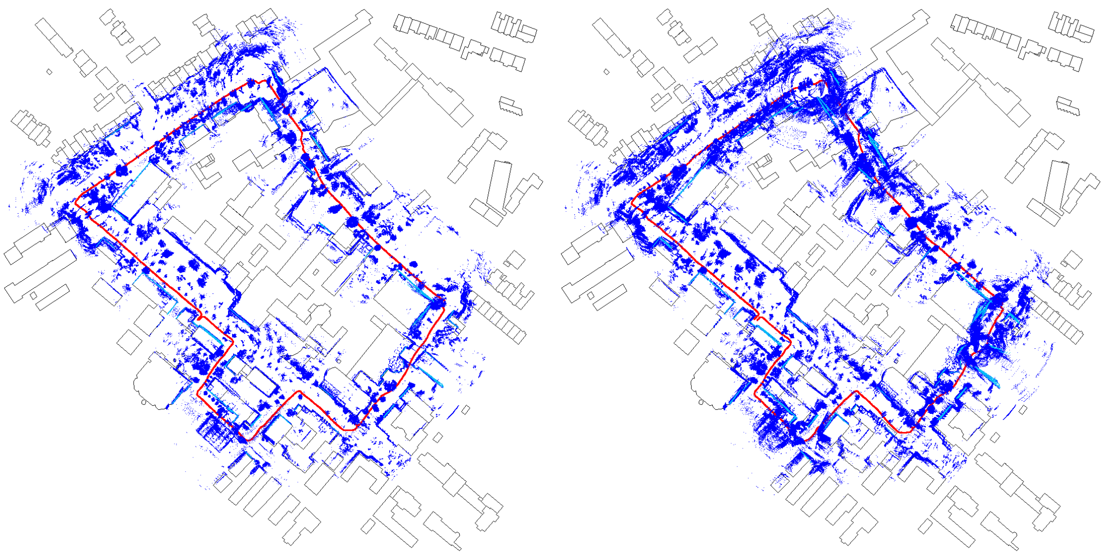


Figure 8.9: Enabling / Disabling robust kernel function (DCS). Left: optimization using DCS. Right: optimization without robust kernel functions.

Table 8.1: Timing results for processing the whole dataset (full) and processing a chunk (per chunk) of the dataset after driving for 25 m; dist - length of the trajectory; pose-graph - processing using standard pose-graph formulation only, OSM — processing time needed to optimize additional edges introduced by OSM constraint.

		full		per chunk	
	dist	pose-graph	osm	pose-graph	osm
dataset 1	168 m	9.89 s	0.9 s	1.75 s	0.16 s
dataset 2	336.6 m	62 s	0.83 s	5.52 s	0.074 s
dataset 3	579.6 m	41.5 s	4.93 s	2.14 s	0.25 s
dataset 4	1040 m	86 s	4.1 s	2.48 s	0.11 s

8.6.4 Active localization

The last set of experiments is designed to show that the localizability maps provide information about the ability of the robot to reduce its pose uncertainty if obtaining scans at given locations in the map. To show this, we initialized the robot's believe with a pose uncertainty of up to 5 m and 20°. Then, the robot had to select a target location within a 100 m range and to update the believe about its own pose based on the measurements acquired on the way. Figure 8.10 illustrates this experiment for two initial locations. In one place, the robot localizes already well given its initial pose (left image) while in the other case, the initial pose does not offer good features for localization (right). For each location, we sample possible target locations and evaluated the ability of the robot to improve its pose estimate while approaching the sampled locations. Several of the selected trajectories end in the likely regions (setup 1 the trajectories 1, 2, 5, 6; setup 2 the trajectories 2, 3), whereas others are located in unlikely regions (setup 1: 3, 4 and setup 2: 1, 4, 5).

We recorded the individual trajectories separately with our robot in Bonn and measured the ground truth locations at the end of each trajectory. Thus, we are able to compute the localization error as the absolute distance between mean estimate and true location. For more quantitative results, we repeated this experiment for randomly sampled starting locations in an area of 5 m and 20° in orientation. The results are summarized in the Table 8.2. The table depicts the number of runs in which the localization errors decreased, increased or even diverged (error larger than 20 m). As can be seen, trajectories that lead through or end in likely regions result in a better localization on average, independent from the starting pose. For the trajectories that lead through the likely regions (setup 1: 1, 2, 5, 6 and setup 2: 2, 3) the localization error is reduced in $86-100\%$ of the cases, whereas for the trajectories that prefer unlikely regions (setup 1: 3, 4 and setup 2: 1, 5, 4) the localization error mostly increases. Additionally, the need of navigating through the likely regions becomes more important if the starting robot pose lies in the unlikely region as in setup 2. If the robot starts in a region that supports localization (and thus it is well localized), the gain of the localizability maps is obviously limited (setup 1: 3). Figure 8.11 shows the distribution of the localization error for setups 1 and 2 respectively, left column belong to setup 1 and right describes setup 2. As can be seen, if the trajectories end up in the likely regions (upper row), the pose uncertainty decreases with the errors less than 5 m. However, if the trajectories end up in unlikely regions for localization (bottom row) the pose uncertainty increases or even the solution completely diverges.

The last experiment is designed to show that trajectories that lead through likely regions in the localizability map lead more often to a decreased localization

Table 8.2: The distribution of the localization errors for the planned trajectories after the execution.

	Endpoint region	trajectory	Localization error, % decreased	increased	diverged
setup 1	likely	1	94	0	6
		2	100	0	0
		5	86	0	14
		6	98	2	0
	¬likely	3	84	4	12
		4	0	94	6
setup 2	likely	3	42	4	54
		2	100	0	0
	¬likely	1	0	100	0
		5	0	84	16
		4	0	100	0

error than the trajectories that that lead through less likely regions. To illustrate this, we recorded the trajectories depicted in the Figure 8.10 (right). We recorded the data using the robot Obelix equipped with a Velodyne HDL-32E in Freiburg. The area depicted in Figure 8.10 (right) and Figure 8.12 is approximately $350\,m$ by $320\,m$. The first trajectory passes through the likely regions and leads to a decrease in the localization error more often than the second trajectory on the right, which leads through regions that do not support localization that well, see Figure 8.12. We would like to point out that the start and end points in this experiment were selected manually so that they either lead trough likely regions in the localizability map or not. Thus, we investigate the relevance of the proposed localizability information on the resulting localization performance of the robot traveling along a path and not the performance of a specific planning or target point selection algorithm.

8.7 Conclusion

In this chapter, we presented a novel approach to improve the quality of maps built with mobile robots by exploiting information from publicly available maps such as Open Street Map data. Our approach seeks to find an alignment between the laser scanner data recorded in the mobile platform and the building information from OpenStreetMap data. In addition to that, we estimate the ability of the robot to localize itself in a given region of the map by computing a so-called localizability map. As we have illustrated through a large set of real world experiments, the exploitation of OSM data improves the map alignment process and provides relevant information about the ability of the robot to localize itself in certain locations.

