Vehicle Ego-Localization using Monocular Vision

David Robert Wong

Abstract

Rapid advances in sensing technologies and information science have brought the goal of replacing human drivers with autonomous systems within reach. An essential element of many autonomous and intelligent vehicle systems is the self-localization, or ego-localization, of the vehicle. Localization has always been a fundamental part of transportation. A map is required for any form of travel, enabling us to determine where we are and how to get to where we want to go. While there are a wide variety of sensors that may be used for mapping and localization, the simple configuration of a single camera is one of the most appealing.

There are many factors which must be considered when designing vehicle localization systems. Inexpensive sensors which can be easily deployed in production vehicles, and compact database map sizes which can be streamed or saved within the vehicle navigation computer, are two important advantages. This leads us to the sensor choice for vehicle localization in this thesis —a monocular camera. Databases can be captured using sophisticated mapping hardware, but reliable performance in the localization phase with a minimum number of simple sensors is central to the research presented in this thesis.

For ego-localization using a monocular camera, there are three levels of localization which can be considered; topological localization, topometric localization, and direct metric localization. This thesis proposes three localization methods which aim to address each of these.

The first research topic that is presented in this thesis investigates a topological localization methodology using a novel image matching technique. This method provides a map-relative position estimation. In this research, "feature-scale tracklets" are proposed to create an image matching technique that selects the corresponding image from a database sequence based on the scale of matched features. Experimental results show how comparison of feature-scales is an effective method for image matching. The localization performance of the proposed method is analyzed and compared to comparative methods using two datasets.

The second research topic investigates visual topometric localization by expanding on the method proposed in the first research topic. This solves the problem of determining the metric location of a query frame within the continuous plane of a coordinate system. The linear change of feature-scale with capture position is used to perform feature-scale regression. This creates regression coefficients for each feature-scale tracklet that parametrize the relationship between capture position and feature scale, allowing interpolation of a query image's capture position based on feature scale. A Bayes filter is used to provide a location estimate from the individual matched query feature measurements. Experimental results show a decimeter level of localization accuracy. A discussion on the performance of the system compared to comparative methods is provided.

The third research topic uses a continuous 3D map made up of voxel data, which is constructed using LIDAR, to perform direct metric localization with camera pose estimation covering a full six degrees of freedom. Direct metric localization is performed relative to the voxel map by using mutually shared edge information between query images and rendered views of the voxel map. This allows accurate pose estimation and 3D localization of query frames. The experimental results show that even a compact voxel map, containing only edge information, can be used for precise direct metric localization. An analysis of the performance of the proposed method and its use with a speed measurement is provided, along with a discussion on outstanding issues and considerations for actual implementation.

In summary, this thesis presents methods for three localization methods that perform vision-based localization at different levels of precision —topological localization, topometric localization, and direct metric localization. Chapter 1 provides an introduction and background to this research. In Chapter 2, related research on ego-localization is presented, followed by detailed descriptions of the proposed solutions to topological localization, topometric localization, and direct metric localization in Chapter 3, Chapter 4, and Chapter 5 respectively. Finally, in Chapter 6, this thesis is summarized and further development and applications are discussed.

Acknowledgments

This dissertation is formally submitted for fulfilling partial requirements for the degree of Doctor of Information Science from Nagoya University. This work would not have been possible without the help of many people to whom I owe the sincerest gratitude.

I would first like thank to Prof. Dr Hiroshi Murase, not only for supervising this research, but also for accepting me into his laboratory and for all of his kind help and support throughout my time in Japan.

I would also like to express my gratitude to Prof. Dr Daisuke Deguchi for his supervision and assistance in this research. His input and insight provided the direction for this research and I am extremely grateful for his contributions.

I would like to thank Prof. Dr Ichiro Ide, who has meticulously edited every paper I have written while at the Murase Laboratory. His assistance throughout this research has been invaluable.

I thank Prof. Dr Kensaku Mori for his feedback and insightful comments during the review process.

I would also like to thank Dr Yasutomo Kawanishi for providing advice and feedback regarding research and publications.

Special thanks go to all members of the Murase Laboratory, past and present, who have helped me throughout this journey. The secretaries, Mrs Hiromi Tanaka and Mrs Fumiyo Kaba, have always been a great help with the documents and official procedures that I have required assistance with. I am grateful to Dr Keisuke Doman, Dr Haruya Kyutoku, and Brahmastro Kresnaraman for their support and help. I would like to thank all of the students, of whom there are who are too many to name one by one, for their friendship.

I thank the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) that has supported me in the form of a Monbukagakusho scholarship throughout my study as a research student (2012–2013), and PhD student (2013–2016).

Finally I would like to thank my family. Despite living in countries far away, they have always provided support and encouragement for my research.

Contents

List of Figures

List of Tables

Abbreviations

2D	**2 D**imensional
3D	**3 D**imensional
A-KAZE	**A**ccelerated-**KAZE**
BRIEF	**B**inary **R**obust **I**ndependent **E**lementary **F**eatures
BRISK	**B**inary **R**obust **I**ndependent **S**calable **K**eypoints
CT	**C**omputed **T**omography
DOF	**D**egrees **O**f **F**reedom
DoG	**D**ifference of **G**aussians
DP	**D**ynamic **P**rogramming
DTW	**D**ynamic **T**ime **W**arping
EM	**E**xpectation **M**aximization
FAST	**F**eatures from **A**ccelerated **S**egment **T**est
FREAK	**F**ast **RE**tin**A** **K**eypoints
GLONASS	**GLO**bal **NA**vigation **S**atellite **S**ystem
GMM	**G**aussian **M**ixture **M**odel
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
ICP	**I**terative **C**losest **P**oint
IMU	**I**nertial **M**easurement **U**nit
LIDAR	**LI**ght **D**etection **A**nd **R**anging
MMS	**M**obile **M**apping **S**ystem
MRI	**M**agnetic **R**esonance **I**maging

NDT	**N**ormal **D**istributions **T**ransform
NID	**N**ormalized **I**nformation **D**istance
ORB	**O**riented FAST and **R**otated **B**RIEF
PDF	**P**robability **D**ensity **F**unction
RADAR	**RA**dio **D**etection **A**nd **R**anging
RANSAC	**RAN**dom **SA**mple **C**onsensus
RMS	**R**oot **M**ean**S**quare
RTK-GPS	**R**eal **T**ime **K**inematic-**G**lobal **P**ositioning **S**ystem
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SIR	**S**ample **I**mportance **R**e-sampling
SURF	**S**peeded **U**p **R**obust **F**eatures
SSD	**S**um of **S**quared **D**istance
UKF	**U**nscented **K**alman **F**ilter
WISURF	**W**hole **I**mage **SURF**

For my parents

Chapter 1

Introduction

Rapid advances in sensing technologies and information science have brought the goal of replacing human drivers with autonomous systems within reach. The ability of an intelligent vehicle system to determine its own location relative to a known map is known as *ego-localization*. Reliable and accurate ego-localization is not only important for in-car navigation systems, but it is also a key component for many of the emerging vehicle technologies that automate driving tasks. In-car navigation systems are standard equipment on many modern automobiles, and usually use a Global Navigation Satellite System (GNSS) such as the Global Positioning System (GPS) for the purpose of localization. While consumer models are effective when the vehicle has a clear view of the sky, localization performance is reduced in typical urban environments. Camera-based localization systems have the potential to provide accurate localization using inexpensive sensors, and unlike GPS, do not need a clear view of the sky. This thesis discusses the use of a single camera to perform automotive ego-localization relative to a pre-constructed database.

In this chapter, the purpose and background of the research are presented in Section 1.1, followed by an introduction to visual localization techniques in Section 1.2. Section 1.3 outlines the research topics that are presented in this thesis, and the thesis structure is summarized in Section 1.4.

1.1 Background

This section presents the aim of this research and the background of vehicle localization for automobiles. In this thesis, *localization* is defined as the determination of position within a known map or a global coordinate system. This thesis deals with the problem of *ego-localization*, where the vehicle is able to localize itself without manual intervention.

1.1.1 Aim of this research

The research presented in this thesis aims to contribute to the realization of a reliable, accurate mapping and ego-localization that can be cheaply deployed on production vehicles, as a central component of autonomous driving systems.

The potential benefits to be made from introducing automation to the task of individual transportation are great. Advances in vehicle safety over the last few decades have already made a large difference to the number of traffic accidents. As well as the development of passive safety design, the introduction of active safety systems such as lane-departure warning, adaptive cruise control, automated emergency braking and so on have also become more commonplace. The number of traffic casualties per number of registered vehicles has already drastically decreased over the last thirty years —for example, in Japan, the road toll has fallen from 138 per million registered vehicles in 1985 to 53 in 2015 [1, 2]. This is a global trend, as displayed in Figure 1.1, which shows the steady decrease in the road toll per million registered vehicles within G7 countries over a 20 year period [2]. Despite the road toll trending downward, 4,859 people were still killed on Japanese roads during 2015 [1, 2]. As robotic systems begin to achieve operational reliability that can exceed human ability, increased automation of driving systems has the potential to further reduce the number of traffic accidents, as autonomous vehicles are not subject to human error and fatigue —the root cause of many accidents. They are also potentially capable of instantaneous communication with other vehicles and infrastructure, improving

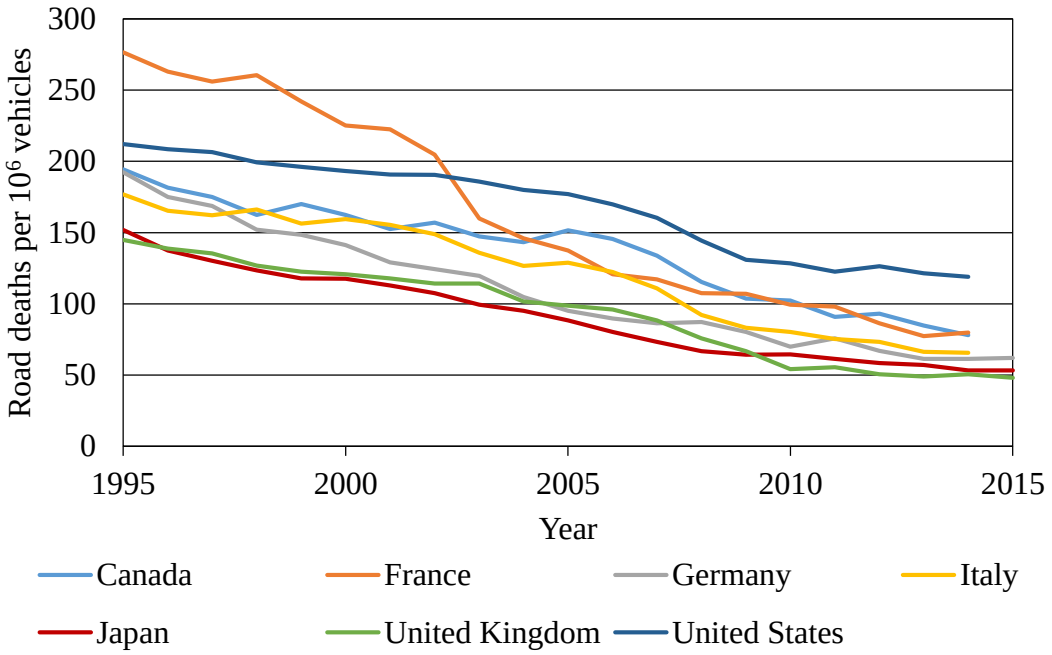


Figure 1.1: The change in the road toll between 1995 and 2015 in G7 member countries. Data source: OECD (2017) [2].

safety and providing a means for cooperative traffic planning to reduce traffic congestion.

Autonomous driving could provide mobility to those who can not, or do not wish to, drive manually. The increasing population of elderly citizens in Japan has led to an increase in the accident rate within the 65 years old and over age group [3, 4], leading to the promotion of voluntarily license surrender for elderly drivers [4]. Self-driving cars could bring safe mobility to an increasing percentage of the aging population.

1.1.2 Localization of automobiles

Localization has always been a fundamental part of transportation. A map is essential for any form of travel, enabling us to determine where we are and how to get to where we want to go. Vehicles used for personal transportation have always required localization, which for most of the history of the automobile has been achieved using paper maps. Even now, localization is often performed using paper maps together

with a human driver (or assistant) who detects features, such as road shape, landmarks, and road names from road signs in the current environment, and matches them to corresponding features displayed on the paper map. Until the introduction of GPS, manual localization relative to paper maps was the usual method for navigation in areas that were not committed to memory.

1.1.2.1 GPS —Changing the way we localize

There are several satellite navigation systems, both regional (which only cover limited terrestrial areas) and global (often referred to as GNSS, with world-wide satellite coverage). All work on a similar principle. Radio signals are transmitted by satellites that orbit the Earth, and are detected by land-based receivers. Every satellite has an accurate atomic clock on-board, and each transmission digitally encodes the orbital location of the satellite and the precise time that the transmission was made. A receiver device can determine its distance from each satellite based on the time each signal was received compared to the time-stamp of the signal. It can then triangulate its position in terms of longitude, latitude and altitude based on signals received from at least three satellites in direct line of sight. The receiver also requires an accurate clock to make these calculations. Since including an atomic clock in a compact and inexpensive receiver is not feasible, in practice a fourth satellite signal is required to enable calculation of the receiver's current time and position. The mechanism of a satellite navigation system is shown in Figure 1.2.

There are three widely used GNSS; the United Sates' GPS, Russia's GLObal NAvigation Satellite System (GLONASS), and the European Union's Galileo system. The fist publicly available GNSS was GPS, and it remains the most widely used satellite navigation system worldwide.

GPS was originally designed in the United States in the 1960s for military applications, allowing navy ships to read signals from orbiting satellites and fix their positions every hour [5, 6]. As the network of satellites increased, the system reached a level of coverage that meant it could assist civilian transportation. To begin with,

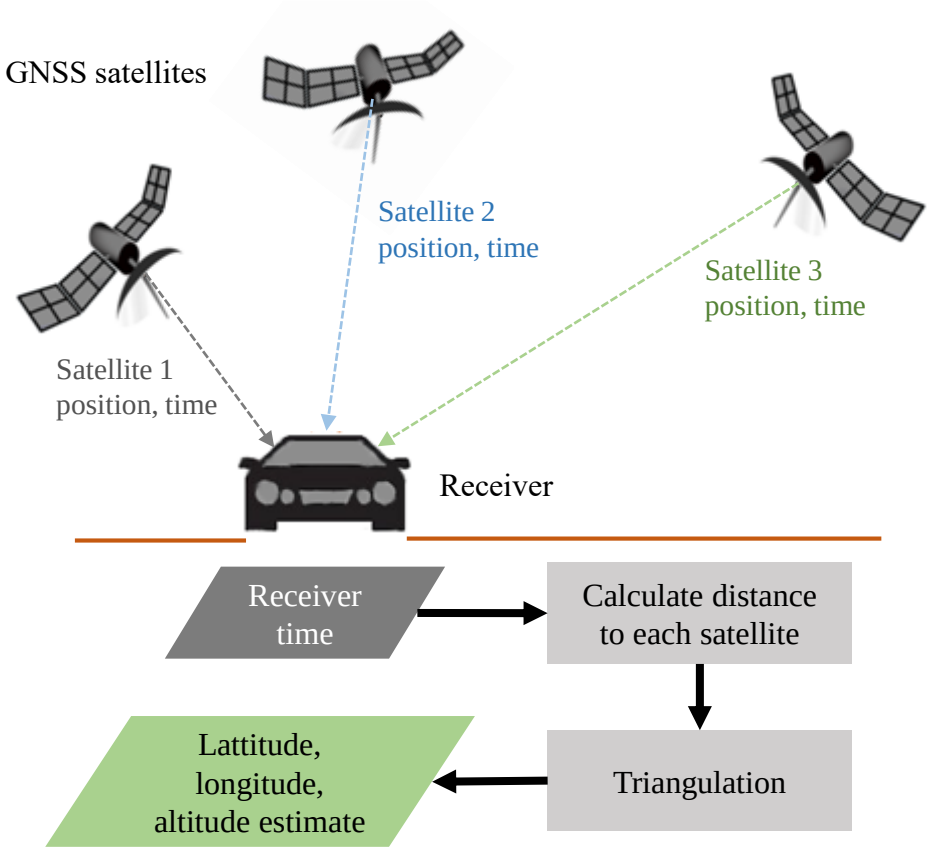


Figure 1.2: GNSS location estimation process.

GPS was used by aircraft and ships to avoid entering restricted or dangerous foreign territory in error, motivated by an event where a passenger aircraft was shot down after wandering off course into Soviet Union airspace in 1983. Until 2000, the US Department of Defense purposefully degraded the precision of the GPS system for national security reasons. After this, the sudden improvement in the accuracy of the system, combined with advances in compact and inexpensive receiver technology, led to GPS being rapidly adopted by many industries. Soon afterwards, in-car navigation systems came on the market and have been the main method for ego-localization in automobiles (besides the use of paper maps) ever since.

However, GPS does not completely solve the ego-localization problem in automobiles. Advances in computer and sensor technology and information science are

driving automation in many industries, and this is spreading to personal motor vehicles as well, increasing the demand for more accurate localization methodologies. Technologies that automate driving tasks are gradually appearing in production automobiles, but there are still many issues that remain to be solved in order to achieve true autonomous driving.