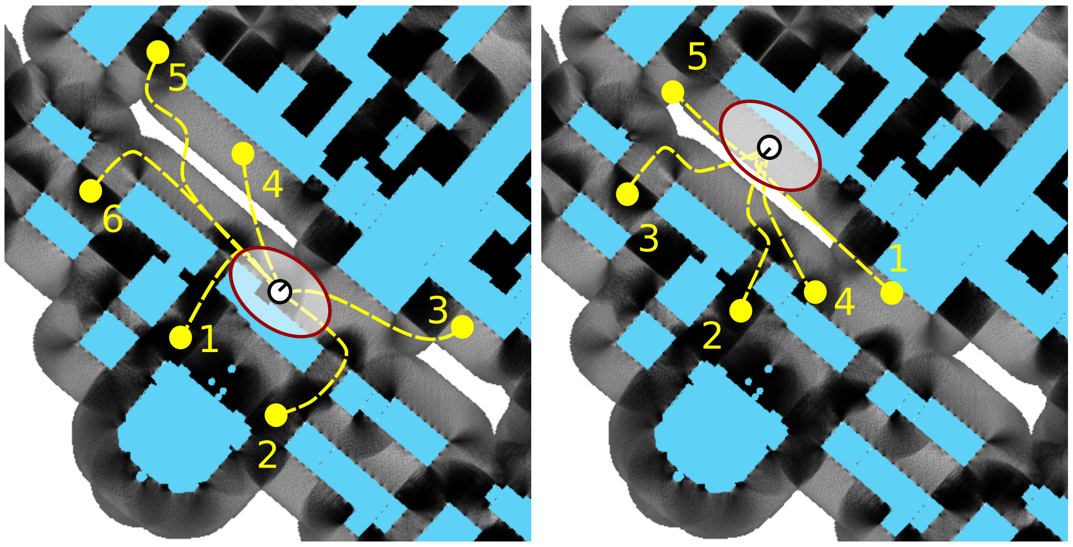


Figure 8.10: Planned trajectories. Left: Robot starts in a location which supports an alignment well (setup 1). Right: Robot starts in a location with low likelihood of being able to compute the right data association between OSM information and its sensor readings (setup 2).

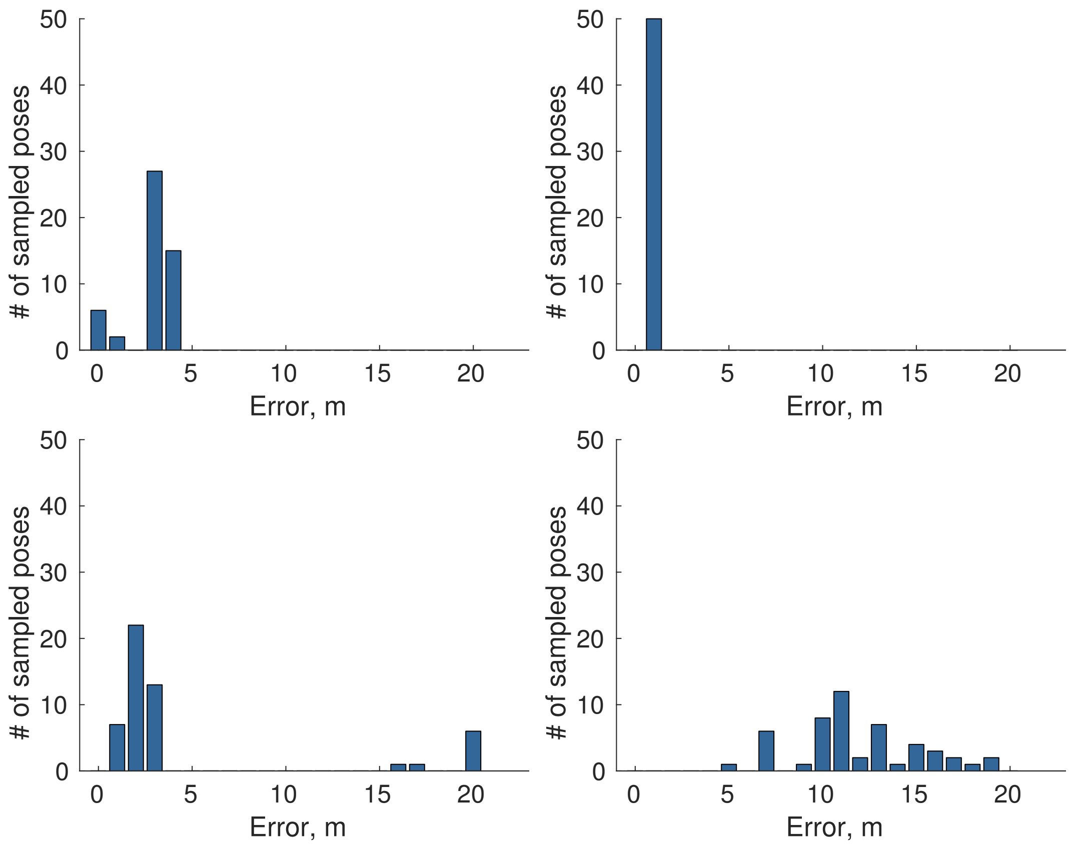


Figure 8.11: Localization error distribution. Left column Setup 1: trajectory 2 with the endpoint in likely region (up) and trajectory 3 with the endpoint in unlikely region (buttom). Right row Setup 2: trajectory 2 with the endpoint in likely region (up) and trajectory 1 with the endpoint in unlikely region (bottom).

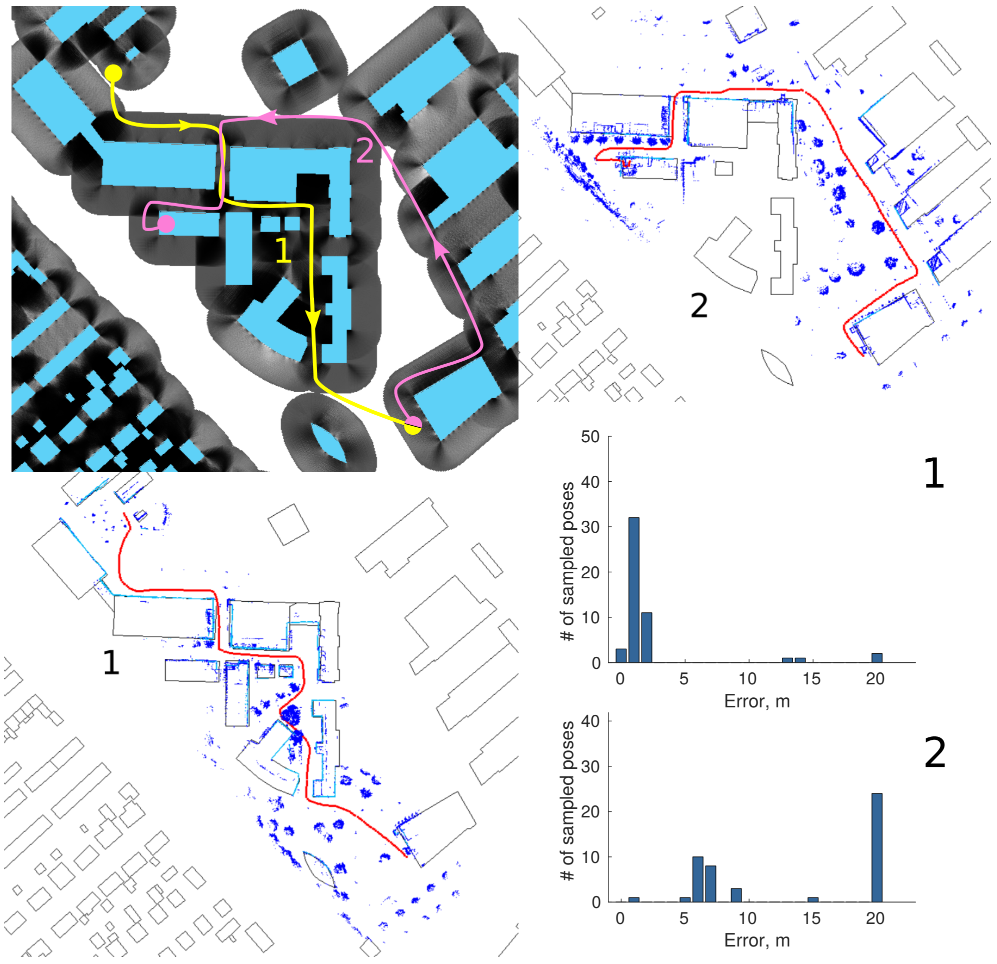


Figure 8.12: Localization experiments in Freiburg. Upper left image shows the localizability information on the campus and overlayed routes of the robot to navigate (1,2). Trajectory 1 passes through the zones with high localizability (black areas), whereas trajectory 2 passes also through the zones of low localizability. The corresponding robot maps are depicted in lower left and upper right images respectively. In the bottom right you can see the localization error distribution for both routes. The plots were generated by applying translational and rotational noise to the starting position of both trajectories. Since trajectory 1 passes through the zones of good localizability, the inaccuracy of starting position is getting compensated by the localization system, whereas the localization diverges for most of the cases for trajectory 2.