Both assisted driving technologies and fully autonomous cars rely on accurate information about the current location of the vehicle. A human driver may use an in-car navigation system to help them determine how to get to their destination, but an autonomous system may also require ego-localization for determining where it is safe to perform certain actions such as turning or lane changes, what speed to drive at, and so on. GPS may provide coarse localization information, but except with the use of specialized Real Time Kinematic GPS (RTK-GPS) and augmentation with other sensors such as Inertial Measurement Units (IMUs) and vehicle odometry sensors, accurate localization is difficult. Such sensor suites are very expensive for production vehicles, and do not provide localization based on directly sensing the environment around the vehicle. Consumer GPS systems suitable for production vehicles may suffer from errors exceeding 5 m even in good conditions [7]. Tall buildings, overhead roads, tunnels or any other structures common in city driving that prevent a direct signal path to satellites can cause even higher errors or even complete failure to localize. GPS localization relies on a receiver interpreting signals from multiple overhead satellites, so when the signals are blocked entirely (signal shadowing [5]) or bounced and reflected between structures (multipath propagation [8]), erroneous localization readings occur. Insufficient satellites within direct view of the receiver will cause the system to fail to provide a localization estimate entirely. There are methods which attempt to overcome some of the positioning errors of GPS using the shape of the road as a prior map, which are discussed in more detail in Chapter 2. However, in general, a system which directly senses the environment and compares it to a pre-constructed map or database is essential for critical driving tasks that depend on ego-localization. Even the most accurate GPS systems will still fail where there is

no clear line-of-sight to satellites. Also, intelligent vehicle systems may need to respond directly to the environment, which requires direct sensing of the world around the vehicle rather than relying on position data provided by satellite triangulation only.

1.1.2.2 Alternative sensors

There are a variety of sensors available for the task of environment sensing in vehicles. The most common ones used in autonomous driving are LIght Detection And Ranging (LIDAR), RAdio Detection And Ranging (RADAR), and visible light cameras.

LIDAR sensors project laser light onto the environment, and detect the backscattered light. By measuring the time between the light emission and detection of the reflection, a 3D point cloud that models the environment structure can be constructed. The major downside to LIDAR is the large expense and physical size of the sensors, which for vehicle use are typically roof-mounted. They are not yet suitable for production vehicles because of this. However, there is a lot of research on using LIDAR for both map creation and ego-localization, for which a brief overview is provided in Chapter 2.

RADAR sensors are much less expensive and easier to integrate into vehicle body design. They work on a similar principle to LIDAR sensors, but use electromagnetic waves in the radio spectrum rather than the visible (or close to visible) spectrum used by LIDAR. The longer wavelength means that RADAR sensors can not provide the accuracy or angular resolution that LIDAR can, and the detection of small objects and fine structure is much more difficult. There are a number of recent research projects that use RADAR for ego-localization, some of which are described in Chapter 2.

Visible light cameras are passive sensors, so they do not rely on detecting light emitted from the sensor itself, but measure the ambient light reflected from the environment. This makes the images that they produce heavily dependent on the lighting conditions, and means that they can not measure the distance to the environment directly, unlike RADAR and LIDAR. However, they are very inexpensive and are already integrated into many consumer vehicles for the purpose of lane-keeping, pedestrian detection, and even insurance purposes. They potentially provide much greater resolution and more information about the environment than LIDAR and RADAR while being fast and reliable to operate. Because cameras capture a lot of information about the environment, visual databases for localization are typically very large. Changes in lighting and weather also change the appearance of a scene, adding a further challenge.

This thesis focuses on the use of cameras for visual localization. Cameras were chosen because they are sensors that are already present in many production vehicles. A visual map is an intuitive one and services such as Google Street View [9] have illustrated that large-scale visual databases are now possible. An effective visual localization method potentially provides inexpensive and rapid deployment to modern vehicle systems, and also a more compact database structure. The following section provides an introduction to visual ego-localization using cameras.

Table 1.1 provides a comparison of the different sensor types for vehicle localization. While the camera is not definitively the best sensor for all criteria, its low cost coupled with potentially high localization accuracy make it a very attractive option. In real-world localization systems, the fusion of multiple sensor types promises the most reliable configuration. This is particularly important when considering localization throughout a variety of environments. Localization in a particular environment might be more effective with a particular sensor, which is not as useful in a different place or situation. For example, the localization problem in urban streets is very different from that in rural roads. Urban environments provide more close-by scene structure and distinctive visual features which make cameras potentially effective for localization. However, in a rural setting, visual scenes may appear very self-similar

Table 1.1: Summary of typical properties of sensor types for vehicle localization.

	GNSS	**LIDAR**	**RADAR**	**Camera**
Localization accuracy	Low–Med.	High	Low	Low–High
Sensor expense	Low–Med.	High	Low	Low
Database size	N/A	High	Med.	Med.–High
Complexity	Low	High	Med.	Med.–High
Robustness to occlusion	Low	Low	Low	Med.
Robustness to weather	High	Low	Med.	Low
Robustness to lighting changes	High	High	High	Low

for long stretches of road. This makes it difficult for visual localization systems to discriminate between different areas of the map, potentially reducing localization accuracy. In such environments, the lack of tall man-made structures may also result in better GPS satellite visibility, so GPS may provide a stronger location estimate.

Since improving the performance and reliability of localization with cameras will help to improve the usability of inexpensive sensor suites, this thesis focuses on monocular vision for ego-localization. Unlike stereo methods, monocular vision systems do not require camera-to-camera calibration, and use the simplest and most inexpensive hardware for a vision system.

1.2 Visual ego-localization: Where are we?

Computer vision for localization is a very active area of research for automotive and robotics applications. Camera systems may be divided into those which use a single

Figure 1.3: Example monocular image from a vehicle-mounted camera.

camera, also called monocular vision, and those which use multiple cameras —either positioned at various places around the vehicle, or configured as stereo cameras. Stereo systems are able to be used to determine 3D information about the scene being measured, but calibration is a challenging problem. Some systems use multi-directional cameras for sensing a large field-of-view around the vehicle. More detail on automotive localization methods is presented in Chapter 2 and here the problem to be overcome by visual localization is discussed.

This research focuses on the use of a single (monocular) camera for localization. Although a single camera provides a more limited amount of information when compared to stereo or omni-directional cameras, the simplicity and lack of a need for re-calibration while in service make it an appealing configuration for localization in production vehicles. Figure 1.3 shows an image from a vehicle-mounted camera similar to the ones used in this research.

This thesis presents research on three localization types for differing applications, as follows:

1. *Topological localization* provides a location estimate within a discrete set of map points. It is the most general method, suitable for location estimation refinement from a noisy GPS reading. It is also computationally the simplest method.

2. *Topometric localization* provides a more refined, 2D or 3D position estimate in a continuous coordinate frame. It still uses a topological map made up of discrete locations.

3. *Direct metric localization* returns a complete 6DOF, precise position estimate in a continuous coordinate frame. It is the most expensive to compute and uses a larger, continuous database.

These three localization schemes provide different levels of localization, and may be used alone or together depending on the level of positioning precision required by the particular application. A topological localization method may provide a fast and general localization for autonomous vehicle systems that only require map-relative localization —for example, environment-adaptive pedestrian detectors that use a different classifier depending on the current location [10, 11], or for initialization of a more accurate metric localization technique. Topometric localization methods may be used for more precise positioning of the vehicle where metric distances to objects of known location are required, such as pedestrian crossings and the boundaries of speed limit restricted areas. Direct metric localization can add further precision where the exact position and vehicle pose are required —for example, in precise maneuvering around difficult road structures like expressway on-ramps, or parking buildings. Figure 1.4 gives a simple overview of the differences between the three localization types. Table 1.2 compares the advantages and disadvantages of each.

In the following sections, the concept of topological localization is introduced in Section 1.2.1 followed by topometric localization in Section 1.2.2 and direct metric localization in Section 1.2.3.

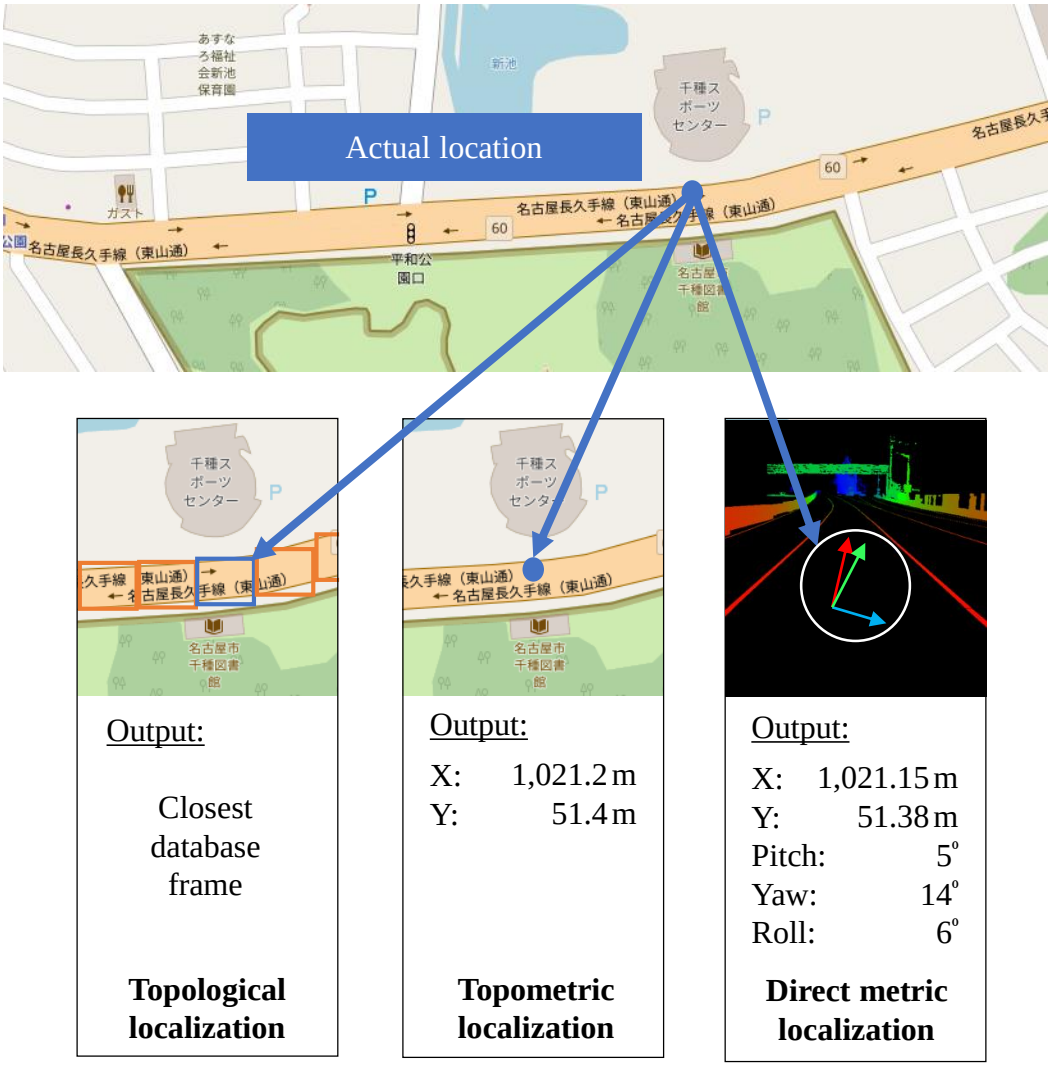


Figure 1.4: Topological, topometric, and direct metric localization. Map graphics: OpenStreetMap.

1.2.1 Topological localization

Generally, the most useful information for human drivers trying to navigate unknown areas is *map-relative* localization. That is, localization relative to a known map such as a paper street map, or a stored map in a navigation system. For this purpose, *topological* localization [12] is suitable. It provides a discrete set of locations, or frames, joined by paths to form a topological map. Localization is performed by matching the current location to one of the discrete map frames, making use of the path information which shows how the nodes are spatially ordered.

Table 1.2: Comparison of the localization types discussed in this chapter.

Localization type	Localization precision	Computational complexity	Scalability to large maps
Topological	Lower	Lower	Higher
Topometric	↓	↓	↑
Direct metric	Higher	Higher	Lower

The concept of topological localization originated with the work of Kuipers [13], in the field of artificial intelligence. It was then adopted early on in robotics localization research [14–16].

A visual topological localization system will use a pre-constructed database which contains map nodes in the form of visual information frames. The discrete locations within the database are linked through paths to create a topological map. The input image from a vehicle-mounted camera is used to determine the frame within the database which is the most visually similar. This provides an estimate of the current vehicle location within the map.

Topological localization requires some form of cost function to determine the most visually similar part of the database to the current query frame. The localization process may use matching of sequences of images along paths in the topological map rather than individual frames. Topological localization methods have the advantage that they can usually scale to a relatively large database, so that general localization can be performed with only a coarse initial estimate. This is because of the discrete nature of the database used.

1.2.2 Topometric localization

For a human navigator, topological localization is usually sufficient. By recognizing where the topological localization system has placed the vehicle within the map based on visual similarity, the driver can understand how to relate the surrounding environment to the map and use this information to navigate. The human driver handles the finer details of control, such as exactly where to turn, by direct visual observation of the environment. For autonomous driving systems, this may be inadequate. While environment sensing and interaction capability using vision are improving, there still exists a need to precisely position the vehicle such that exact information about a particular location can be used to assist with driving the vehicle. To this end, *topometric* localization [17] is necessary. Topometric localization is an extension of topological localization, also using a database of discrete, path-connected nodes, but determines the exact location of the vehicle relative to a reference frame —for example, the Universal Transverse Mercator (UTM), a 2D Cartesian coordinate system of the Earth's surface. The reference frame may be an arbitrary reference frame for the particular map being used, and may define a continuous position property along the paths of a topological map.

A topological space equipped with continuous local metric information for use in vehicle localization was first referred to as *topometric* by Badino *et al*. [17], but the term was used previously with the same meaning in the field of mathematical logic [18]. The concept was also developed earlier in robotic localization [19, 20], with the terminology "topological" and "metric" used uncombined.

Topometric localization determines the position of the vehicle in a quantitative way, allowing the exact estimation of the distance to objects and features within the map which are of known coordinates in a reference frame. The simplest form of topometric localization using vision can be performed by using a topological localization method and then applying known locations of the visual information stored in the database. Image matching of a query frame to a database frame will then provide an estimate of the current query location. For useful accuracy, this requires small

spatial distances between the database frames, to provide sufficient positioning resolution. In this thesis, *database spacing* refers to the metric value of the spatial distance between database frames. Topometric localization can also be performed more accurately by calculating the position of a query frame relative to information in the visual database by applying the geometric constraints, provided by features matched between the query frame and database frames.

1.2.3 Direct metric localization

A thorough localization method is *direct metric* localization. Direct metric localization is performed by directly comparing the query camera image with database information. The database contains a continuous scene representation. The position (and often also pose) of a query camera is estimated by iteratively altering the estimation parameters to align the query image to the database scene representation through a dissimilarity cost function.

Direct metric localization has its roots in the field of direct image registration [21]. The goal of image registration is to transform one set of data into the coordinate system of another, thereby determining the spatial transformation between the two sets. This is a broad field with early research in the applications of remote sensing [22], and medical image analysis [23].

For visual direct metric localization, image alignment can be accomplished by rendering virtual images of the database for comparison with query images, or the query image information can be projected onto the database map. While potentially very accurate and robust to changes in viewpoint, conventional direct metric localization methods require large databases and are often very computationally intensive.

1.3 Research overview

Intelligent vehicle systems cover broad and varied fields of research. There are many different systems with different requirements for ego-localization. For example, while one system may emphasize exact localization accuracy over speed of operation, another may place more focus on reliable localization in the presence of dynamic objects. While it is extremely difficult to design a mapping and localization method which will be suitable for all automated driving systems, this thesis aims to present three methods which address different levels of the ego-localization problem.

All of the research that is presented in this thesis makes use of images from a single vehicle-mounted camera to achieve localization relative to a pre-constructed database. The pre-constructed database was captured in all cases by specialized hardware in the mapping stage, but the aim of this research is to present localization systems which require the bare minimum of hardware in the localization stage. Therefore, this research attempts to limit the dependency on multiple sensors for the capture and position estimation of query frames.

The first research topic that is presented in this thesis performs a topological localization using a novel image matching technique. This method provides a map-relative discrete position estimation, with an extension to provide a coarse topometric localization by using metric localization information from database frames. The second research topic performs visual topometric localization to position the vehicle by expanding on the method proposed in the first research topic, interpolating the query image capture position between database frames. The third research topic uses a different form of pre-constructed database, with a continuous 3D voxel map providing the basis for direct metric localization with full 6DOF camera-pose estimation.