Chapter 9

Related work

At the core of every visual place recognition technique lies the fundamental step of describing the images as well as methods for comparing them. In our work, we rely on so-called whole image descriptors as HOGs, NetVLAD, and OverFeat features, whereas other aproaches deploy other kinds of features. We furthermore presented techniques using image sequences and scan alignment to perform place recognition. In this chapter, we provide an overview of the related work in the field of place recognition as well as discuss the advantages of our aproach with respect to others.

9.1 Describing an image with features

Traditional approaches to visual place recognition typically rely on feature descriptors that represent the content of an image and thus the visual appearance of the place. Nowadays, there exists a large variety of feature descriptors that are suitable for visual place recognition including visual place recognition under challenging conditions. In this section, we discuss the features that are shown to perform well for general visual place recognition, i.e., when the appearance of the scene is only influenced by translation, in-plane rotation, and a reasonable amount of tilt as well as modern features extracted from convolutional neural networks.

To describe an image, there exist two main paradigms, except of using the images as it is. The first one is to extract regions of interest in a single image and compute descriptors for those regions. This allows for representing a place as a collection of local feature descriptors. The second paradigm is to compute a whole-image descriptor. These descriptors represent an image as a whole and avoid representing them by specific regions.

The most popular local feature descriptors are scale-invariant feature transform (SIFT) descriptors [82] and speeded-up robust features (SURF) [20]. Both of these feature descriptors are based on building histograms of oriented gradients of the intensity values in a local neighborhood. SURF is a speeded-up version of SIFT that produces a descriptor of smaller size and makes use of the integral images. One can also opt for upright SURF only, which neglects the orientation estimation, and results in a faster computation time. SIFT and SURF are shown to be a powerful tool for place recognition especially within SLAM frameworks [118, 125, 52, 7, 6]. An extensive evaluation of the different feature descriptors performance can be found in the work of Mikolajczyk *et al.* [90]. The authors test several feature descriptors like SIFT [82], PCA-SIFT [70] and propose a new descriptor Gradient location-orientation histogram GLOH. GLOH is an extension of the SIFT descriptor designed to increase the robustness and distinctiveness of the descriptor. In their experiments, the authors show that GLOH performs better than SIFT as well as other feature descriptors.

In order to extract meaningful feature descriptors, one needs to select the regions of interest or a keypoint in the image, where the chosen descriptor should be extracted. Several descriptors like SIFT and SURF already have a built-in functionality to select regions of interest. For example, SIFT relies on a blob detection scheme known as Difference of Gaussians (DoG). For those descriptors that do not have a keypoint detector, there exist independent alternatives. One of the most intuitive regions of interest are corner points, since the corner elements can be easily detected in the gradient space and are stable and repeatable across the images. They can be detected using Harris corners [63], Shi and Tomas features [120], the Förstner operator [44], or features from accelerated segment test (FAST) [107, 108]. It is not uncommon to mix different keypoint and descriptor strategies. For example Mei *et al.* [88] use FAST detection technique to find keypoints and SIFT for further description.

As proposed by Lowe *et al.* [81], widely used parameters for SIFT descriptors results in a 128 dimensional vector. Since it uses floating point numbers, a descriptor vector requires 512 bytes. Similarly, SURF requires 256 bytes (for 64-dim). Creating such a vector for thousands of features can take a lot of memory, which is not desirable for resource-constraint applications. This led to the development of more lightweight feature descriptors - binary descriptors, for example, Binary Robust Independent Elementary Feature (BRIEF) [27]. This is one of the most simplistic binary descriptors that describes an image patch by randomly sampling pairs of image pixels and constructing a binary feature descriptor based on relation of the sampled pixel intensities. The BRIEF descriptor is fast to compute, however, in its plain form, is not robust to in-plane rotation and scale. To come up with a descriptor that possesses these missing properties, Rublee *et al.* [110] proposed the ORB descriptor, which stands for Oriented Fast and Rotated BRIEF. Nowadays, ORB is almost the standard choice for a binary feature descriptor. Other researches also proposed BRIEF alternatives like BRISK [79] and FREAK [4] that make binary descriptor rotation and scale invariant. The main advantage of binary descriptors is that they are memory efficient and fast to compute and compare. On the other hand, they provide a simplified representation of the local patches and thus may be not robust enough, especially in the presence of visual appearance changes such as illumination changes.

Global image descriptors, on the other hand, have the potential to tolerate strong illumination and dynamic changes better in comparison to local feature descriptors. This capability comes from the fact that most whole image descriptors preserve the spatial relation between the objects in the scene. In this way, the relative position of the features is expected to be roughly similar between the images. This property can improve the image matching process whenever the scene experiences strong illumination changes. Under illumination changes the intensities of the image gradient may change dramatically, so that local features descriptors are not able to cope with. Global feature descriptors can be generated from local feature descriptors by predefining keypoint locations, for example defining a grid-like structure for computing descriptors. Badino *et al.* [14] deploys whole-image descriptor based on SURF for localization applications, while Sünderhauf and Protzel [127] use BRIEF in a similar way to propose BRIEF-Gist. Another popular whole image descriptor is Gist proposed by Oliva and Torralba [102] to model the shape of the scene and was used for place recognition in paper presented by Murillo *et al.* [96]. Since global image descriptors consider an image as a whole, they are able to perform well under illumination changes but their performance drops substantially in the presence of translational or rotational effects, because the same features may not located in the same grid cells. To preserve illumination invariance and to better tackle camera transformations, Naseer *et al.* [99] proposed to use the global descriptor HOG [35] on an image grid and Sünderhauf *et al.* [128] use Edge Boxes method for objects detection [146] and extract features in the detected object regions. This allows for matching features that correspond to particular objects across the images. For our visual place recognition system presented in Chapter 3, we select to deploy holistic, also called whole image or global descriptors, as they were known to show better performance under photometric changes.

Since global image feature descriptors represent the whole image, we are able to compute a matching score for two images directly by comparing individual feature descriptors. In case of local feature descriptors, there might be thousands of features generated per image and comparing two images in an all-to-all manner can be a computationally expensive operation. The bag-of-words model proposed by Sivic and Zisserman [122] for image retrieval increases efficiency of place recognition by quantizing the local features in to a vocabulary. Effectively leading to the fact that every feature vectors is summarized by a visual word. Afterwards, an image is described by the histogram of occurrences of vocabulary words in it. This reduces the problem of matching images to a problem of comparing image histograms, which is much more efficient. In their work, the authors propose to combine the SIFT descriptor with affine covariant regions that give a region description vectors which are invariant to affine transformations of the image. Applying *tf-idf* ideas from the text retrieval community, allows for down-weighting the frequently occurring visual words and for enhancing the rare occurring ones. This leads to a better distinction between images and thus better image retrieval properties. Another popular approach from the robotics community that uses the concept of bag-of-words is FAB-MAP [34]. The authors additionally propose to use inverted index structure to speed-up the retrieval capabilities. The inverted index is a data structure that stores for every feature vector indices of the images it was detected in. Furthermore, FAB-MAP uses a Chow Liu tree to learn offline the word's co-visibility probabilities. These improvements made FAB-MAP a standard appearance-only SLAM system that is based on SIFT descriptors. As also noted previously, the SIFT descriptor is robust against photometric image changes. This, however, comes at a computational cost. To bring image retrieval closer to real world applications that require real time operation capabilities, Gálvez-López and Tardos [49] proposed the approach DBow that uses bag-of-words technique combined with binary features, namely FAST and BRIEF. To speed up the retrieval capabilities even further, the authors have proposed to perform a temporal consistency check as well as to apply a direct index. Direct index is a data structure that stores for each image the indices of the features that were extracted from it.

In context of binary feature descriptors, a further popular image retrieving strategy is a locality sensitive hashing (LSH) [84]. In contrast to conventional hash functions where hash collisions should to be avoided, it aims at maximizing the probability of a collision for similar inputs. This results in the fact that similar images are assigned to the same or near-by buckets of the hash tables. Selecting a matching image simplifies the matching procedure by comparing only to a number of candidate images as oppose to the whole dataset of images. In our work, we use the ideas of locality sensitive hashing to tackle the kidnapped robot problem as well as the problem of quickly finding the set of potential images candidates for successful relocalization. We discuss this approach as well as details about different hashing schemes in the subsequent sections of this thesis.

Recent advance in object recognition and detection using convolutional neural networks (CNN) led to new types of image descriptors. One of the first researchers who explored the properties of CNN feature vectors in the context of visual place recognition were Chen *et al.* [32]. They found that feature vector

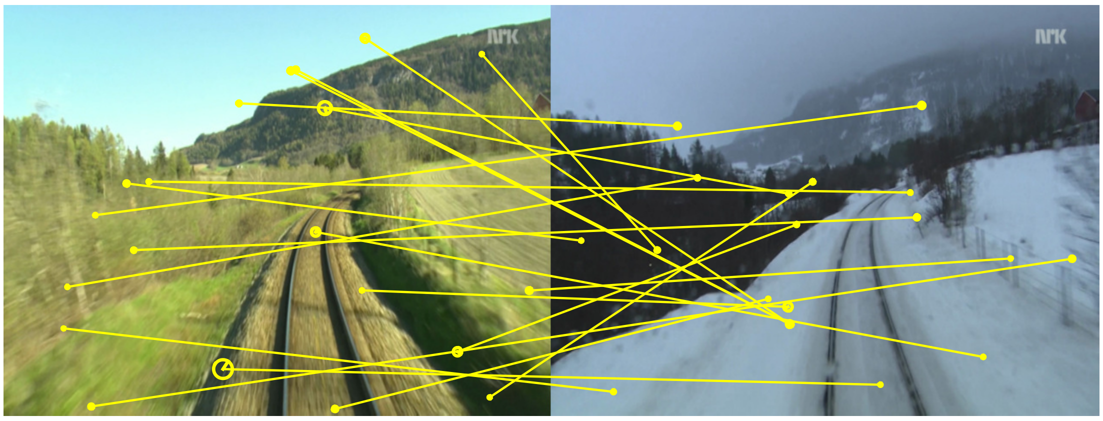


Figure 9.1: Illustration of the fact that SIFT features do not perform well under strong seasonal variations. As a result of the seasonal changes, most SIFT matches illustrated by the lines between the images are outliers as the lines do not connect corresponding points.

from certain layers of the convolutional neural network OverFeat [119] can be successfully used as a holistic image descriptor. After this initial proposal, Sünderhauf *et al.* [130] published a comprehensive study on performance of ConvNet features for visual place recognition in changing environments. In that paper, the authors investigate, which layers of the AlexNet [75], originally proposed for image classification, are suitable for particular visual appearance changes. Advances in neural networks spark the interest of creating new feature descriptors. A recent comprehensive study of local feature descriptors by Balntas *et al.* [18] reveals, however, that simple normalization of the traditional handcrafted features can boost their performance to the level of deep learning based descriptors within a realistic benchmarks evaluations.

In our approach, we opted for the CNN features that from our experience show the best performance for the visual place recognition in changing environments. In Chapter 4, we use a feature vector from 10th layer of convolutional neural network OverFeat [119]. We found those features to be more discriminative than tessellated HOG descriptors, which is the property that fosters better place recognition. With increasing interest for the convolutional neural networks, many more networks appeared in the community that proposed better results for classification, object detection and networks for image recognition. All of these networks were trained for the tasks different from place recognition but showed to produce more discriminative descriptors for place recognition than traditional feature descriptors. Even with such improved features, the place recognition task was still considered far away from solved especially for the recognition in changing environments. Arandjelovic *et al.* [8] propose a NetVLAD architecture, a convolutional neural network that was trained specifically for the place recognition task. One of the properties of the NetVLAD network is its ability to aggregate local features in the final layer. The network was inspired by ideas of Vector of

Locally Aggregated Descriptors (VLAD) [9] encoding. This encoding can be seen as an extension of the bag-of-words approach in which the feature descriptor also stores the distances to the cluster (word). So, the NetVLAD network produces a feature vector which is aggregated from features of a final layer. This brings the ability for the feature vectors to be translational and rotational invariant, the property that all CNN features were missing. Additionally, the NetVLAD features are comparably low dimensional, which makes them suitable for the real world application. So in the process of working on this thesis, in more detail in Chapter 6, we change our feature descriptor to the features from NetVLAD network and as a result we are able to better recognize places in presence of dramatic view point change, e.g., by localizing against Street View images.

9.2 Visual place recognition

Pose estimation is a frequently studied problem in robotics and different approaches have been proposed for visual localization [3, 22, 34, 36, 47]. The ability to localize is an essential prerequisite for most autonomous navigation systems. Visual place recognition has received a significant amount of attention also in computer vision community [10, 28, 30, 73, 114, 115, 116, 134, 133] as well as robotics. Some approaches focus on autonomous driving [87] while others on applications for augmented reality [89] or geo-localizing archival images [12]. Dealing with substantial variations in the visual input has been recognized as an obstacle for persistent autonomous navigation and this problem has been investigated by different researchers [34, 55, 80].

The majority of visual place recognition systems exploit features such as SURF [20] or SIFT [82] and several approaches apply bag-of-words techniques, i.e., they perform matching based on an appearance statistics of such features. To improve the robustness of appearance-based place recognition, Stumm *et al.* [126] consider the constellations of visual words and keeping track of their covisibility. Often, approaches using SIFT or SURF show a great performance if the appearance of the environment does not change radically. As also noted before, the matching performance of SIFT or SURF degrades under strong perceptual changes. An example, which illustrates this fact, is shown in Figure 9.1. The two images are taken from the same place and similar view points but during different seasons. As can be seen from the matches illustrated through the yellow lines, most of the correspondences are outliers. This examples illustrates our experience, that SIFT and SURF features are not well-suited for image matching under strong seasonal variations. Across season matching using SIFT and SURF has been investigated by Valgren and Lilienthal [136] by combining features and geometric constraints, which can improve the matching. In Chapter 3, we deploy a tessellated HOG features following the ideas of Naseer *et al.* [99, 138]. In contrast to that, in subsequent Chapter 4, Chapter 5 we apply deeply learned features proposed by Sermanet *et al.* [119] and suggested for place recognition by Chen *et al.* [32]. We use them as an alternative to tessellated HOG features as they provide a better matching performance in our settings. Another recent work [131] suggests a technique for place recognition, where features stem from convolutional neural networks. The authors extract features from the landmark proposals, construct the similarity matrix by comparing the landmark features using the cosine distance and also take into account the size of the bounding boxes for the matched landmarks. The recognition task is then performed by selecting individual matches based on the highest similarity score.