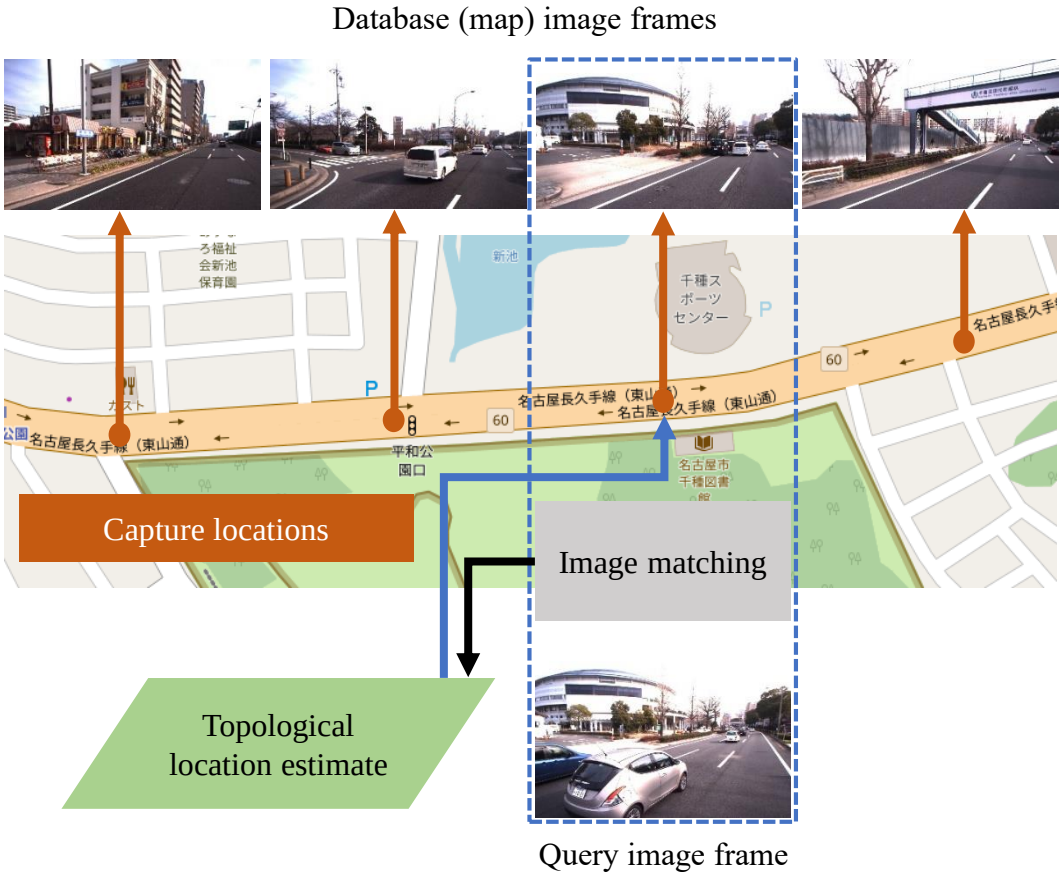


Figure 1.5: Topological localization from visual similarity. Map graphics: OpenStreetMap.

1.3.1 Research topic 1: Topological localization with feature-scale tracklets

This research presents a method in the form of topological localization that was introduced in Section 1.2.1. If database images have been captured along a route in the mapping stage, then by performing frame alignment of input image frames to the database image frames in the map, we can determine the vehicle's map position in the localization stage. The basic concept is illustrated in Figure 1.5.

The important component of this method is the image matching system for frame alignment. Where conventional methods will either use some form of template matching or whole-image descriptor matching [24–27], the proposed method makes use of scale-invariant image feature points to perform image matching. This method

extracts feature points from database images and pre-matches them, recording the change in scale of the features with their capture location. The strings of interconnected feature points are termed "feature-scale tracklets". Here, a "tracklet" is a string which tracks the changes in a property (in this case, feature scale) throughout corresponding features. The localization stage consists of matching query image frame features to the feature-scale tracklets, and then predicting the closest database image frame based on the scale differences of the matched features. This work often refers to the use of scale-invariant image features, and the reader is referred to Appendix A for details on the theory of image feature points and their extraction and matching.

In order to understand how feature scale can be used to match images, we must consider the constraints present in vehicle motion. Automotive localization deals with vehicles that travel along prescribed roads, with limited lateral motion relative to the lane centers. In addition, scale-invariant image features have a scale, or size property. This property increases as the capture location becomes closer to the feature's physical position. That is, the feature size as observed on the image plane of the camera is inversely proportional to the distance between the feature's real-world position and the focal point of the camera. Therefore, when the scales of a corresponding query and database feature are similar, their capture locations are close. This is the property used for image matching in this research. Many query image frame features are matched to the database tracklets, and each votes for the most likely database frame match.

The result is a robust image matching method which is effective at finding the closest database image location, even when significant scene changes have occurred —for example, under occlusion or lane changes. Chapter 3 describes this research topic in detail, with experimental results and analysis.

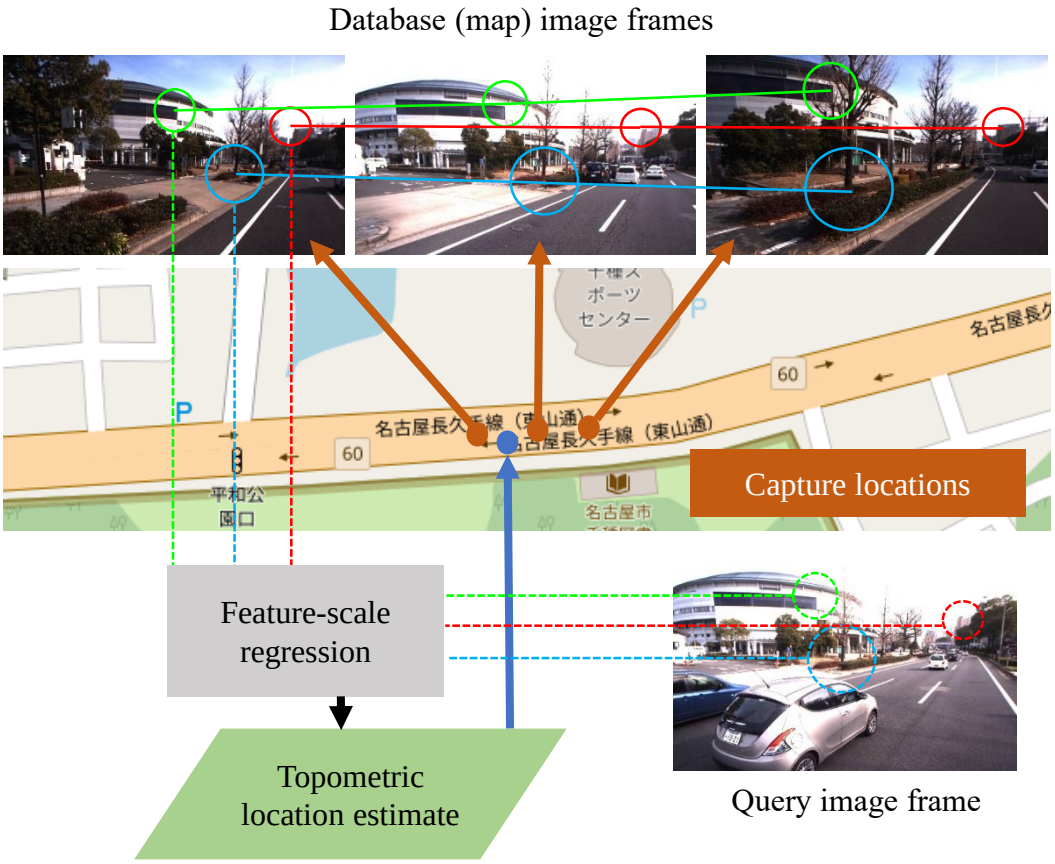


Figure 1.6: Topometric localization from feature-scale regression. Map graphics: OpenStreetMap.

1.3.2 Research topic 2: Topometric localization with feature-scale regression

Ego-localization for applications that require a precise metric position should not be limited by the spacing of the database frames in the pre-constructed map. Topometric localization using frame alignment is possible, but requires closely spaced database frames for high accuracy. This corresponds to a large database size. The method proposed in this research topic again uses the scale of extracted feature points to perform a regressive metric positioning. This is achieved by interpolating the query frame position between several database frames using the scale of extracted features. The concept is illustrated in Figure 1.6.

Methods for topometric localization using vision usually involve extracted feature points, as does this research. However, typically the detected feature-point locations in the image plane are used to determine the camera pose [28, 29]. These methods require the calculation of the essential matrix between frames [30]. Essential matrix calculation introduces significant complexity and also suffers from failed localization when too few features are matched, or the matched features result in poorly conditioned geometry [31, 32]. In this research, geometry calculations are avoided, and using the same feature-scale tracklets as the above topological localization method, feature-scale regression within each tracklet is performed. This allows the interpolation of a (metric) query capture position between the closest database frames of known capture location. In the construction of the database, for each feature-scale tracklet, a set of regression coefficients that describe the linear relationship between the feature scale and capture position within the tracklet are calculated. This allows each query feature that is matched to a database tracklet to provide a position estimate, based on its scale property. This research uses a Bayesian estimator in the form of a Kalman filter to combine the individual query feature position estimates. Bayesian estimation for vehicle position estimation is described in general in Appendix B.

This research shows how 2D topometric localization is possible using local feature points without the need to perform pose estimation from geometric constraints between matched query and database features. Chapter 4 describes this research topic in detail, with experimental results and analysis.

1.3.3 Research topic 3: Direct metric localization within sparse voxel maps

While the localization methods outlined above solve the ego-localization problem to either a discrete topological level or a 2D metric level, they do not perform precise positioning in terms of a 6DOF camera-pose estimate of the query frame. This can

be accomplished with the calculation of the essential matrix, which describes the relative pose between two cameras (i.e., query frame and database frame) or between a query camera and 3D feature points in the database whose positions have been calculated with pre-processing. However, this research proposes a different methodology to estimate 6DOF camera pose, and focuses instead on a method which does not use features at all, but performs direct metric localization relative to a 3D map. Conceptually, this method is very different from the feature-based methods of research topic 1 and research topic 2. However, the problem of ego-localization using monocular vision is potentially solved with both feature-based and direct methods. In applications where precise vehicle pose is required, for example augmented reality for driver assistance, direct metric localization is an interesting solution.

The increasing availability of inexpensive parallel processing power in computer processors and Graphics Processing Units (GPUs) has led to a number of direct metric localization techniques that render a dense 3D map for direct appearance comparison with input camera images [33–35]. In the field of medical imaging, a similar concept is used for the pose estimation of flexible endoscopes in surgical navigation [36], where renders of 3D scans are used for registration of endoscope images. Direct metric localization methods generally require a high level of detail in the rendering process, with dense 3D models leading to large database sizes.

The approach of this research topic employs direct metric localization techniques, based around a relatively compact map which contains much less information than most of the state-of-the-art methods. The map contains a series of points representing corners of voxels that make up a coarse occupancy grid, or sparse voxel map. The voxel map provides no color or texture information, so common techniques that use joint image entropies can not be easily applied to the image registration problem. Instead, this research uses an image gradient-based objective function that is minimized where mutual edges exist between rendered and real camera images. A non-linear Levenberg-Marquardt [37] optimization is then applied to determine the camera pose within the map. The proposed system does not use lighting in the map rendering, and only mutual edges are included in the cost function, allowing camera

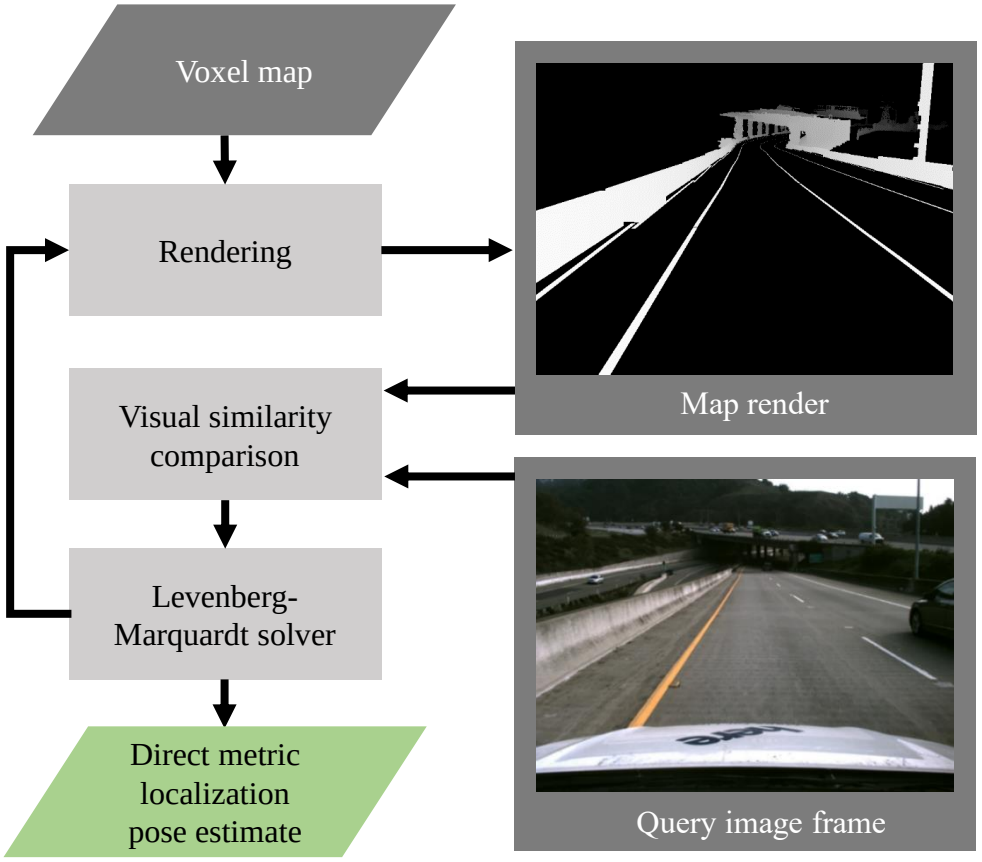


Figure 1.7: Direct metric localization from rendering a voxel map.

localization which is robust to lighting changes and the occlusions typically found in traffic environments. An overview of the system is illustrated in Figure 1.7, and Chapter 5 describes this research topic in detail, with experimental results and analysis.

1.4 Thesis structure

This thesis contains six chapters and two appendices. The relationships between the different parts of the thesis are visualized in Figure 1.8.

This chapter has discussed the background of the thesis and provided an overview of the three research topics which are proposed to overcome the visual ego-localization

problem. Chapter 2 gives a more thorough introduction to the body of existing research which relates to the methods and concepts included in this thesis, including a discussion on vehicle localization methods which are based on sensors other than cameras. Chapter 3 covers in detail the first research topic of this thesis: performing topological localization using an image matching method based on feature scale. Chapter 4 continues with the second research topic: performing topometric localization by position estimation using feature-scale regression. Chapter 5 presents the third research topic: direct metric visual localization using an alternative database consisting of a voxel occupancy grid. Finally in Chapter 6, this thesis is concluded with a summary of the presented localization methods, directions for future research, and closing remarks.

The first appendix, Appendix A, covers the background theory of feature scale extraction, which provides the basis for the first two research topics introduced in Chapter 3 and Chapter 4. The second appendix, Appendix B, gives a background on Bayesian estimation for automotive localization, which relates to the methods of the second and the third research topics introduced in Chapter 4 and Chapter 5.

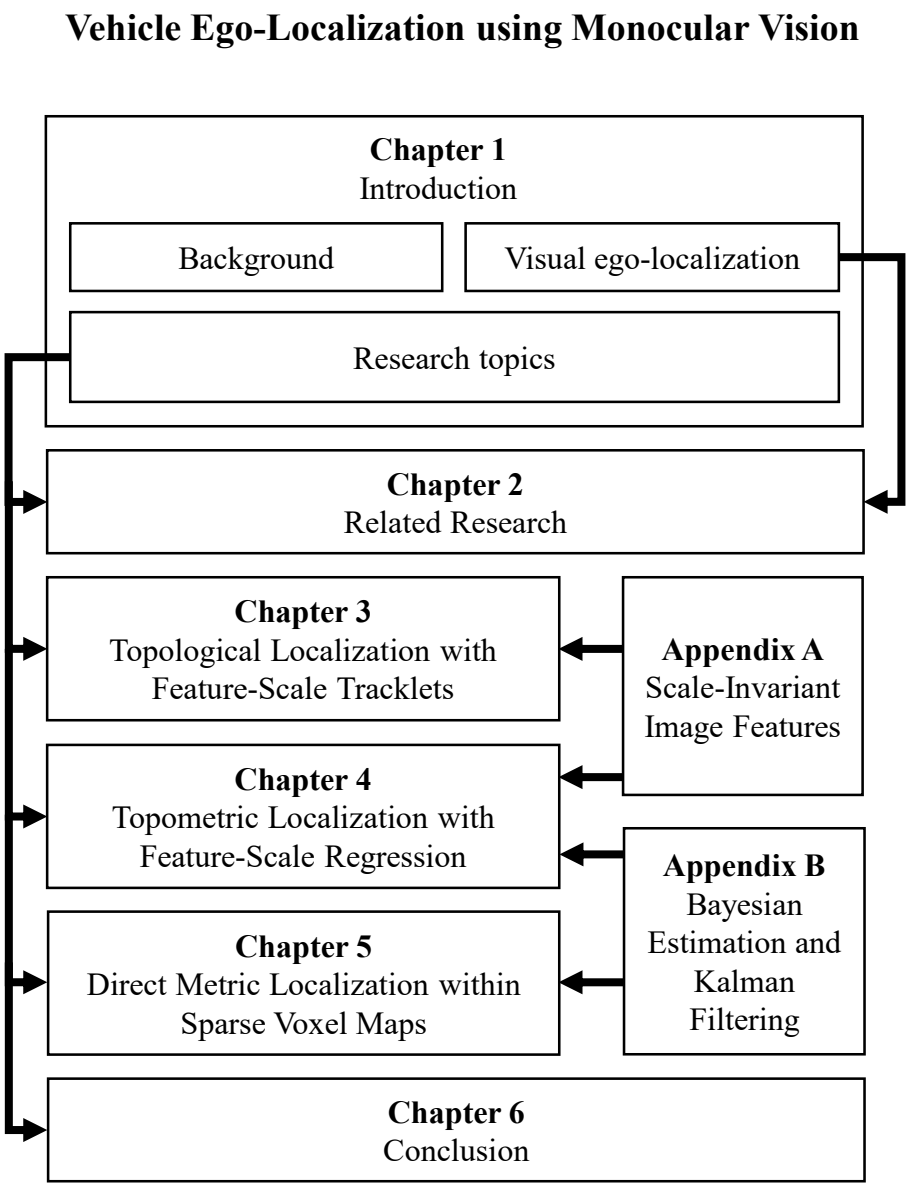


Figure 1.8: Overview of the chapters of this thesis.

Chapter 2

Related Research

Ego-localization is a very active area of research in automotive and robotics applications. There are many methods that have been proposed for localization in the robotics research community, usually referred to as Simultaneous Localization and Mapping (SLAM) [38] which aims to overcome the problem of ego-localization of a robot in an unknown environment. SLAM using cameras [39–42] is an active area of research. These methods construct a map for localization at the same time as positioning the robot. Ego-localization relative to a pre-constructed database is a similar problem. However, unlike SLAM, a map is not created on-the-fly, but built beforehand. The database can be constructed using a different capture platform from the one used on the vehicle to be localized, allowing a much more detailed database to be created. Ego-localization relative to a pre-constructed database is appropriate for vehicle localization on road networks which can be mapped in advance.

This chapter reviews some of the methods used for achieving localization using prior information. Here the literature review is not restricted to a certain type of map or sensor, but provides a general overview of localization types. Methods using the Global Navigation Satellite System (GNSS) are introduced in Section 2.1, and LIDAR systems are discussed in Section 2.2. In Section 2.3 a brief overview of RADAR systems is provided. Finally, the works most closely related to the research of this thesis, localization using cameras, are discussed in Section 2.4.

2.1 Localization with a Global Navigation Satellite System

The use of a GNSS is a common for ego-localization, with the ubiquitous and inexpensive Global Positioning System (GPS) receiver often being a central component in navigation systems. As was discussed in Chapter 1, GPS provides metric localization which suffers from poor accuracy when satellite signals are blocked or reflected [43]. However, there are methods which aim to combine the metric output of a GPS receiver with a pre-constructed map to provide a form of topological localization, with a notable example being the work of Drevelle *et al*. [44, 45]. This method works by tightly coupling a GPS receiver with a 3D map of the drivable area. The map allows the possible location of the vehicle to be subjected to geometric constraints, which are combined with GPS measurements to produce a position hypothesis (or multiple hypotheses where poor satellite visibility causes measurement ambiguity). While the use of possible road-path constraints may be effective for overcoming poor GPS results in urban environments, with Drevelle *et al*. [44] citing 6.5 m accuracy over 95% of the tested dataset, more accurate positioning is desirable for automated driving.

2.2 Localization with LIDAR

Another sensor mentioned in Chapter 1 was LIDAR. There are many implementations of vehicle localization using LIDAR for both map construction and position estimation [46–49]. LIDAR mapping methods combine rotating laser scans to make a point cloud or occupancy grid map, and localization is achieved by aligning 3D scans from the vehicle LIDAR unit with the map. In the work of Levinson *et al*. [46, 47], the LIDAR map consists of 2D scans of the road surface including infrared ground reflectivity (which is also measured from the reflected light of a LIDAR sensor). Localization is then performed by extracting the ground plane from query scans, and

then finding the map location with the maximum correlation to the query scans. A particle filter [50] is used as an estimator. The authors reason that this method is robust to changes in environment dynamics, because it uses only the road ground plane which is likely to be static. Localization methods that perform full 3D point-cloud matching often use the Normal Distributions Transform (NDT) [51] to perform matching between map and query scan point clouds [48, 52]. Alternatively, Wolcott *et al*. [53] proposed the use of multiple 3D LIDAR scanners to build multi-resolution Gaussian Mixture Maps (GMM), with localization performed using a lookup-table scheme.

The above solutions require expensive 3D LIDAR scanners not just for mapping, but also for the localization stage. There also exist much cheaper (although still expensive) 2D LIDAR sensors, which only operate over a single pitch angle. Localization methods using 2D LIDAR sensors [54, 55] still use an expensive 3D LIDAR for map construction, but then perform localization with one or more 2D sensors, typically using a particle filter and incorporating other sensor measurements (such as GPS and an Inertial Measurement Unit (IMU)).

While many of the above localization methods provide metric localization with up to centimeter-level accuracy, both 2D and 3D LIDAR sensors are still prohibitively expensive. They are also difficult to mount and calibrate on production vehicles. Also, in poor weather, reflections from rain or snow reduce the performance of LIDAR and can affect localization performance.

2.3 Localization with RADAR

While research based on RADAR sensors is less common than LIDAR, the lower cost and compact form factor of RADAR systems give them some advantages for automotive use. They are already used for parking assistance and vehicle following applications in production vehicles. They are also potentially much more reliable than LIDAR sensors, which depend on moving parts for operation. However, for

localization tasks, their low resolution is a difficult challenge to overcome. Because of the limited information in each RADAR scan, the removal of temporal objects is necessary to avoid false localizations. Dynamic objects such as moving cars are difficult to distinguish from stationary landmarks, complicating the localization process. Some more recent RADAR localization implementations include works by Lundgren *et al*. [56, 57], Rapp *et al*. [58], and Ward *et al*. [59]. Lundgren *et al*. [56] use off-the-shelf automotive RADARs (normally used for parking sensors) to augment inputs from other sensors for performing positioning within a map. The map consists of clustered RADAR measurements to create a RADAR landmark map (which is further optimized using the Expectation Maximization (EM) algorithm [60] to remove temporal objects in an extended version of the method [57]), and localization is performed with a particle filter. This method relies heavily on other sensors such as GPS and an IMU to achieve reliable localization. Rapp *et al*. [58] use dynamic occupancy grid maps which are constructed to enable the identification of RADAR measurements which are observing temporal objects, such as parked cars. However, while experiments using this technique gave results with a decimeter-level accuracy, it was only tested over a very small (48 m) trajectory in a confined parking-lot style space, so its performance in real-world traffic environments is unclear. The method by Ward *et al*. [59] replaces Monte-Carlo style methods with a Kalman estimator, which is used with the Iterative Closest Point (ICP) algorithm [61] to perform a topological positioning of a RADAR measurement relative to captures from a previous pass.

2.4 Localization with cameras

Visible light cameras are maybe the most appealing sensors for localization, perhaps because humans primarily use vision for navigation. Cameras are completely passive sensors and are available with very high resolution at relatively low expense. They are also easily fitted to production automobiles, and are already available on some models to support lane-keeping, pedestrian detection, and other intelligent vehicle

systems. There are a number of difficult challenges in performing localization using in-vehicle camera images. Lighting conditions drastically change the way a scene appears —for example, night-time localization may be difficult when using a database that was captured on a brightly lit day. Differences in weather can also cause similar lighting changes or even change to the scene structure itself —for example, when snow settles on the ground. While also a problem for other sensor types, occlusions from other vehicles and temporal objects are a serious issue. In this section, methods relating to general topological localization are presented in Section 2.4.1 followed by topometric localization methods in Section 2.4.2. Direct metric localization methods are presented in Section 2.4.3.

2.4.1 Topological localization

There are many implementations of topological methods using vision, which generally provide more scalable performance at the cost of lower precision when compared to topometric and direct metric localization. Topological localization methods typically compute whole-image discrepancies to determine the most likely position of a query image along a stream of database images. A whole-image discrepancy measure can be formulated by using color histograms [24], the Euclidean distance of dimension-reduced images [25], whole-image descriptors constructed in a similar way to SURF descriptors [17, 27], or a low bit-rate image sequence instead of single images [26]. Dynamic Programming (DP) [62] or Dynamic Time Warping (DTW) [63] can then be used for image matching [25, 26, 64, 65]. Techniques that rely on whole-image matching can be affected by situations where images captured at the same location may vary significantly, for example where dynamic traffic environments create occlusions. There are also methods that use invariant feature points for image matching without camera pose estimation, offering advantages in these situations —since usually visible features can still be matched even where a scene may be partially blocked, for example by a passing truck. Kyutoku *et al*. [66] uses feature points and monitors the epipole position of features matched between query and

database images. When two images are captured from similar locations, the epipole position of matched features moves towards the outside of the image, and this can be used as a similarity measure within a DTW framework. However, this system requires calculation of the essential matrix, similar to many topometric localization feature-based methods. Essential matrix calculation is a computationally intensive process, and requires many matched features which are spatially well-distributed for reliable estimation. It also relies on well-calibrated cameras.

A different method for performing topological localization is by using visual odometry [67] or lane markers [68] to determine an approximate vehicle trajectory or road shape, and then find the the segment of a road map which fits this trajectory [69]. Only a small database is required as no visual features are included, and feature matching only occurs between query images to perform odometry calculations. However, these methods clearly have limitations in accuracy, and also may fail on long straight sections of road where there is a lack of distinctive road shape.

2.4.2 Topometric localization

The simplest topometric localization systems use a topological whole-image matching methodology and localization information from the closest matched database frame [17, 27, 70]. To achieve practical metric localization accuracy, the database frames must be spatially close together, and a Bayes estimator is used to improve results.

Other topometric visual localization methods use invariant feature points [28, 71–75] which offer some robustness to illumination changes. As mentioned above, because an occlusion will rarely block all features from an image frame, they can also usually operate even under moderate occlusion. These methods usually match feature points between captured views and determine their 3D locations, creating a 3D feature-point map arranged into a topological factor graph [28, 29, 71, 72, 75, 76]. Metric localization is performed in the query stage by using the geometry constraints provided by query image features that are matched to the database features.

Local-feature-based methods that calculate 3D geometry suffer from several issues when applied to the vehicle ego-localization problem. The number of matched features necessary for exact camera pose estimation is large, with a recursive inlier selection process such as Random Sample Consensus (RANSAC) [77] required for estimation of image-to-image geometry. When too few consistent inlier feature matches are found, localization fails. Features must be well distributed over the captured scene, or else the geometry may be poorly conditioned, giving a false localization estimate. Sons *et al.* [75] use multiple cameras to help overcome this issue. The calculation of the camera pose is a computationally intensive process, and when the distance between the query and database images is small, short baseline degeneracies [78, 79] can occur. In addition, the number of database features required leads to very large databases which can be difficult to handle for real-world localization, where the database for an entire region or city may have to be stored or downloaded to the system. Pose estimation from extracted features also requires accurately calibrated cameras, and maintaining calibration for a camera in service on a consumer vehicle is a potentially difficult task.

2.4.3 Direct metric localization

A natural extension of image matching methods is to generate database images using a 3D prior map, allowing any potential view to be rendered within the map for direct metric localization. Aligning the database rendered views and the query images becomes an image registration problem, similar to the one encountered in medical imaging for the alignment of images from different modalities such Computed Tomography (CT) and MRI (Magnetc Resonance Imaging) scans [80, 81]. Medical image-registration techniques often make use of the maximization of mutual information between images of different photometric properties, in order to determine their geometrical correspondence. Even closer to the vehicle ego-localization problem is bronchoscope tracking by using image registration between real and virtual endoscope images [36, 82, 83].

For visual vehicle localization, continuous maps can be created using LIDAR scanners and calibrated cameras, giving textured photo-realistic maps [35]. LIDAR reflectance data can additionally be used to provide photometric information instead of a camera [33, 34]. Virtual renders of these maps allow localization to be performed by employing optimization of the camera pose with an objective function relative to the current query image. Textured maps can employ a simple per-pixel Sum of Squared Distance (SSD) objective function [84], but where a modality change occurs (i.e. with LIDAR reflectance maps), mutual information between the virtual rendered camera and real camera images is commonly used as an objective function [85]. Mutual information is a measure of entropy correlation between the images. Normalized Information Distance (NID) is a true metric version of mutual information which has also been applied in pose estimation from rendered map views [33–35].

Chapter 3

Topological Localization with Feature-Scale Tracklets

This thesis covers three main research topics within the theme of visual localization for vehicles using monocular vision. This chapter presents the first research topic on using feature-point scale for matching image frames between query and database sequences. A database sequence is constructed by capturing images and annotating them with capture position information. Localization of a query image frame is performed by determining which image frame from within the database sequence was captured from the closest location, through a similarity measure. This research uses image feature points, or more specifically, the comparison of the scale of feature points matched between images as a similarity measure between query and database image frames. Each query image frame is assigned a matched database frame, therefore performing topological localization. Evaluation of the proposed method was performed by comparing image matching results from the proposed method with manually matched images, and also by using images labeled with accurate localization information obtained by using a Mobile Mapping System (MMS).

This chapter begins with a background of the research in Section 3.1. An introduction to the novel contributions is presented in Section 3.2. The proposed methodology is described in Section 3.3, followed by an explanation of the experiments and

experimental results that were used to evaluate the method in Section 3.4. The results are further discussed in Section 3.5 before this chapter is summarized and concluded in Section 3.6.

3.1 Background

In Chapter 1, an introduction to vehicle localization was presented, proposing that solving the problem of ego-localization can be divided into three stages; topological localization, topometric localization, and direct metric localization. This chapter primarily addresses the problem of topological localization, where the location of the vehicle is selected from a discrete set of points that make up a topological map. The aim of the research presented in this chapter is to reliably and accurately perform image matching. Given a query image frame from an in-vehicle camera that requires localization, and a sequence of pre-captured database image frames, this method aims to determine the database image which was captured from the spatially closest position to the query image. Therefore, topological localization relative to the database (or map) is performed.

This method is based around the extraction and matching of scale-invariant features. There is an increasing number of proposed scale and illumination invariant feature detectors and descriptors, with potentially the two most frequently used methods being the original SIFT [86, 87] and its faster variant SURF [88]. The theory behind scale-invariant features, and why scale is suitable as a measure for comparing capture positions, is explained in Appendix A.

A feature detector determines the feature locations and scales, whereas a descriptor describes the feature region such that it can be matched to corresponding features in other images. The work presented here uses the SIFT feature detector. It was found that the feature scales extracted with SIFT keypoints were more consistent than those extracted by SURF. The reason behind this is explained in more detail in Appendix A. For feature description, SURF descriptors were chosen as they provided

better matching performance. However, in general, the proposed method is applicable to any feature extraction process that is performed at multiple scales for scale invariance, so it would work with many of the modern feature detection and description methods such as the Features from Accelerated Segment Test (FAST) [89], KAZE, and A-KAZE [90], and binary descriptors such as Binary Robust Independent Elementary Features (BRIEF) [91], Oriented FAST and Rotated BRIEF (ORB) [92], Binary Robust Invariant Scalable Keypoints (BRISK) [93], and Fast Retina Keypoints (FREAK) [94].

3.2 Contributed concepts

This section briefly summarizes the novel contributions of the research described in this chapter. The proposed localization method is based around two main components; the pre-matching of database features into feature-scale tracklets, and image matching of query image frames to database image frames using a novel per-feature image match voting technique.

3.2.1 Concept 1: Feature-scale tracklets

Image databases for visual localization typically store descriptor representations for each image [25, 27], and feature-based methods will contain a set of many feature descriptors per database image [64, 66]. Since feature-based localization methods perform feature matching between query and database images, a carefully constructed database of known inlier features is important for better matching performance. In the construction of the database, the vehicle is constantly moving forward. Therefore, the scale of features will increase as they are observed in consecutive database images. By making use of this and matching features between database frames, "feature-scale tracklets" can be created, where features matched between database frames are arranged into individual interconnected strings. This allows two things to be achieved in the database construction:

1. Pruning of feature matches where feature scale does not increase significantly with forward motion. Since the scale of matched features is used for localization of a query image, only selecting database features which exhibit a linear scale change with changing capture distance reduces the number of features which make an incorrect contribution to location prediction.

2. As each feature in the database is matched to corresponding features in adjacent frames, matching of query features to database features is performed just once instead of many times over several database images. The scales of the corresponding database features can also be quickly looked up, allowing a query feature's scale to be compared to a stream of matching database feature scales. In this thesis, the string of corresponding database features with scales is referred to as a "feature-scale tracklet".

The construction and use of feature-scale tracklets is the main contribution of this research, and it provides a novel use of the multi-scale extraction of invariant feature points. Feature-scale tracklets are described in more detail in Section 3.3.1.2.