Visual place recognition can also be formulated as a sequence matching problem. In terms of aligning image sequences, several approaches have been proposed. For example, Matsumoto *et al.* [86] use the image sequences and directional relations between images to perform visual navigation in a corridor environment. SeqSLAM [92], which also aims at matching image sequences under strong seasonal changes, computes an image-by-image matching matrix that stores dissimilarity scores between all images in a query and database sequence. SeqSLAM computes a straight-line path through the parts of a matching matrix and select the path with the smallest sum of dissimilarity scores across image pairs to determine the matching route. Milford *et al.* [91] also present a comprehensive study about the SeqSLAM performance on low resolution images. Related to that, Naseer *et al.* [99] focus on offline sequence matching using a network flow approach. If odometry is available, this approach can also be combined with a least squares SLAM system to build metric maps [98]. Another way to perform robust sequence matching is to create a notion of the place, by combining the descriptors from couple of consecutive images in so called subsequence descriptor as presented by Bampis *et al.* [19].

The approach by Neubert *et al.* [101] aims at predicting the change in appearance on top of a vocabulary. For the vocabulary, the method predicts the change of the visual word over different seasons but the learning phase requires an accurate image alignment over seasons. A subsequent approach by Johns and Young [69] builds a statistic on the co-occurrence of features under different conditions. It relies on the ability to detect stable and discriminative features over different seasons. Finding such discriminative and stable features under strong changes is however a challenge. To avoid addressing the problems of finding features that are robust under extreme perceptual differences, Churchill and Newman [33] store different appearances for each place. These so-called experiences enable a robot to localize in previously learned experiences and associate a new data to places. A recent extension of that paradigm targets vast-scale localization by exploiting a prioritized recollection of relevant experiences so that the number of matches can be reduced [80].

Long-term place recognition has not only been addressed using cameras but also other sensing modalities such as LiDARs. For exmaple, Biber and Duckett [24] address the problem of dealing with changes in the environments by representing maps at different time scales. Each map is maintained and updated using the sensor data modeling short-term and long-term memories. This enables handling variations. In contrast to that, Stachniss and Burgard [124] model different instances of typical world states using clustering in the 2D grid representations of the world. There are furthermore approaches combining laser and visual information for large-scale localization at city scale. The approach of Pascoe *et al.* [104] exploits laser data and vision information during the mapping phase with a survey vehicle but can then localize a car only using a camera.

To achieve a visual localization in a long term autonomy setup, Furgale and Barfoot [48] propose a teach and repeat system that is based on a stereo setup. The approach exploits local submaps and enables a robot to navigate on long trajectories but this method does not address large perceptual changes with respect to the taught path.

Another set of approaches for visual place recognition rely on extracting the 3D geometry of the scene. Dubé *et al.* [39] propose a loop detection algorithm based on the matching of 3D segments to be able to reliably detect and close loop in real time. For environments that do not change their visual appearance dramatically, loops can be detected with high precision by using a probabilistic voting concept based on nearest neighbors descriptor voting proposed by Gehrig *et al.* [50]. An approach by Caselitz *et al.* [29] also aims at localizing the monocular camera within the previously build 3D LiDAR maps. For fast loop closure detection within a sequence that has been acquired through the continuous camera movement, Schlegel and Grisetti [117] track binary features and present a method for efficient similarity search using decision trees.

In Chapter 4, we introduced a lazy data association scheme inspired by the work of Hähnel *et al.* [61] to come up with an online solution that can be executed on a robot while navigating. A key goal is to reduce the number of image-to-image comparisons with respect to existing methods such as [92, 99, 138]. In contrast to the work by Hähnel *et al.*, we use a heuristic that considers the cost of the path taken so far in order to speed-up the search. As we showed in this thesis, it allows us to dramatically reduce the number of image comparisons which is one of the elements of realizing an online place recognition system.

In visual place recognition, relocalization after getting lost can be achieved by a brute force approach to image retrieval such as comparing the query image to all images in the reference dataset as it was used by Neubert *et al.* [100].

To optimize the process of finding similar images in large datasets Gionis *et al.* [53] propose using hashing algorithm, which was intensively used to solve text retrieval problems, to search for duplicates and even similar images in the large dataset, known as locality sensitive hashing (LSH). In LSH, slight variation in the image domain should only lead to slight variations in the hash. This leads to the fact that images with similar feature descriptors will land in the same buckets. When using hashing technique, we obtain mostly constant query time and small amount of candidate images to further refine the search. To achieve good retrieval performance on real world data LSH requires a substantial amount of different hash table to differentiate between the images with similar feature vectors and different visual appearance. This may be seen as a disadvantage of LSH, since requires a quite large number hash tables (> 100 is suggested in practice) to obtain a high retrieval accuracy. To tackle this problem, Lv *et al.* [84] propose an efficient indexing strategy, which allowed to reduce the number of hash tables. The popularity of the CNNs resulted in learned features, which are high-dimensional in comparison to those used by Lv *et al.* This slows down multi-probe LSH when matching full image sequences. An alternative approach to improve retrieval is spectral hashing [144], where a variant of spectral clustering is performed on the database before the operation in order to find better hash codes. Due to the high- dimensionality of CNN features, spectral clustering becomes computationally intractable and thus we rely on a variant of LSH proposed by Lv *et al.*

One of the particularly challenging branch of relocalization is the so-called kidnapped robot problem. Here, the robot is said to be kidnapped when the consecutive measurement that a robot records differ completely as if the robot was instantly teleports in some other place in the environment. For the localization systems that take the sequentiality of the sensor readings into account, this situation breaks the localization process. One of the ways to tackle this issue by deploying particle filter methods [46]. In our framework, we are able to deal with kidnapped robot problem by performing efficient hashing based relocalization step.

Visual place recognition plays a major role for robot localization. Typically, a visual localization system finds the images that represent the same place and afterwards tries to estimate the 6-DoF transformation between them. This provides an exact position of the camera in a given reference system, which is a typical final goal of the localization systems. Instead of performing these two steps, Kendall *et al.* [71] propose to train a so-called regression neural network PoseNet that given an image returns the 6DoF pose of the camera directly. The advantage of this approach is obtaining a precise pose estimate directly given an image. This, however, is shown to be possible only for limited amount of the environments.

Radwan *et al.* [106] propose an architecture that employs a multitask learning approach to exploit the inter-task relationship between learning semantics and not just regresses 6-DoF global pose but also produces an odometry estimate.

9.3 Improving localization of the robots using publicly available data

Typical approaches to visual place recognition start with collecting the datasets, sequences or experiences to recognize later on the places against. Collecting this data is a time and resource consuming operation. Recently, there started to appear several approaches in the literature that tried to overcome the burden on collecting the reference dataset by exploiting already exciting sources, like Google Street View or other publicly available sources. Badino *et al.* [13] proposed a method for long-term vehicle localization that can localize a vehicle with respect to previously collected topometric map as well as Google Street View. Their method deploy local keypoint features U-SURF [20] and performs localization and tracking by applying discrete Bayes filter. To implement the state transition probability density function, the authors assume to know the velocity of the robot at every point in time, whereas our approach only relies on maximal possible velocity or in other words maximum possible distance in frames (fanout). Another approach by Majdik *et al.* [85] uses Google Street View to localize a micro aerial vehicle. This setup imposes particular viewpoint challenge which they overcome by generating virtual views and match them against the street view images. Agarwal *et al.* [1] also use the imagery from Google Street View to perform a metric robot localization. They compute rigid body transformation between input image stream from the monocular camera and geotagged rectilinear panoramic views. Afterwards, they perform a two phase nonlinear least square estimation to obtain the refined robot poses. The authors rely on a rather inaccurate, consumer GPS device to preselect the set of panoramas to perform metric localization against. In contrast to that, our approach can directly provide the matching street view image to perform a more precise metric localization.

Besides localization, mapping also plays an important role in robotics. The first work in robotics that addressed SLAM through least squares was the work of Lu *et al.* [83]. Subsequently, Gutmann *et al.* [60] focused on means for constructing pose-graphs and for detecting loop closures. Over the last 15 years, a large number of different approaches to graph-based SLAM have been proposed, for an overview see [15, 16, 57, 113, 123].