3.2.2 Concept 2: Per-feature image match voting

Localization methods that use image similarity to predict location require the matching of a query image to a number of database image candidates, often using probabilistic methods [27, 70], or Dynamic Time Warping (DTW) [25, 65, 66]. To find the database image that matches the query image, typically a match cost is minimized and the query image is tested against many sequential database images. This is normally the most time-consuming part of the localization process, particularly for feature-based methods, as feature matching is relatively slow.

This research introduces a novel method for rapid convergence on the correct database match without the need to calculate the match cost for all local database images. The proposed method allows individual features to vote on the most likely database match, and convergence usually occurs within two or three image match tests. This

has the added advantage that an image match can be calculated with only a few feature matches. This is significant when occlusions or lighting changes reduce the number of feature matches available. The proposed image match selection process using this method is described in Section 3.3.2.1

3.3 Topological localization using feature-scale tracklets

In this research, visual localization depends on the information stored in a pre-captured visual database. The system comprises of two distinct stages: the database capture and construction, and the localization phase where an input sequence of image frames is used to determine the position of the vehicle relative to the sequence of database image frames. An overview of the system process can be seen in Figure 3.1. The process for the database construction phase is described below in Section 3.3.1. This is followed by an explanation of the localization phase in Section 3.3.2.

3.3.1 Database construction

In this research, the database is constructed by capturing a series of images at known locations. All vehicle routes to be localized must be previously traversed by the database capture vehicle —while this step appears prohibitive for large-scale localization, datasets such as Google Street View [9] have demonstrated that it is a possible task.

One consideration in the database capture process is the frame rate of image capture. The localization position of a query image comes directly from the closest database match, so a high frame rate is preferable to create a database with high spatial resolution. However, at high frame rates, image discrepancy between adjacent database images becomes very small and all image matching methods will struggle to distinguish between them. In addition, a small spacing between database frames leads

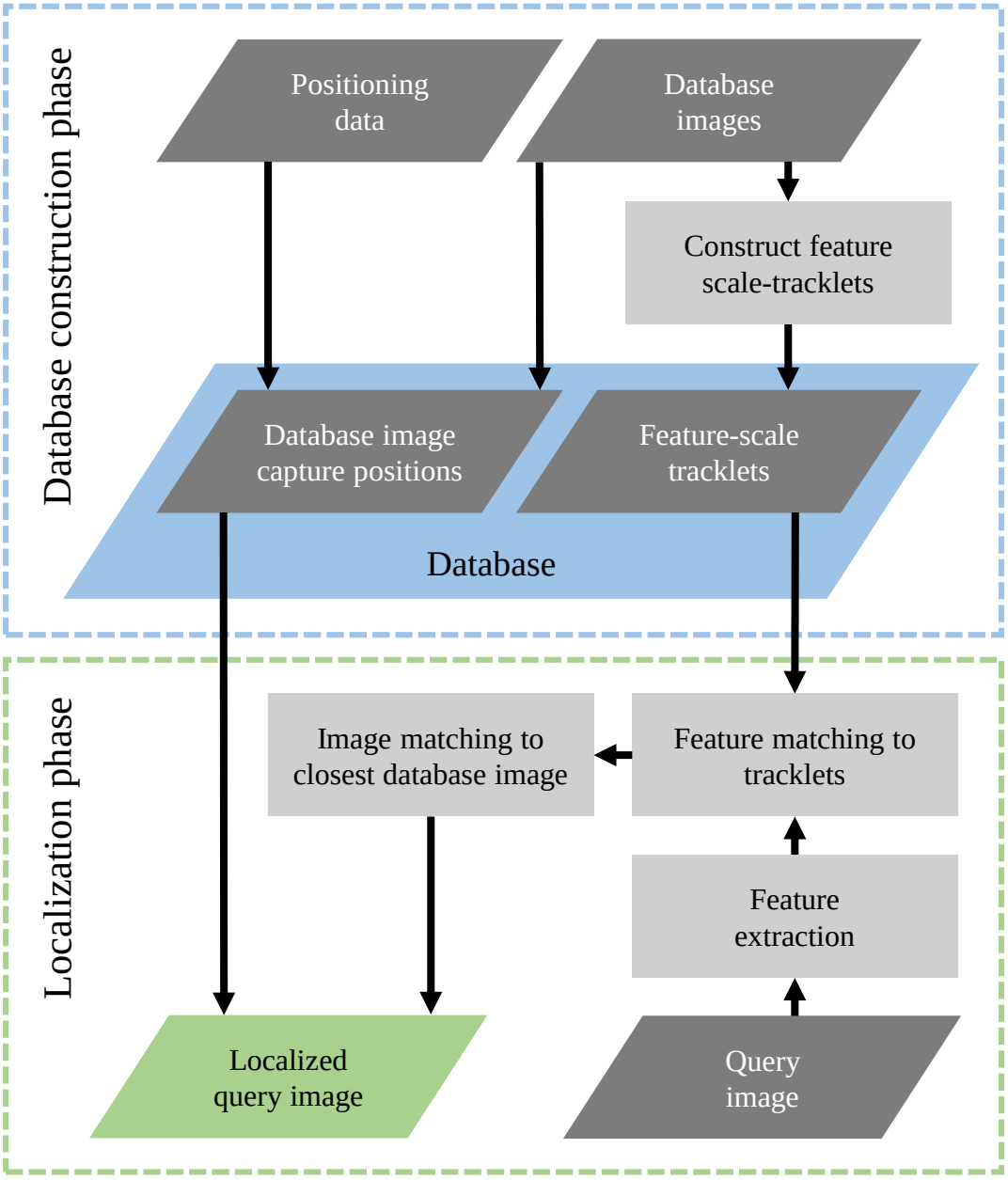


Figure 3.1: System flow diagram of the method described in this chapter.

to a very large database and slower localization. In this research, when localization was performed by applying the metric capture position data of the database images to matched query images, it was found that a 2 m spacing between database images resulted in approximately 1 m average localization error. The relatively wide image spacing also resulted in a lower database size. There is a trade-off between database

size and localization precision, with a relatively wide spacing between database image frames giving a smaller database but a larger average localization error. More information on experimental setup and results is presented in Section 3.4. Topometric localization at sub-database spacing resolution is the topic of the next chapter, Chapter 4 of this thesis.

To create the database, standard SIFT features are extracted. This method is relatively agnostic to the type of feature used, as long as they are extracted at multiple scales. Through experimentation, it was found that using moderate extraction parameters (approximately 400 features per image) gave enough matches with significant scale change.

One of the major issues with using features to calculate the relative pose between views is an ill-conditioned essential matrix. Using many features extracted from specific areas of an image, or many points that are related by a homography (i.e. on the same plane), becomes an issue when image geometry constraints are used to calculate the relative pose between views, as the pose becomes poorly conditioned. This can lead to inaccuracy or complete failure of the camera pose estimation. There are a number of methods that check features for such degenerate configurations [31, 32, 95]. However, with the method proposed in this chapter, the position of the features within the image is not important, and as long as they are not all distant, feature scale will change with distance regardless of where in the image they are extracted from, or whether they are related by a plane homography.

3.3.1.1 Feature-point matching between database images

The next step is to match extracted feature points between sequential database images. Since the proposed localization method does not calculate relative camera poses from image geometry, typical inlier selection schemes for feature matching using RANSAC [77] cannot be used. However, in road environments, there are certain constraints that can be applied to improve the rate of inlier matches. It is assumed

that the camera is forward facing, and that the primary camera motion is in the direction of the optical axis of the camera. Together with the assumption that camera height is constant, the search area for feature matches can be restricted. Within the limited search area, all candidate matches are checked for increasing scale. Any candidate match that shows a significant decrease in feature scale is discarded from the collection of potential matches. A weighted match cost is then applied. This employs scale and feature response weights to help identify the best potential matches. The best feature match for feature f is calculated by finding the feature g in the next database image containing the pre-pruned set of features which minimizes the following equation:

$$\gamma(f,g) = w_s|s(f) - s(g)| + w_r|r(f) - r(g)| + w_d(\mathrm{SSD}(f,g)), \tag{3.1}$$

where function $s(\bullet)$ returns the feature scale, function $r(\bullet)$ the feature response, and $\mathrm{SSD}(f,g)$ is the standard sum of squared differences between the feature descriptors. The weights w_s, w_d, w_r are adjusted to give a strong inlier set while maintaining a high number of matched features. Finally, the best matches are selected by dropping any match which has a descriptor distance of more than twice the minimum descriptor distance within the set of matches. All un-matched features are discarded from the database.

3.3.1.2 Feature-scale tracklet structure

The resulting database is a web of interconnected features, arranged into feature-scale tracklets, $T \in \mathcal{T}$ where $\mathcal{T}$ is the full set of database tracklets. A feature-scale tracklet T contains a list of M pre-matched database features from sequential frames,

$$\begin{aligned} T &= \{g_1, g_2, ..., g_m, ..., g_M\}, \\ g_m &= [\lambda_m, a_{\lambda_m}, \mathbf{u}_m, s_m, \mathbf{p}_m], \end{aligned} \tag{3.2}$$

with each $m = 1, 2, ..., M$ as the index along the tracklet. Here λ_m is the database image index which refers to the database image containing the feature g_m, and a_{λ_m} is the feature index within the corresponding λ_m. The descriptor of feature g_m is denoted as $\mathbf{u}_m$, s_m is the feature scale, and $\mathbf{p}_m$ is the vector containing the feature pixel positions.

3.3.2 Localization

This section describes how the proposed method achieves localization of input query images. The localization phase starts with feature extraction and matching to the features of the first candidate database image frame, and then comparison of feature scales within the database is used to converge on the closest database image match.

In the localization step, first a likely database image match is selected as a candidate based on the last match. In a real-world system, this could be initialized either with a normal consumer GPS or by using an even coarser topological visual search within a database to find a likely region to start in, such as sequence matching over dimension reduced images [26]. This would remove the dependence on GPS, but would potentially require a few minutes to capture a sequence suitable for general position recognition. The increase in database size would be small as only a few bits are required for sequence template matching.

SIFT features are extracted from the query image, and then matched to the recorded features of the candidate database frames. The same selective weighted feature matching given by Equation (3.1) is used, with the exception of the restriction on increasing feature scale, since it is unknown if the query image is before or beyond the first candidate database image. This results in each suitable query feature being matched to a feature-scale tracklet. The precise database frame match is then predicted using the proposed per-feature image match voting within the feature-scale tracklets, which is described in more detail below in Section 3.3.2.1. Once the database match g_m is selected, localization of the query image frame is complete.

If available, metric capture position information from the database frame can be applied directly to the query image frame. For the next query image, the next database image is used as the first candidate image.

This method has merit in that it does not require a motion model or velocity estimates to proceed with localization. While starting the process from the next database image assumes that the vehicle is moving forwards, it is not difficult to modify the method to accommodate backward motion. This research also does not consider localization at junctions. However, a discussion on possible implementation for backward motion and junctions is presented in Section 3.5.

3.3.2.1 Image match selection

In the per-feature, look-ahead image matching method used by this system, features extracted from a query image are matched to the features from a candidate database image λ. The feature matching process results in N matched query image features, with each feature f_i ($i = 1, 2, ..., N$) being mapped to a feature tracklet $T_t \in \mathcal{T}$ which contains the corresponding database feature g_m, as follows:

$$f_i \mapsto T_t \quad \text{where} \quad \min_t \gamma(f_i, g_m \in T_t). \tag{3.3}$$

Here the subscript t of T_t refers to the tracklet index within the database, and $\gamma(\bullet)$ is the feature matching function which finds the corresponding database feature by searching for the minimum descriptor distance, which was presented in more detail in Section 3.3.1.1.

The feature within the tracklet that is closest in scale to the current query image feature f_i can now be identified, and therefore the database image match can be predicted by the feature:

$$\tilde{\lambda}(f_i) = \lambda_{m'} \in g_{m'}, \tag{3.4}$$

$$m'(f_i) = \arg\min_m |s(f_i) - s(g_m \in T_t)|, \tag{3.5}$$

where the function $s(\bullet)$ returns the scale of a feature. All of the matched features can individually estimate and vote on which database image provides a match, with the most voted database image being selected as the image match as follows:

$$\lambda_{\text{match}} = \arg\max_{\lambda} \sum_{i=1}^{N} v(\lambda, \tilde{\lambda}(f_i)), \tag{3.6}$$

$$v(\lambda, \tilde{\lambda}(f_i)) = \begin{cases} 1 & \text{if} \quad \tilde{\lambda}(f_i) = \lambda \\ 0 & \text{otherwise} \end{cases}, \tag{3.7}$$

where λ_{match} is the database index of the image match.

An overview of the per-feature image match voting method is presented in Figure 3.2, and the scale comparison process for an individual feature is illustrated in Figure 3.3. By comparing query feature scale within the feature-scale tracklets, the many expensive feature descriptor matching steps can be replaced with a series of feature scale comparisons. Particularly if the candidate database image is actually quite far from the true match, there may be some variations in the voted match. In any case, the resulting database image is selected as the next candidate and the process is repeated. The localization method terminates when the current candidate database image results in a majority of votes for itself. When this happens, the database image is selected as a match, and the topological localization is complete.

3.4 Experiments

To evaluate the performance of the proposed method, two experiments were conducted, with two different datasets. These are summarized as follows:

1. **Traffic dataset experiment.** This experiment used images captured from a vehicle mounted camera which was driven on the same urban route twice, with one pass making up the database images and the next pass the query images. No localization information was available, so only the image matching

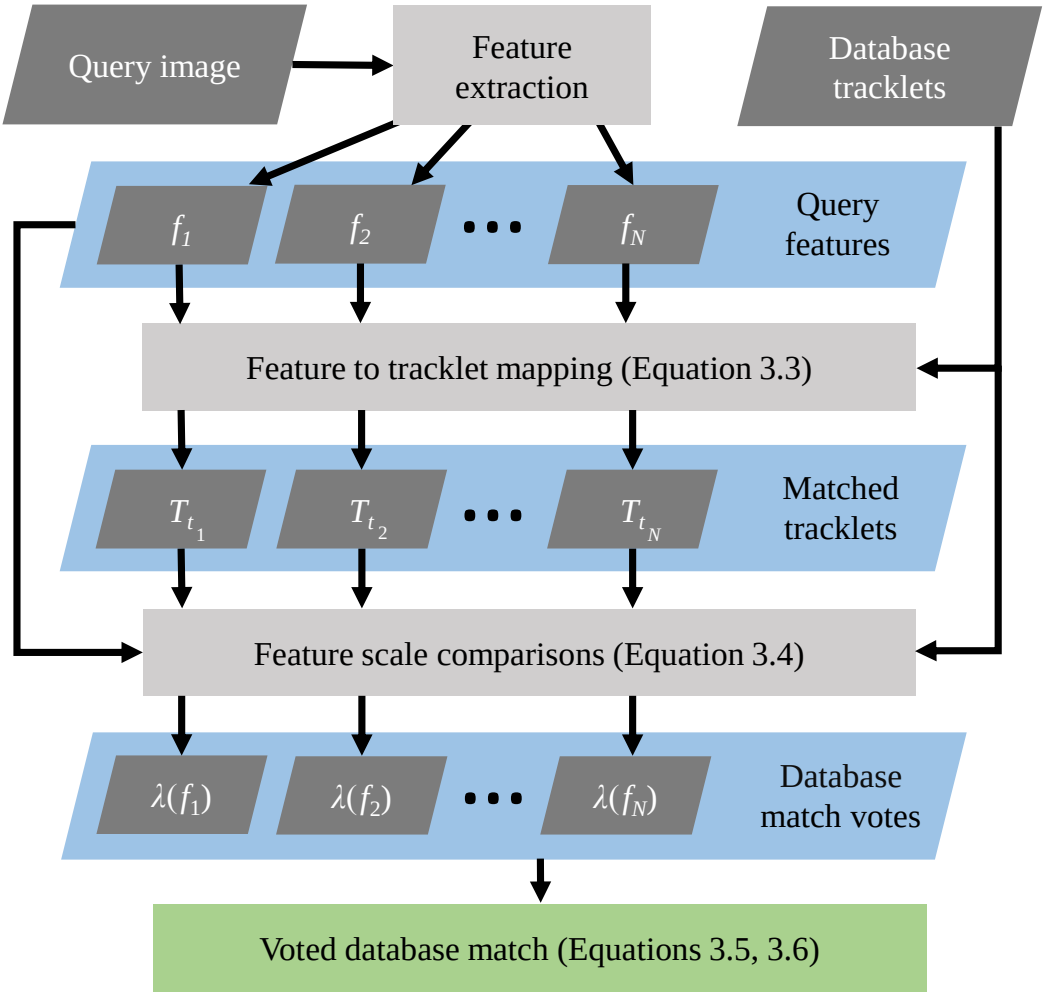


Figure 3.2: Overview of the per-feature voting process for selecting the closest database match. Each feature extracted from the query image is matched to a database tracklet feature, and supplies a vote for a candidate database image. The most voted database image is selected as a match.

performance of the system was evaluated by comparing results with a manually constructed ground-truth. This experiment allowed testing of the method with a relatively low-resolution camera in a real traffic environment, in the presence of other vehicles and so on.