One prominent example of graph-based SLAM approaches is the work by

Konolige *et al.* [74] that describes a pose-graph implementation for building the linearized system in an efficient way. Solving the linear system of equations leads to optimizing the robot poses and thus map estimates. Olson *et al.* [103] investigates the use of stochastic gradient descent and Grisetti *et al.* [59] proposed an extension of Olson's approach that uses a tree parametrization of the nodes in 2D and 3D. Thrun's GraphSLAM approach [132] applies variable elimination techniques to reduce the dimensionality of the optimization problem as well as hierarchical techniques. Most SLAM approaches assume Gaussian errors in the constraints. This renders them sensitive to data association outliers. A number of approaches has been proposed to overcome this problem. For example, the approach of Sünderhauf *et al.* [129] scales the effect of potential outlier constraints while [2] proposed dynamic covariance scaling as an alternative scaling approach that does not increase the number of variables that need to be optimized. RRR, short for Realizing Reversing Recovering, proposed by Latif *et al.* [78] tries to identify outliers by searching for the set of edges that are consistent with each other. It then rejects potentially wrong constraints.

Recently, several localization approaches were proposed that use the information from OpenStreetMap [62] to improve robot localization. Most of them incorporate this information into the observation model of the Monte Carlo localization (MCL). For example, Hentschel *et al.* [65] represent buildings as 2D line features. This line map is then used to calculate the expected range measurement at a certain robot's locations and combines MCL with a form of Kalman filtering. In Chapter 8 of this thesis, we also use the information about buildings, but integrate the correspondences through an ICP (Iterative Closest Point)-based matching procedure into a graph-based SLAM framework. Another approach, which is proposed by Floros *et al.* [43], uses chamfer matching to align robot's trajectory with the road network extracted from publicly available maps. Each particle in MCL is weighted according to the reported chamfer matching cost. In contrast to that, we select building information as this enables our system to deviate from the exact structure of the road network. Also, typically the metrical information about the road size is missing. Moreover, buildings are easier to detect in 2D range scans then the road surface. Ruchti *et al.* [111] also use the information about the road network. Instead of relying on visual odometry as in [43], they classify the 3D laser scans into road/non-road surfaces and the classification result is incorporated into the weight of the particles in the Monte Carlo localization. In contrast to that, our work incorporates building information obtained from OpenStreetMap into graph-based SLAM as additional edge constraints. The approach by Pink *et al.* [105] generates features like markings of the street lanes from aerial images, matches them to the features extracted from the camera mounted in the car, and uses this information in visual navigation framework. In our work, we use a laser scanner and thus are not bounded to follow the road network. Similar to the other approaches, Brubaker *et al.* [26] also consider the road network from the OpenStreetMap and a camera pair to perform localization.

Another approach to maintain the global consistency of the robot's maps was proposed by Kummerle *et al.* [77]. The consistency is achieved by augmenting the pose-graph formulation with additional constraints that come from the matching the robot's perceptions to the information from the aerial images. The aerial images are transformed into a line map using the Canny edge detector, whereas we render the map directly from the OSM information. Additionally, the authors use a variant of Monte-Carlo localization for localizing the robot within the line-map and then optimize the robot poses using a pose-graph SLAM formulation. In contrast to that, we use the information about the building locations directly from the publicly available data and incorporate this information directly into the pose-graph formulation without deploying Monte-Carlo sampling techniques.

Computing likelihood maps for localization is a well studied problem, for example in the field of active vision. The main focus of the work proposed in this thesis is to find the suitable vantage points for the camera for better object detection or to enhance the visual SLAM algorithms [17, 5, 31, 72]. In our work, we are interested in the similar goal of finding regions where good vantage points are located, but our primary sensor setup is a laser scanner. In the robotics community, the problem of estimating the localization likelihood maps or localizability maps has also been studied in context of Teach and Repeat paradigm. For example, Furgale *et al.* [48] compute a teach corridor within which the robot can localize well in a repeat phase. Dequaire *et al.* [37] deploy Gaussian Processes to predict the localization envelope, the region around the taught trajectory, where the robot obtains reliable visual features for localization in a repeat phase. The authors use robust visual features as well as local path curvature to make the predictions. Velez *et al.* [137] propose an approach to improve the object detections by planning the navigation in that way that allows the detector to be certain about the object. In the vicinity of each detectable object, they compute a likelihood field that indicates locations where reliable measurements can be taken. In our work we do not rely on any pre-trained detectors. An approach of Nardi *et al.* [97] uses the localizability information developed in this thesis to make uncertainty aware path planning in outdoor environments.

The work with which our approach for estimating localizability shares most similarity is the work of Roy *et al.* [109]. In this work, the authors present an approach for navigating a robot, called coastal navigation, which generates the trajectories for the mobile robot that reduce the likelihood of localization errors. They estimate the likelihood of a point in the map as the amount of information content. It is computed as the difference between the expected entropy of the robot's pose given a sensor measurement and the entropy of the prior belief about the pose. The more information the cell in the map contains, the higher the likelihood. In our case, we compute the eigenvalues of the covariance estimate of the robot's pose and consequently, the smaller the selected eigenvalue, the smaller the uncertainty and the higher the likelihood will be. For the computational reasons, Roy *et al.* [109] assume to have a map of the environment constructed by a robot and the prior probability distribution about the robot's position. In our approach, we also consider to have a map of the environment, but in the form of a coarse map, rendered from the OpenStreetMap data.

Chapter 10

Conclusion

In this thesis, we showed that visual place recognition in changing outdoor environments is a challenging and active research topic. There exist multiple approaches tackling this type of place recognition problem. In our approaches, we consider sequence information and formulate the place recognition problem as a graph-based sequence matching and search problem, which can be efficiently solved using the approaches proposed in this thesis.

As a first contribution, we proposed an approach to visual image matching under substantial appearance changes by exploiting sequence information. The proposed approach is an extension for an approach by Naseer *et al.* [99] so that it can exploit noisy GPS pose priors and at the same time substantially reduces the number of required image comparisons. This enables us to find data associations orders of magnitude faster than the previous approach. In addition to that, our approach can naturally handle loops in the input image sequences. We implemented and tested our approach using real-world image and GPS data acquired in summer and in winter. Throughout our extensive experimental evaluation section, we showed that our the approach can increase the matching performance while reducing the computation time and in this way outperforms the existing methods such as SeqSLAM [92] and others.

To further bring the matching approach to real-world application, we further proposed an incremental approach to visual image sequence matching under substantial appearance changes that is able to operate in online fashion. The key idea is to apply a lazy data association approach and to define a heuristic for the search in the data association graph that estimates the similarity of images. This enables us to achieve online performance for image sequence matching under substantial appearance changes. We furthermore illustrated that noisy location priors can be exploited during an online search. We implemented and tested our approach using real-world image sequences acquired in summer and in winter as well as under different weather conditions. Our comparisons to other methods as well as the results from the VPRiCE 2015 place recognition challenge suggest that our approach provides competitive results and avoids expanding large portions of the data association graph or building a large matching matrix.

A further contribution is the image sequence matching approach that is able to deal with flexible robot trajectories. It relaxes the assumption that the reference and query image sequences should be roughly synchronized. This allows considering situations that frequently occur in everyday robot operation. For example, the deviation of query trajectory from the previously mapped areas or dealing with situations when reference trajectory visits the same place multiple times. We build a data association graph incrementally and search for a data association sequence using an effective search heuristic. This contribution overcomes two key limitations of our previous method. First, we provide an efficient way for re-localizing the robot in situations, in which it got lost after the robot has left the previously mapped areas and is reentering the known part of the environment or to solve the kidnapped robot problem. Second, our new approach can deal with loops in the reference sequences effectively without additional pose priors, like GPS, while simultaneously avoiding expensive network flow search as proposed by Naseer *et al.* [99]. We implemented and evaluated our approach on different publicly available datasets. Our evaluations and comparisons show that we can handle the above-mentioned situations, which could not be solved with the approach in [140]. We furthermore show through our experiments that our approach runs online, provides an effective image matching in the presence of flexible robot trajectories.

Our final contribution to improving the visual place recognition is a novel approach for finding image correspondences between a currently observed image stream and a map consistuing of several previously recorded image sequences under substantial appearance changes. Matching, in this setup, is also performed through an informed search in a data association graph that is built incrementally. By deploying the hashing technique, we are able to relocalize the robot if it is lost as well as between multiple image sequences. Additionally, we showed how to leverage publicly available Google Street View imagery within our framework. Our evaluations show that we can perform place recognition faster than offline, fully informed search with the comparable matching performance in presence of drastic visual appearance changes as well as viewpoint changes.

In this thesis, we also contribute to improving the quality of maps built with mobile robots by exploiting information from publicly available maps such as Open Street Map data. Our approach seeks to find an alignment between the laser scanner data recorded in the mobile platform and the building information from OpenStreetMap data. In addition to that, we estimate the ability of the robot to localize itself in a given region of the map by computing a so-called localizability map. As we have illustrated through a large set of real-world experiments, the exploitation of Open Street Map data improves the map alignment process and provides relevant information about the ability of the robot to localize itself in certain locations.

Parts of this thesis are contributions to the European project EUROPA2, grant number FP7-610603-EUROPA2, the DFG research unit FOR 1505 Mapping on Demand. Furthermore, we ontributed to the Excellence cluster PhenoRob supported by the German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] P. Agarwal, W. Burgard, and L. Spinello. Metric Localization using Google Street View. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[2] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust Map Optimization using Dynamic Covariance Scaling. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, Karlsruhe, Germany, 2013.

[3] M. Agrawal and K. Konolige. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Trans. on Robotics (TRO)*, 24(5), 2008.

[4] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517, 2012.

[5] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *Intl. Journal of Computer Vision (IJCV)*, 1(4):333–356, 1988.

[6] H. Andreasson and T. Duckett. Topological localization for mobile robots using omni-directional vision and local features. *IFAC Proceedings Volumes*, 37(8):36–41, 2004.

[7] A. Angeli, S. Doncieux, J.A. Meyer, and D. Filliat. Incremental vision-based topological slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1031–1036. ieee, 2008.

[8] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307, 2016.

[9] R. Arandjelovic and A. Zisserman. All about vlad. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[10] R. Arandjelović and A. Zisserman. Dislocation: Scalable descriptor distinctiveness for location recognition. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, pages 188–204. Springer, 2014.

[11] R. Arroyo, P.F. Alcantarilla, L.M. Bergasa, and E. Romera. Fusion and Binarization of CNN Features for Robust Topological Localization across Seasons. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[12] M. Aubry, B. Russell, and J. Sivic. Painting-to-3d model alignment via discriminative visual elements. *ACM Trans. on Graphics (TOG)*, 33(2):14, 2014.

[13] H. Badino, D. Huber, and T. Kanade. Visual topometric localization. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 794–799, 2011.

[14] H. Badino, D. Huber, and T. Kanade. Real-time topometric localization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1635–1642. IEEE, 2012.

[15] T. Bailey and H.F. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I. *IEEE Robotics and Automation Magazine (RAM)*, 13(2):99–110, June 2006.

[16] T. Bailey and H.F. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine (RAM)*, 13(3):108 –117, September 2006.

[17] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[18] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. HPatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] L. Bampis, A. Amanatiadis, and A. Gasteratos. Encoding the description of image sequences: A two-layered pipeline for loop closure detection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4530–4536. IEEE, 2016.

[20] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Journal of Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[21] O. Bengtsson and A. Baerveldt. Robot Localization Based on Scan-Matching – Estimating the Covariance Matrix for the IDC Algorithm. *Journal on Robotics and Autonomous Systems (RAS)*, 44(1):29–40, 2003.

[22] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer Tracts in Advanced Robotics*, pages 143–157. Springer Verlag, 2006.

[23] P.J. Besl and N.D. McKay. A Method for Registration of 3-d Shapes. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[24] P. Biber and T. Duckett. Dynamic Maps for Long-Term Operation of Mobile Service Robots. In *Proc. of Robotics: Science and Systems (RSS)*, pages 17–24, 2005.

[25] I. Bogoslavskyi, M. Mazuran, and C. Stachniss. Robust Homing for Autonomous Robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[26] M. Brubaker, A. Geiger, and R. Urtasun. Map-based probabilistic visual self-localization. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 38(4):652–665, 2016.

[27] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 778–792. Springer, 2010.

[28] S. Cao and N. Snavely. Graph-based discriminative learning for location recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 700–707, 2013.

[29] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard. Monocular Camera Localization in 3D LiDAR Maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[30] D. Chen, G. Baatz, K. Köser, S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzezczuk. City-scale landmark identification on mobile devices. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 737–744. IEEE, 2011.

[31] S. Chen, Y. Li, and N. M. Kwok. Active vision in robotic systems: A survey of recent developments. *Intl. Journal of Robotics Research (IJRR)*, 30(11):1343–1377, 2011.

[32] Z. Chen, O. Lam, A. Jacobson, and M.Milford. Convolutional neural network-based place recognition. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, 2014.

[33] W. Churchill and P. Newman. Experience-Based Navigation for Long-Term Localisation. *Intl. Journal of Robotics Research (IJRR)*, 2013.

[34] M. Cummins and P. Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[35] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[36] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 29, 2007.

[37] J.M.M. Dequaire, C.H. Tong, W. Churchill, and I. Posner. Off the Beaten Track: Predicting Localisation Performance in Visual Teach and Repeat. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[38] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[39] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C.C. Lerma. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[40] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[41] A. Fischer, T. Kolbe, A. Lang, F.and Cremers, W. Förstner, L. Plümer, and V. Steinhage. Extracting buildings from aerial images using hierarchical aggregation in 2d and 3d. *Computer Vision and Image Understanding*, 72(2):185–203, 1998.

[42] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 1981.

[43] G. Floros, B. van der Zander, and B Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[44] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 281–305. Interlaken, 1987.

[45] W. Förstner and B. Wrobel. *Photogrammetric Computer Vision*, chapter Robust estimation and outlier detection, pages 141–159. Springer Verlag, 2016.

[46] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In *Sequential Monte Carlo methods in practice*, pages 401–428. Springer, 2001.

[47] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J.M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, pages 1–27, 2012.

[48] P.T. Furgale and T.D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics (JFR)*, 27:534–560, 2010.

[49] D. Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. on Robotics (TRO)*, 28(5):1188–1197, Oct 2012.

[50] M. Gehrig, E. Stumm, T. Hinzmann, and R. Siegwart. Visual Place Recognition with Probabilistic Voting. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[51] M. Gerke. Using horizontal and vertical building structure to constrain indirect sensor orientation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 66(3):307–316, 2011.

[52] A. Gil, O. Reinoso, O. Martínez-Mozos, C. Stachniss, and W. Burgard. Improving Data Association in Vision-based SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[53] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[54] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. Openfabmap: An open source toolbox for appearance-based loop closure detection. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4730–4735, 2012.

[55] A.J. Glover, W.P. Maddern, M. Milford, and G.F. Wyeth. FAB-MAP + RatSLAM: Appearance-based slam for multiple times of day. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3507–3512, 2010.

[56] R. Gomez-Ojeda, M. Lopez-Antequera, N. Petkov, and J. Gonzalez-Jimenez. Training a Convolutional Neural Network for Appearance-Invariant Place Recognition. *arXiv preprint*, 2015.

[57] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[58] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, Anchorage, Alaska, 2010.

[59] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear Constraint Network Optimization for Efficient Map Learning. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 10(3):428–439, 2009.

[60] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Intl. Symp. on Computer Intelligence in Robotics and Automation (CIRA)*, pages 318–325, 2000.