2. **Driving-school dataset experiment.** This experiment made use of a sophisticated Mobile Mapping System (MMS) to provide accurate localization information for each frame capture using RTK-GPS, an IMU, and wheel odometry. This allowed database frames to be annotated with image capture locations in

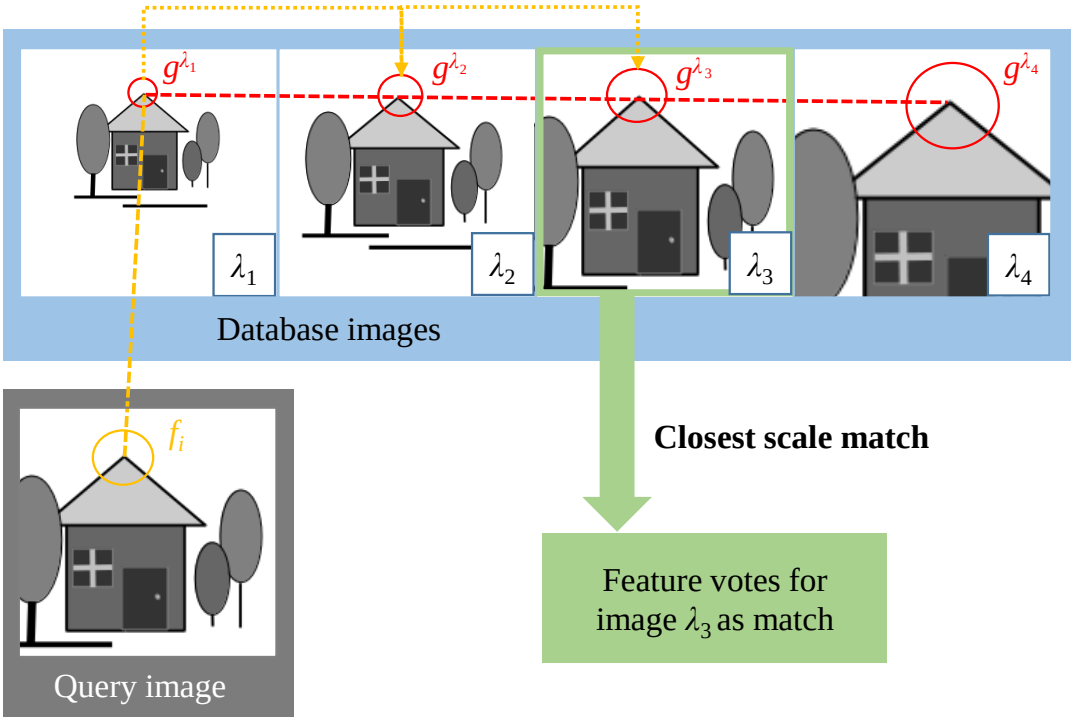


Figure 3.3: Simplified diagram of the per-feature image match voting. Only one feature is shown here for clarity. The scale of the feature in the query image is scanned through the corresponding database feature-scale tracklet, shown here in red. It can then vote on the predicted database image match, in this case image λ_3. The database image with the highest number of votes over all the features is then progressed to as the next candidate.

world coordinates, so query image frames matched to database frames using the proposed method could be generally positioned by applying the capture location of the matched database frame. This experiment was conducted at a driving school, where multiple passes and lane changes could be safely performed. This also allowed localization evaluation in lane change conditions.

This section starts with a summary of how the feature matching weights were determined in Section 3.4.1. The comparative and proposed methods used for evaluating the performance of the proposed method are explained in Section 3.4.2. The process and results of the topological localization experiments are presented in Section 3.4.3 and Section 3.4.4 respectively.

3.4.1 System parametrization

For feature matching in the database construction and localization processes, the weights w_s, w_d, w_r from Equation (3.1) were selected by visually checking feature correspondences, and choosing values that minimized incorrect matches. The scale difference of correct matches varies depending on feature size, so a relatively small w_s value of approximately one tenth of w_d and w_r (which were approximately equal) was found to be effective. This configuration prioritizes the SSD of feature descriptors for determining the best feature match. The weights were normalized to sum to one, resulting in a w_s value of 0.0476, with both w_d and w_r values set to 0.476.

3.4.2 Evaluated methods

To evaluate the performance of the proposed system, four localization methods were compared. The tested methods are summarized as follows:

1. **Proposed method: Feature-scale tracklet image matching.** This is the method described in this chapter, which uses the pre-matched feature-scale tracklets presented in Section 3.2.1 to match the query frames to their corresponding closest database frame, applying the per-feature image match voting presented in Section 3.3.2.1.

2. **Comparative method 1: Feature-scale DTW.** This method is based on previous work [65] and uses the average scale change between matched features of individual images as a cost measure. This method is implemented using DTW to compare image sequences. The image match cost can be considered as the averaged cost between all matched features, as in Equation (3.1) but using only the scale difference and setting w_d and w_r to zero. Like the proposed method, it uses matched feature-scale comparisons; however, it does not use the pre-matching and tracklet construction techniques presented in this chapter.

3. **Comparative method 2: Feature match cost DTW.** This method uses the same DTW framework as comparative method 1, but instead of using average scale change as a cost measure for image matching, it uses the feature match cost from Equation (3.1), averaged over all feature matches. It includes scale difference, response difference, and descriptor distance. The results for comparative method 2 as shown here are using the weights w_s, w_d, w_r as optimized for feature matching. Again, this method omits the proposed pre-matching and tracklet construction techniques.

4. **Comparative method 3: WISURF-DTW.** This is a modified implementation of WISURF [27] for image matching. The WISURF method creates a single SURF descriptor for each image, and chooses a database image match based on descriptor distance. Instead of using a Bayesian filter [27], here image matching is performed using the WISURF image descriptor distance and DTW to remove the dependence on motion estimation for localization. When the same lane was traversed in both database and query sequences, this method gave similar results to those presented in [70] in downtown areas —which is impressive, considering that only one camera was used in this testing.

Table 3.1 provides a comparison of the techniques used in the evaluated methods.

3.4.3 Traffic dataset experiment

This experiment was conducted to test the image matching performance of the proposed method in real traffic environments, with low-resolution images.

3.4.3.1 Image capture

A Point Grey Ladybug camera [96] was roof-mounted on a test vehicle, as shown in Figure 3.4, and used to capture both database and query image sequences in a city driving environment. The Ladybug camera is an omni-directional device made

Table 3.1: Summary of the techniques used in the evaluated methods.

Method	**Feature points**	**Feature-scale comparison**	**DTW**	**Tracklets**
Proposed: Feature-scale tracklets	✓	✓	—	✓
Comparative 1: Feature-scale DTW	✓	✓	✓	—
Comparative 2: Feature match cost	✓	—	✓	—
Comparative 3: WISURF-DTW	—	—	✓	—

Figure 3.4: Point Grey Ladybug camera [96] mounted on the test vehicle.

up of six cameras, covering a full 360° field of view. The images used to create the database and query image sequences came from the forward-facing camera only, with the additional cameras being used to assist in manual image matching for construction of the ground-truth, which is described below.

The camera capture rate was 15 frames per second (fps), and the captured images had their distortion removed prior to use using the proprietary Point Grey FlyCapture software [97]. The images were cropped to give a relatively small size of 406×300 pixels, which allowed for fast feature extraction and processing. The data capture route covered approximately 1 km of road, with the database and query sequences both being captured in the left-hand lane. The specifications of the data capture are summarized in Table 3.2, and the route traversed is shown in Figure 3.5.

To construct the ground-truth, the query images were manually matched to their corresponding database counterparts. The Ladybug camera captures six images from different directions simultaneously, and the FlyCapture software stitches the resulting images into a panorama at each capture point. The panorama images were used to find the closest corresponding capture locations from two image sequences, since the side views from the panorama allow easier comparison because of the limited lateral distance between the image sequences (which were captured in the same lane) [98]. This is illustrated in Figure 3.6, which shows how small angular differences between the correct query-to-database image matches make alignment using the forward-facing camera only difficult. By aligning the corresponding panoramas

Table 3.2: Capture specifications for the traffic dataset experiment.

Property	**Specification**
Camera model	Point Grey Ladybug 3
Imaging sensor	Color 2MP Sony ICX274 CCD, 1/8”
Lens field of view	Horizontal: 95°, vertical: 78°
Image resolution [pixels]	406×300 (calibrated, cropped)
Image capture rate [fps]	15
Capture route length [m]	~1,000

Figure 3.5: Traffic dataset capture location, showing the driving paths used. Satellite imagery: Google, ZENRIN.

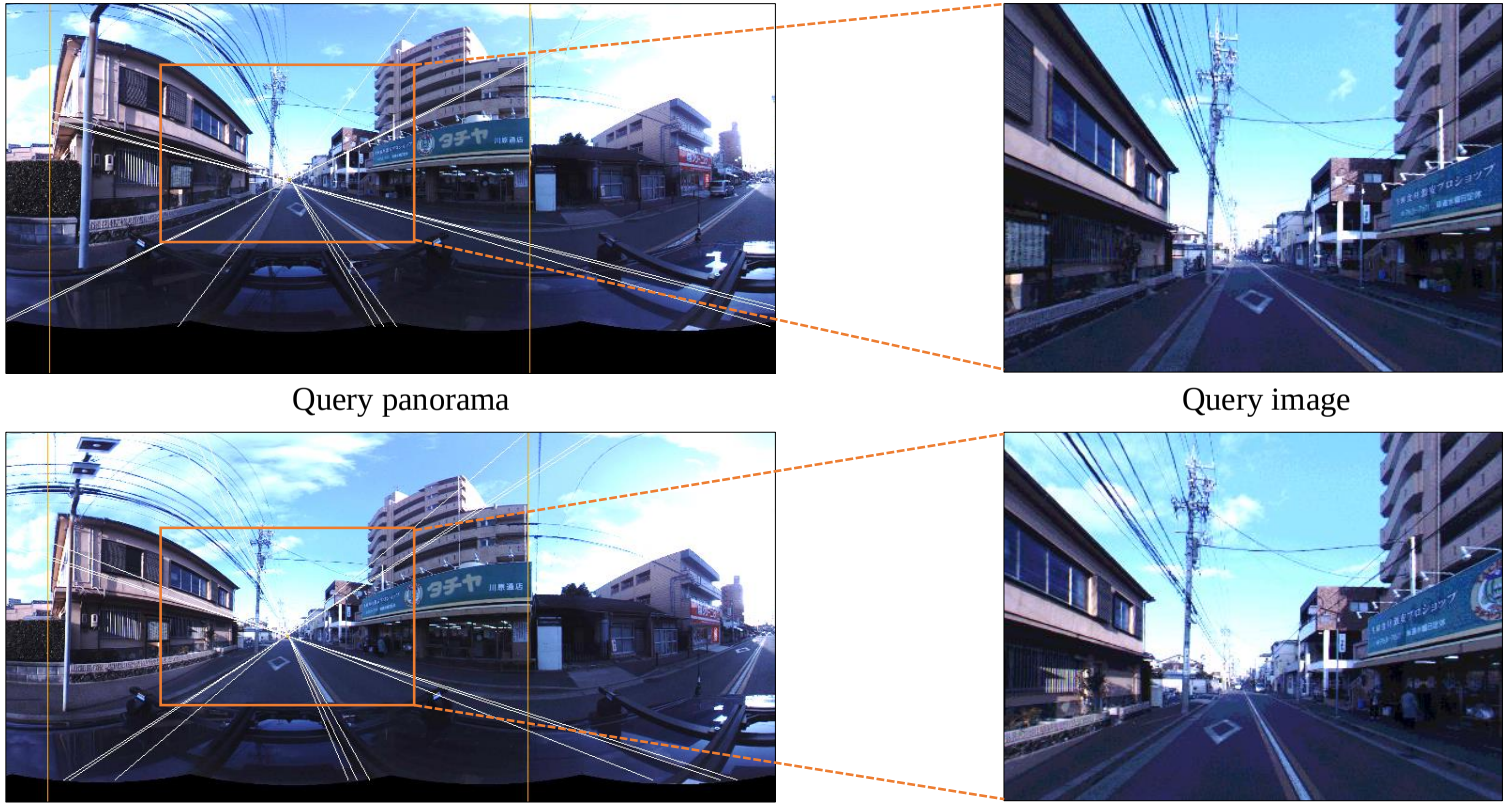


Figure 3.6: Ground-truth matches selected manually using Ladybug panoramas. The guidelines shown in the panorama images were drawn to aid in alignment. The corresponding forward-facing camera image of each panorama is shown on the right.

instead, the correct match can be found more easily. While still a time-consuming process, this greatly assisted in determining the correct ground-truth match as opposed to using the forward-facing camera images only.

3.4.3.2 Localization performance

Databases were constructed for the proposed and comparative methods, and localization was performed using them together with the query image frames. The experiments used a standard desktop computer with a 3.5 GHz Intel i7 processor. No GPU, multi-threading, nor optimization was used to increase performance, and the OpenCV C++ library [99, 100] was used to implement feature detectors and descriptor extractors. The proposed method was able to perform localization at the frame rate of the camera (15 fps).

Some examples of the resulting image matching are shown in Figure 3.7. As can be seen from the example images, lighting conditions are challenging for this dataset as there is a lot of contrast and shadow in the images.

By comparing to the image match results to the manually computed ground-truth, a frame match error was determined for each query image frame. The frame match error is defined as the discrepancy between the database image match result using the proposed method and the ground-truth image match, in terms of distance in number of frames. The results are presented in Figure 3.8.

3.4.4 Driving-school dataset experiment

This experiment provided a more thorough evaluation of the proposed method, with accurate capture position information available for both query and database images. This enabled more accurate determination of the ground-truth database image match for each image, and also evaluation in terms of metric localization error.

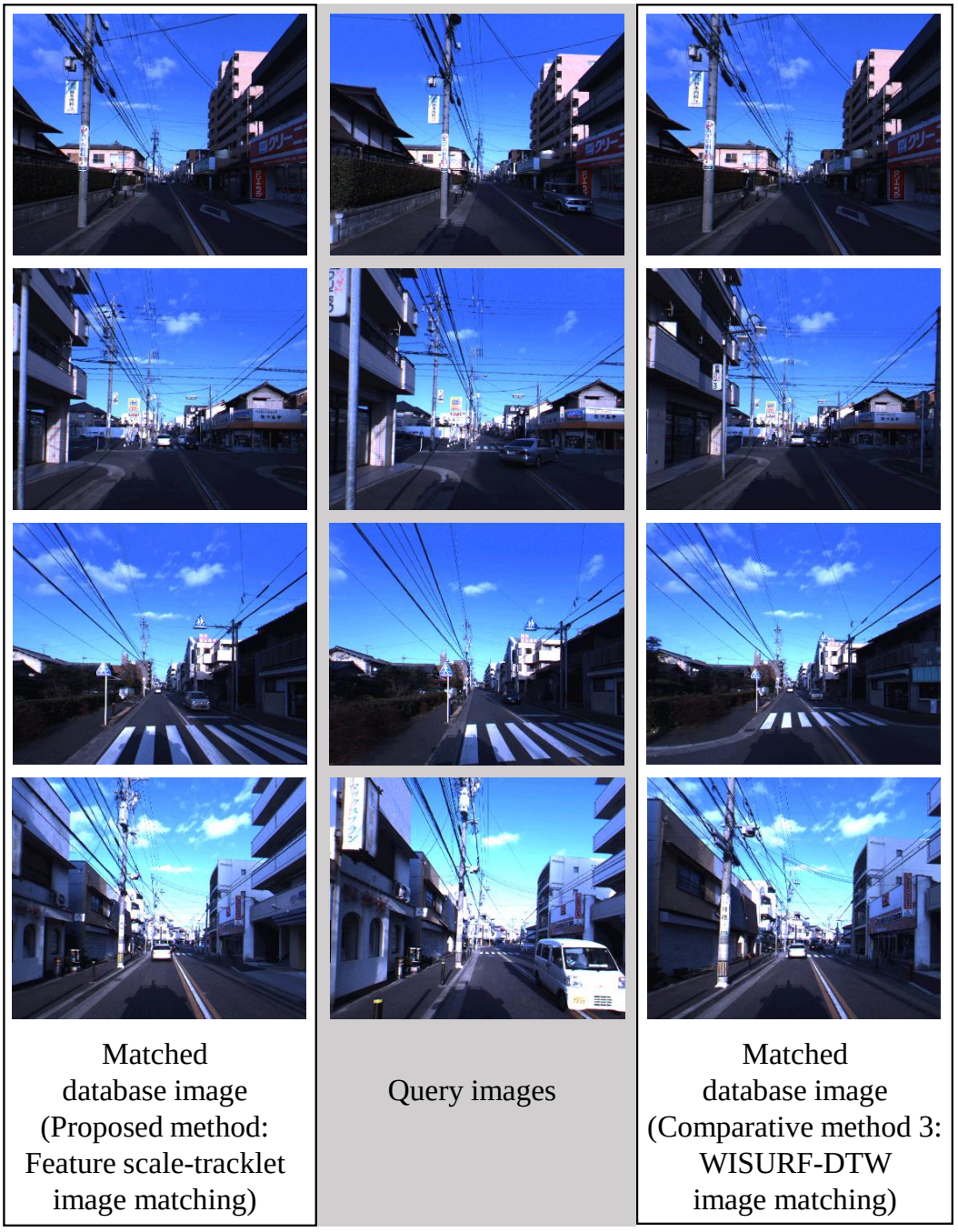


Figure 3.7: Example image matching results from the traffic dataset. The images include temporal features such as moving vehicles and harsh shadows.