[61] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in slam. In *Proc. of the Intl. Symposium on Robotic Research (ISRR)*, pages 421–431, Siena, Italy, 2003.

[62] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[63] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[64] M. Hentschel and B. Wagner. Autonomous robot navigation based on open-streetmap geodata. In *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 2010.

[65] M. Hentschel, O. Wulf, and B. Wagner. A gps and laser-based localization for urban and non-urban outdoor environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 149–154, 2008.

[66] https://mapstreetview.com.

[67] https://www.openstreetmap.org.

[68] M. Huber, W. Schickler, S. Hinz, and A. Baumgartner. Fusion of lidar data and aerial imagery for automatic reconstruction of building surfaces. In *Remote Sensing and Data Fusion over Urban Areas, 2003. 2nd GRSS/ISPRS Joint Workshop on*, pages 82–86, May 2003.

[69] E. Johns and G.-Z. Yang. Feature Co-occurrence Maps: Appearance-Based Localisation Throughout the Day. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[70] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–II. IEEE, 2004.

[71] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2015.

[72] Ayoung Kim and Ryan M. Eustice. Active visual slam for robotic area coverage: Theory and experiment. *Intl. Journal of Robotics Research (IJRR)*, 34(4-5):457–475, April 2015.

[73] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 748–761. Springer, 2010.

[74] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Sparse pose adjustment for 2d mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 22 – 29, 2010.

[75] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[76] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613, 2011.

[77] R. Kümmerle, B. Steder, C. Dornhege, G. Kleiner, A.and Grisetti, and W. Burgard. Large scale graph-based slam using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.

[78] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time. *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[79] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

[80] C. Linegar, W. Churchill, and P. Newman. Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[81] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999.

[82] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Intl. Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[83] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[84] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[85] A.L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza. Mav urban localization from google street view data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3979–3986, 2013.

[86] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1996.

[87] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 901–906. IEEE, 2014.

[88] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant-time efficient stereo slam system. In *BMVC*, pages 1–11, 2009.

[89] T. Middelberg, S.and Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 268–283. Springer, 2014.

[90] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 27(10):1615–1630, Oct 2005.

[91] M. Milford. Vision-based place recognition: how low can you go? *Intl. Journal of Robotics Research (IJRR)*, 32(7):766–789, 2013.

[92] M. Milford and G.F. Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[93] D. Mishkin, M. Perdoch, and J. Matas. Place recognition with wxbs retrieval. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[94] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 36, 2014.

[95] R. Mur-Artal and J.D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. on Robotics (TRO)*, 2017.

[96] A. Murillo and J. Kosecka. Experiments in place recognition using gist panoramas. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2196–2203. IEEE, 2009.

[97] L. Nardi and C. Stachniss. Uncertainty-Aware Path Planning for Navigation on Road Networks Using Augmented MDPs . In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[98] T. Naseer, M. Ruhnke, L. Spinello, C. Stachniss, and W. Burgard. Robust Visual SLAM Across Seasons. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[99] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Robust Visual Robot Localization Across Seasons using Network Flows. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 2014.

[100] P. Neubert, S. Schubert, and P. Protzel. Exploiting intra database similarities for selection of place recognition candidates in changing environments. In *Proc. of the CVPR Workshop on Visual Place Recognition in Changing Environments*, 2015.

[101] P. Neubert, N. Sunderhauf, and P. Protzel. Appearance Change Prediction for Long-Term Navigation Across Seasons. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2013.

[102] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[103] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[104] G.M. Pascoe, W. Maddern, A. Stewart, and P. Newman. FARLAP: Fast Robust Localisation Using Appearance Priors. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[105] O. Pink, F. Moosmann, and A. Bachmann. Visual features for vehicle localization and ego-motion estimation. In *IEEE Intelligent Vehicles Symposium*, pages 254–260, 2009.

[106] N. Radwan, A. Valada, and W. Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters (RA-L)*, September 2018.

[107] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1508–1511, October 2005.

[108] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, volume 1, pages 430–443, May 2006.

[109] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation – robot motion with uncertainty. In *Proceedings of the AAAI Fall Symposium: Planning with POMDPs*, Stanford, CA, USA, 1998.

[110] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[111] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard. Localization on openstreetmap data using a 3d laser scanner. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5260–5265, 2015.

[112] S. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[113] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. HJ Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359, 2013.

[114] T. Sattler, M. Havlena, F. Radenovic, K. Schindler, and M. Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2102–2110, 2015.

[115] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 667–674. IEEE, 2011.

[116] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2007.

[117] D. Schlegel and G. Grisetti. Visual Localization and Loop Closing Using Decision Trees and Binary Features. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[118] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Intl. Journal of Robotics Research (IJRR)*, 21(8):735–758, 2002.

[119] P. Sermanet, D. Eigen, Z. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *Intl. Conf. on Learning Representations (ICLR)*, 2014.

[120] J. Shi and C. Tomasi. Good features to track. Technical report, Cornell University, 1993.

[121] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, abs/1409.1556, 2014.

[122] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 1470–1477 vol.2, Oct 2003.

[123] C. Stachniss. *Springer Handbook of Robotics*, chapter Simultaneous Localization and Mapping. Springer, 2016.

[124] C. Stachniss and W. Burgard. Mobile Robot Mapping and Localization in Non-Static Environments. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1324–1329, Pittsburgh, PA, USA, 2005.

[125] E. Stumm, C. Mei, and S. Lacroix. Probabilistic place recognition with covisibility maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4158–4163. IEEE, 2013.

[126] E. Stumm, C. Mei, S. Lacroix, and M. Chli. Location Graphs for Visual Place Recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[127] N. Suenderhauf, T. Pham, Y. Latif, M.J. Milford, and I. Reid. Meaningful Maps with Object-Oriented Semantic Mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[128] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P.I. Corke, G. Wyeth, B. Upcroft, and M. Milford. Place categorization and semantic mapping on a mobile robot. *arXiv preprint*, abs/1507.02428, 2015.

[129] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1879–1884, 2012.

[130] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford. On the performance of convnet features for place recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4297–4304. IEEE, 2015.

[131] N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place Recognition with ConvNet Landmarks : Viewpoint-Robust, Condition-Robust, Training-Free. *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[132] S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *Intl. Journal of Robotics Research (IJRR)*, 25(5-6):403, 2006.

[133] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1808–1817, 2015.

[134] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. Visual place recognition with repetitive structures. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 883–890, 2013.

[135] J. Unger, F. Rottensteiner, and C. Heipke. Integration of a generalised building model into the pose estimation of uas images. *ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B1:1057–1064, 2016.

[136] C. Valgren and A.J. Lilienthal. SIFT, SURF & Seasons: Appearance-Based Long-Term Localization in Outdoor Environments. *Journal on Robotics and Autonomous Systems (RAS)*, 85(2):149–156, 2010.

[137] J. Velez, G. Hemann, A. S Huang, I. Posner, and N. Roy. Planning to perceive: Exploiting mobility for robust object detection. In *Proc. of the Intl. Conf. on Image Analysis and Processing (ICIAP)*, pages 266–273, 2011.

[138] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Efficient and Effective Matching of Image Sequences Under Substantial Appearance Changes Exploiting GPS Prior. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[139] O. Vysotska and C. Stachniss. Exploiting building information from publicly available maps in graph-based slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[140] O. Vysotska and C. Stachniss. Lazy Data Association for Image Sequences Matching under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 2016.

[141] O. Vysotska and C. Stachniss. Improving slam by exploiting building information from publicly available maps and localization priors. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, 85(1):53–65, 2017.

[142] O. Vysotska and C. Stachniss. Relocalization under substantial appearance changes using hashing. In *Proc. of the IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2017.

[143] O. Vysotska and C. Stachniss. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[144] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, pages 1753–1760, 2009.

[145] O. Wulf, K. O Arras, H. I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4204–4209, 2004.

[146] C. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 391–405. Springer, 2014.