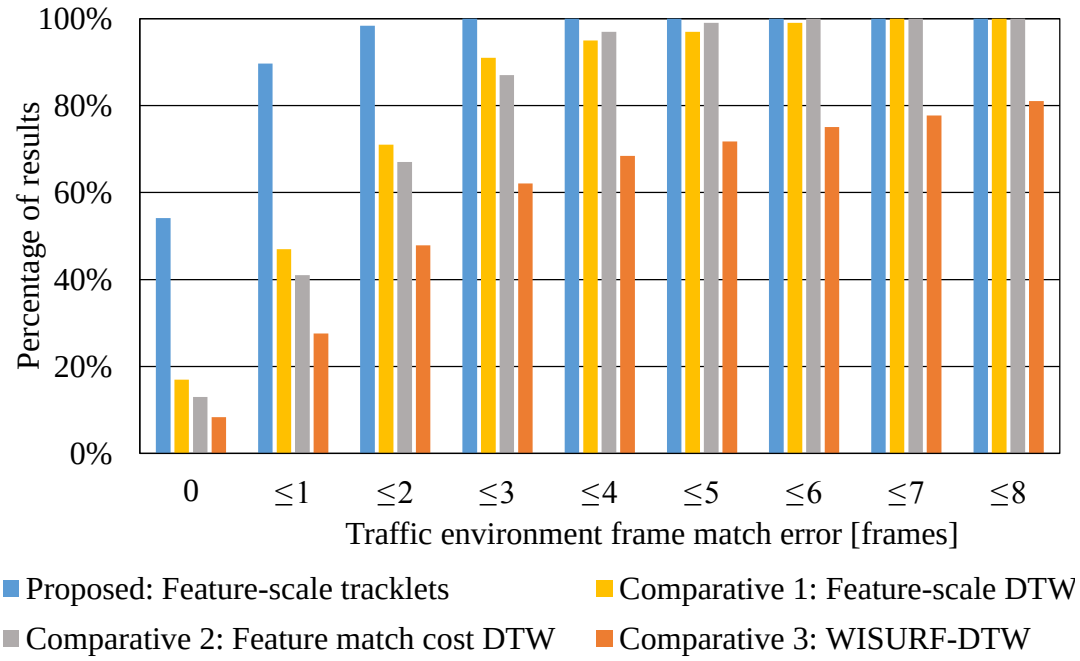


Figure 3.8: Image matching results from the traffic dataset. The *x*-axis shows the match error in terms of number of image frames between the selected database match and the actual closest database image. The *y*-axis shows the percentage of image match results that are within each error level.

3.4.4.1 Image capture

The dataset for this experiment was captured at a driving school. The location allowed various lane changes, traversal of intersections and maneuvers to be safely performed away from busy traffic. A sophisticated data capture system was employed to provide accurate ground-truth data in this dataset. Both database and query image frames were captured using a Mitsubishi Electric MMS-X320R MMS [101], shown in Figure 3.9. This system incorporates three roof-mounted 5 megapixel cameras, three laser scanners, GPS, and odometry hardware. Only one forward-facing camera was used in these experiments. None of the data collected from the laser scanners was used in this research. Capture position estimation for database construction and ground-truth is by a fusion of wheel encoder odometry, a high-accuracy GPS unit, and an IMU. This process is performed with proprietary tools and gives a claimed Root Mean Square (RMS) localization error of 6 cm. The system provided an estimated average error of below 1 cm in the experiments that were conducted,

Figure 3.9: The MMS used for data capture.

aided by good GPS satellite coverage. The MMS produces timestamped images with corresponding capture position coordinates, with an image capture rate of up to 10 fps. Because the system can accurately measure distance traveled, it can be set to capture images at set distance intervals, rather than a fixed frame rate. The actual image rate depends on synchronization with other hardware, but was at approximately 2 m intervals for most of this dataset. The camera uses a wide angle lens (horizontal: 80°, vertical: 64°), and although the native capture resolution of the MMS cameras are $2{,}400 \times 2{,}000$ pixels, it was found that localization performance was maintained after down-scaling to 600×500 pixels, which improved computation speed. The localization information supplied by the MMS for each image was used for tracklet construction in the database sequences, and for the query stream images, provided the ground-truth that was used in performance evaluation. Several database sequences were captured around the driving school, always using the left-hand lane. Query image sequences used a mixture of the left-hand lane and the right-hand lane where multiple lanes were available. Table 3.3 provides a summary of the capture hardware specifications and Figure 3.10 shows the vehicle paths.

3.4.4.2 Localization performance

Databases were constructed for the proposed and comparative methods, and localization was performed through sections of the dataset. The capture location of this

Table 3.3: Capture specifications for the driving-school dataset experiment.

Property	**Specification**
MMS model	Mitsubishi Electric MMS-X320R
Localization accuracy [m]	0.06 RMS
Imaging sensor	Color 5MP
Lens field of view	Horizontal: 80°, vertical: 64°
Image resolution [pixels]	600×500 (calibrated, scaled)
Image capture rate [fps]	10 (maximum)
Capture route length [m]	~1,000

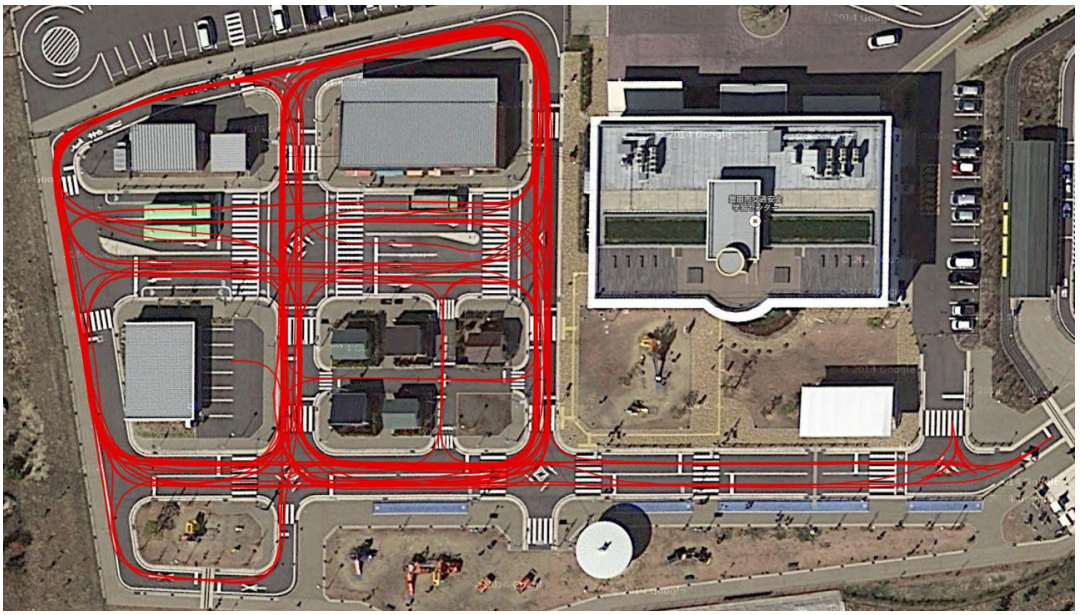


Figure 3.10: Driving-school dataset location, showing the driving paths used. Satellite imagery: Google, ZENRIN.

dataset allowed evaluation in two distinct situations: standard localization when the query images were captured in the same lane as the database images, and the lane change condition where the query images were captured in a different lane from the database images (the right-hand lane). The lane change situation often creates a substantial change in the appearance of the observed scene, which can be challenging for image matching methods. Some examples of the resulting image matching are shown in Figure 3.11, and Figure 3.12 shows the matched features of a sample set of tracklets spanning four database frames.

The same machine used for the previous experiment described in Section 3.4.3.2 was used to perform the localization process. Again, without any GPU implementation, multi-threading, nor optimization of any kind, localization was performed at approximately 15 Hz. At 100 km/h, a 14 fps image capture rate is required to achieve the 2 m image spacing used in these experiments. Therefore 15 Hz localization would be possible at up to 100 km/h without increasing the image spacing beyond 2 m. Processing time varies drastically with image resolution, so a balance between processing time and the number/quality of extracted features must be found.

Localization accuracy was evaluated using the MMS localization data. For an image matching ground-truth, image matches were determined by finding the spatially closest database frame capture position based on each query frame's capture location.

Topological localization using only image matching was performed and the results are shown in Figure 3.13. Here Figure 3.13(a) shows the image matching performance of the proposed method together with the comparative methods over the majority of the database where database and localization sequences were in the same (left-hand) lane. Figure 3.13(b) shows the image matching performance on a short sequence where the query images were captured in a different lane from the database images (the right-hand lane).

Absolute metric localization errors from topological localization were calculated using the distance between each query image and the chosen database image match.

Figure 3.11: Example image matching results from the driving-school dataset. The top two rows show image matching in the same lane as the database lane, whereas the lower two show matching where the query lane is different from the database lane.

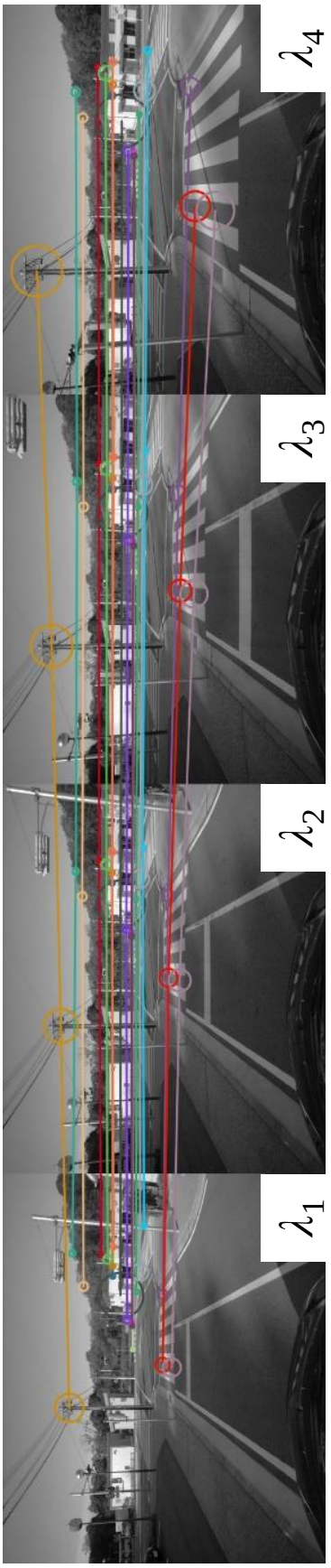
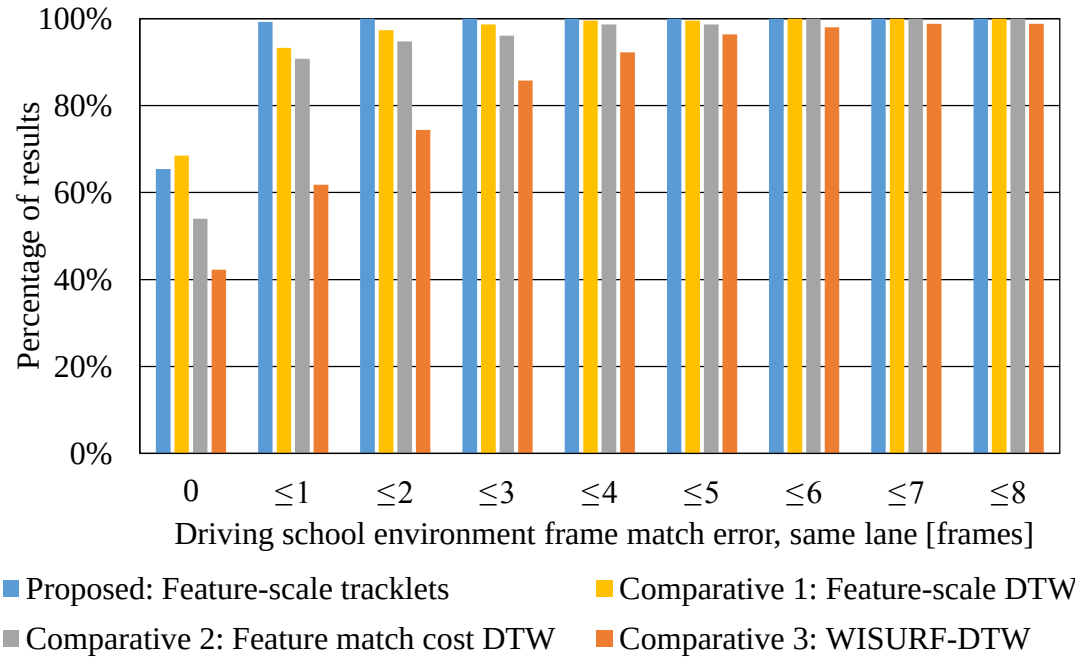
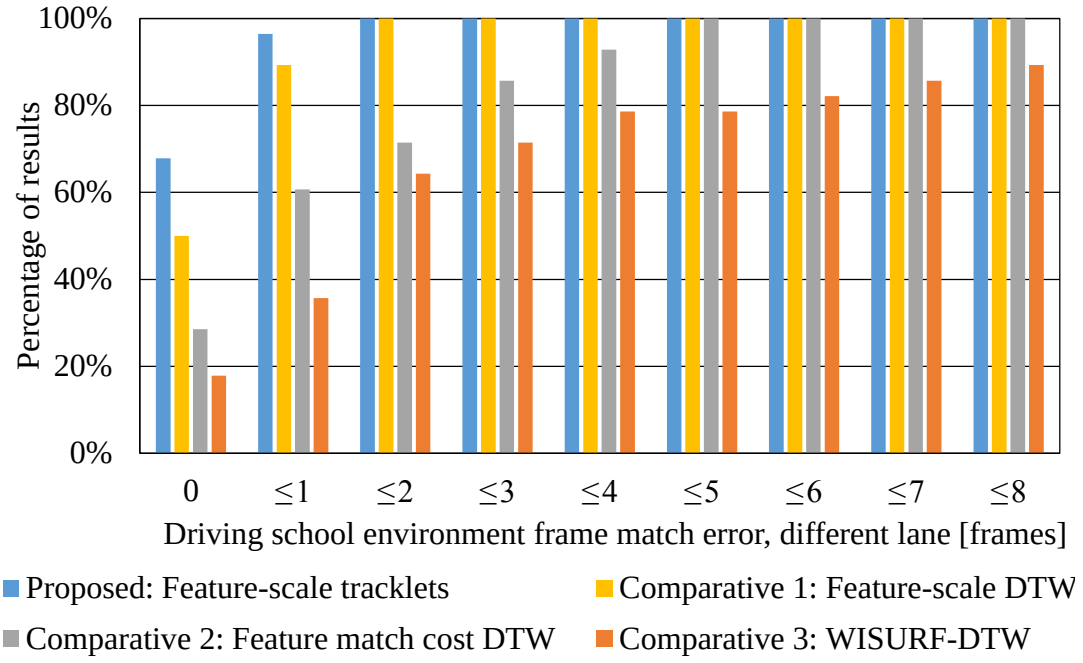


Figure 3.12: Sample feature-scale tracklets, with each tracklet shown in a different color. The circles represent the feature positions and their diameters show the feature scale, illustrating the scale increase along the tracklet. Only some of the tracklets are shown for clarity.



(a) Same lane.



(b) Different lane.

Figure 3.13: Image matching results from the driving-school dataset. These example sections are taken from areas where the query images were captured (a) in the same lane as the database images, and (b) in the lane to the right of the database image lane. The *x*-axis shows the match error in terms of number of image frames between the selected database match and the actual closest database image. The *y*-axis shows the percentage of image match results that are within each error level.

The results are presented in Figure 3.14 and Table 3.4. Again Figure 3.14(a) shows results from when the database and query frames are from the same lane, and Figure 3.14(b) shows results from the different-lane case.

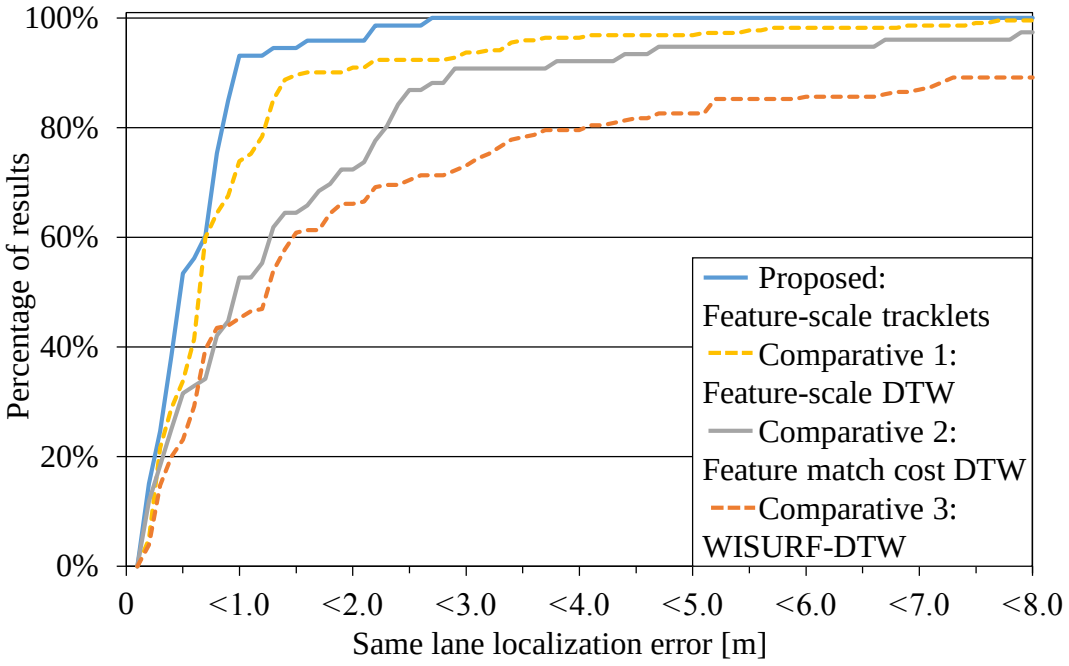
The proposed method achieved an average accuracy of 0.61 m in the same-lane case and 3.17 m in the different-lane case. It should be noted that when the query images were captured in the right-hand lane, even perfect image matching results would always result in at least a 2.0 m error. This corresponds to the approximate width of the lane, as both the proposed and comparative methods can only localize longitudinally in the direction of motion. This can be observed in Figure 3.14(b), where the different-lane curves start at an error level of approximately 2.0 m.

3.5 Discussion and analysis

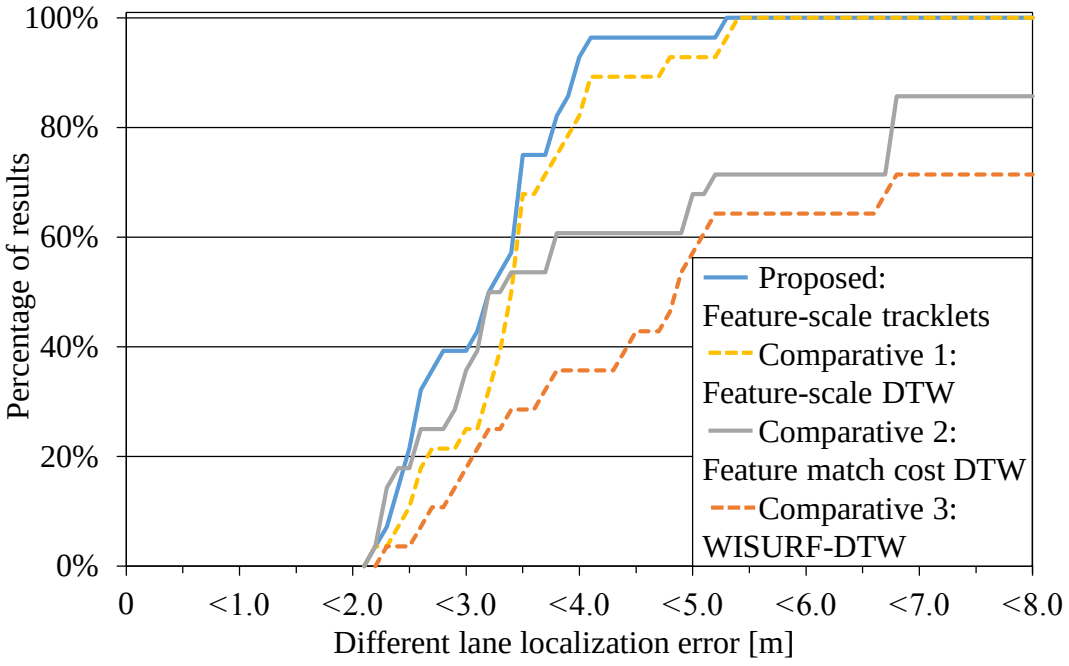
This section provides a discussion about the evaluated methods. The effectiveness of the proposed method is analyzed and compared to the comparative methods and the performance over the two tested datasets is discussed. Further exploration of actual tracklet statistics and the resulting effect on different road areas are presented, together with general considerations for practical real-world applications.

3.5.1 Evaluated methods

The approaches of feature-scale DTW (comparative method 1) and feature match cost DTW (comparative method 2) differ only in the weights used for the image match cost measure. While feature-scale DTW uses only the scale component of Equation (3.1) to determine an average image match cost over corresponding features, feature match cost DTW uses Equation (3.1) directly. Feature response and descriptor comparisons are included in the cost measure, with the same weights as used in feature matching. The superior image matching performance of feature-scale



(a) Same lane.



(b) Different lane.

Figure 3.14: Localization results from the driving-school dataset. These example sections are taken from areas where the query images were captured (a) in the same lane as the database images, and (b) in the lane to the right of the database image lane. The graph shows the percentage of localization results within each metric error level. The localization error starts at about 2.0 m for the different-lane results (graph (b)), which approximately corresponds to the width of the lane.

Table 3.4: Summary of the localization results from the driving-school dataset. Best results are shown in bold.

Method	Query lane	Average error [m]	Standard deviation [m]	Maximum error [m]
Proposed: Feature-scale tracklets	Same	**0.61**	**0.46**	**2.66**
	Different	**3.17**	**0.78**	4.56
Comparative 1: Feature-scale DTW	Same	1.00	1.32	11.04
	Different	3.43	**0.78**	5.25
Comparative 2: Feature match cost	Same	1.54	1.95	8.51
	Different	4.57	2.55	10.47
Comparative 3: WISURF-DTW	Same	3.71	1.68	14.33
	Different	7.14	4.17	24.10

DTW over feature match cost DTW illustrates the strength of using feature scale for image matching in this environment.

The proposed method employs two cost measures; one optimized for finding feature matches (Equation (3.1)), and one for image matching (Equation (3.4)). Comparative method 2 uses the feature match cost of Equation (3.1) directly as an image match cost measure. To compare the image match cost measure of the proposed method with comparative method 2, an experiment to optimize the weight values w_s, w_d, w_r of Equation (3.1) independently of the feature matching procedure was conducted. This allowed experimental determination of the best possible combination of all three weightings for an image match cost measure. It was found that the weight values that were optimal for individual feature matching did not provide optimal image matching performance when using Equation (3.1) averaged over all feature matches as an image match cost measure. The results showed that a scale difference weight w_s of 1.0 and 0.0 for w_d and w_r provided the best image matching. This is a system exactly equivalent to the image match cost within the feature-scale (only) DTW method of comparative method 1, and in principle the same image match cost used in the proposed method. The accuracy decrease when using descriptor distance is potentially because the descriptors of corresponding features from consecutive database frames are very similar. This results in ambiguity when choosing the image match based on descriptor distance, and therefore the occurrence of incorrect image matches. Feature-scale difference appears to be a more discriminative measure for resolving forward camera motion. Note that the feature response difference is very small for matched features and does not contribute significantly to image matching. It is, however, effective for filtering incorrect matches in the feature matching process.

The superior image matching performance observed when using an image match cost measure based on scale difference only, as in comparative method 1 (feature-scale (only) DTW) and the proposed method, confirms that feature scale-change within a tracklet (Equation (3.4)) is a good parameter for image matching. It also illustrates that comparing descriptors within the tracklet is unnecessary for the localization step.

This is where the efficiency gains of the proposed method become apparent, as descriptor comparisons are computationally expensive.

The proposed method also offers other efficiency advantages. Most query images are localized within three image matching steps, compared to at least eight when using DTW and image match cost comparisons. It is estimated that the image matching performance with feature-scale tracklets and the per-feature voting system is on average at least four times faster than the DTW-based feature-scale difference method (comparative method 1) and feature match cost method (comparative method 2) when using a similar number of features. While the image matching step is much faster when using the proposed tracklet system, the feature extraction process is the same for all feature-based methods and takes up a large proportion of the overall localization time. The scale change cost and feature match cost methods used for comparison ran at around 7 Hz compared to 15 Hz for the proposed method. Note that comparative method 1 and comparative method 2 run at close to the same speed, as the image cost measure calculation differs only by two floating point number value comparisons per feature. However, WISURF-DTW runs much faster than all of the feature-based methods and can run in real-time at all practical frame rates. As such, it makes a good basis for general localization in large datasets and could even be applied when a starting estimate is unavailable, for example as a replacement for initialization with GPS.

All three feature-based methods also showed significantly improved localization performance over WISURF-DTW, especially in the maximum observed error. While the use of the WISURF descriptor is a fast and efficient way to compare overall scene similarity, incorrect matches occur when consecutive database scenes either change very little in overall appearance, or when the query image scene is modified from the corresponding database scene —for example, in the case of a lane change or occlusion by another vehicle. Splitting each image into many descriptors can add robustness in these situations. Consecutive database frames have individual feature points which change in scale, and are repeatable even when a lane change takes place

or an occlusion obstructs some of the scene. The advantage of using many feature-point descriptors instead of an overall image descriptor is further discussed in the following sections.

3.5.2 Topological localization in the traffic environment dataset

The image matching performance in the real-world traffic environment of the first experiment (Section 3.4.3) is substantially lower than that in the experiment using the driving-school dataset (Section 3.4.4). There are several reasons for this. Firstly, the camera captured images at a rate of 15 fps, whereas in the driving-school dataset experiment the database image spacing was approximately 2 m. The fast capture rate in the traffic environment dataset resulted in much closer image separation and therefore smaller differentiation between frames. Secondly, occlusions and dynamic traffic scenes causing differences between the database and query images pose a challenge for image matching. Looking at individual matching failures, the comparative WISURF-DTW method was much more affected by both of these issues than the proposed method, which is evident in the image matching results (Figure 3.8). However, what is harder to see, is that partially occluded scenes such as the query image in the last row of Figure 3.7 sometimes caused complete matching failure when using WISURF, whereas the proposed method's matching performance was mostly unaffected by such situations. Finally, the lower resolution of the Ladybug images and the poorer lighting conditions play a large part in extracting quality features for matching.

Database images captured from busy traffic environments will sometimes include tracklets with features extracted from temporal objects such as other vehicles and pedestrians. These tracklets do not have much effect on the localization process, as even if they are incorrectly matched to query features, their database index votes are outweighed by those of static features. In future work, vehicle and pedestrian detection systems could be used to remove the features from temporal objects. This would

slightly reduce the database size and matching computation time. Another potential method to remove temporal object features would be to make multiple database capture passes and create tracklets from only consistent features over different capture sets. Information about matched features in the localization process could also be used to improve the database set of tracklets, which would additionally allow database evolution over time.

3.5.3 Topological localization in the driving-school dataset

The average localization error when database and query images came from the same lane was lower than in previous work [65], even with the use of only one camera, showing that the per-feature database match voting system provides a more stable image matching platform. In the driving-school dataset, the proposed feature-scale tracklet method reduced the average same-lane localization error by 32% and 44% over the feature-scale DTW and feature match cost DTW methods (comparative method 1 and comparative method 2), respectively. The average localization error was reduced by 82% when compared to the comparative WISURF-DTW method.

Image matching is more challenging in the different-lane case because the scene viewed by the camera changes significantly between lanes. This is illustrated by the comparative WISURF-DTW method results shown in Figure 3.13(b) and Figure 3.14(b). The WISURF-DTW method showed much higher localization error in the different-lane case when compared to localizing in the same lane as the database, even when considering the 2 m lane offset. The proposed method, after taking into consideration the lane offset, actually achieved similar image matching and localization performance to the same-lane case. This illustrates one strength of using features for localization, as opposed to whole image similarity.

The proposed database construction method makes use of the forward motion of the vehicle for filtering strong features for localization. However, as can be seen in Figure 3.10, the database includes many sharp corners. At these locations, the change in vehicle direction of motion causes some of the tracked features to disappear from

Table 3.5: Tracklet statistics for the driving-school dataset.

	Maximum	**Minimum**	**Average**
Tracklet length [frames]	17	2	3.3
Number of tracklets per frame	789	78	255.4

the field of view. This results in the end of the tracklets and the creation of new ones as the view changes. To see how many frames the tracklets typically cover, the average number of tracklets passing through each frame and average length of each tracklet was calculated. The tracklet length is defined as the number of consecutive database images in which it appears, and the number of tracklets per frame is the number of tracklets that pass through a database image. A summary of the statistics of the tracklets constructed within the complete driving-school dataset is presented in Table 3.5. The tracklet statistics show that while long tracklets that contain features in many consecutive frames do exist (and are probably important for the method to work effectively), there are also many short tracklets which bring the average tracklet length down to not much more than the minimum length of two frames. Further testing is required to see if there is the possibility to determine the likelihood that a tracklet will make a strong contribution to localization based on the number of frames it runs through. Additionally, these short tracklets may provide reliability where sharp corners exist.

To determine the affect of corners on localization accuracy, one small section of the driving-school database including two straight sections and two 90° corners was isolated, and the average overall localization error from this area was compared to the average localization error of both corner areas and straight areas independently. The results showed an average localization error of 0.73 m in corner areas, and 0.68 m in straight sections with an overall error of 0.69 m for the complete tested area.

3.5.4 Database size

One consideration is the database size. The proposed method creates a database of up to about 100 times the size of the WISURF-DTW comparative method, depending on the number of features extracted per image. The descriptors average about 120 KB per meter. The localization accuracy would definitely be improved by creating a database with 1 m image spacing instead of the 2 m used in these experiments; however, this would also result in a database of twice the size. There are potentially many ways to reduce the database size, including using simpler descriptors, or monitoring and pruning descriptors which never get matched to query image features over time. Within the feature-scale tracklets, maintaining just one descriptor per tracklet rather than the full set of feature descriptors over several database images is also a possibility. This is investigated further in the next chapter, Chapter 4. While the database is relatively compact when compared to storing dense feature point maps or complete visual information, it would still require careful compression to be used in a real-world localization system. Modern in-car navigation systems store the entire map for a whole region or country. With the database of the method proposed in this paper, this may be difficult to realize in practice due to its larger size. However, it is compact enough to be streamed over modern mobile data networks, and as solid-state storage technology becomes more affordable, larger databases may be able to be stored for an entire region or country within the vehicle's navigation system.

3.5.5 Reversing and junctions

In this chapter, backward motion and junctions (where the vehicle may move down one of several potential roads at an intersection) are not considered. While forward motion is assumed, backward motion would usually be detected and correctly handled as long as the matched tracklets sufficiently cover database frames behind the current camera position. However, also testing the previous database image as a match candidate would potentially improve performance where backward motion is likely. Places where the road forks would also not be difficult to include. At such

locations, several candidate images (one for each possible traversal route) would be selected as opposed to a single one. As localization progresses along each potential traversal route, the route with the lowest match cost would be selected as the correct one. Topological systems inherently support this methodology, and a similar process has been used successfully with the WISURF method [70].

3.5.6 Camera calibration

In both the datasets that were used to evaluate the method proposed in this chapter, calibrated camera images were used. Most feature-based visual localization methods will strictly require calibrated images, as the exact pixel location of each matched feature contributes to the calculation of the relative pose between views. However, the proposed method does not require the exact pixel location of feature points to perform localization, only the scale of the features. Therefore, in theory, camera images could be used without calibration. Camera lenses that have a lot of distortion would potentially still cause issues with feature description and matching, and probably even alter the scale of feature points; but if the lens is relatively distortion-free, it is unlikely that localization performance would be greatly affected. This is a large advantage when considering real-world localization systems, where a method that is robust to small changes in calibration is desirable. Changes in temperature, external impacts, and so on can cause small changes in intrinsic and extrinsic camera calibration, and re-calibrating cameras that are in service is potentially troublesome. In future experiments, thorough testing of the method with both calibrated and uncalibrated cameras would be an interesting extension to this research.

3.6 Summary

In this thesis, three vehicle localization types using monocular cameras are explored. As introduced in Chapter 1, these are topological localization, topometric localization, and direct metric localization. These localization types provide progressively

refined ego-localization, with topological localization giving the current general vehicle position relative to a map, topometric localization positioning the vehicle metrically in a coordinate frame, and direct metric localization performing a full 6DOF position and pose estimate.

In this chapter, a topological methodology for positioning the current vehicle position in a pre-constructed database based on visual similarity was presented. This solves the problem of general ego-localization relative to a map.

The methodology, experiments, and analysis provided in this chapter presented an effective system to perform topological localization of a vehicle using a single camera and a pre-constructed database. The proposed database construction method makes use of the typically linear and forward motion of a vehicle to perform feature matching and feature-scale tracklet construction, providing a framework for image matching of a query image. Query image localization uses matching of scale-invariant feature points to the database tracklets to perform a per-feature database image match voting platform for determining the spatially closest database frame. This method does not calculate epipolar geometry constraints between features so does not require a large number of adequately spaced feature matches to operate.

The experimental results showed that robust image matching can be achieved using this method. Strong image matching performance was observed in the case of a traffic environment dataset with low-quality images of dynamic road scenes. In the case of a driving-school dataset, accurate localization ground-truth was available for evaluation. This allowed the effectiveness of the localization process to be tested, by applying capture positions from the matched database image frames to their corresponding query image frames. Calculating the distances between the query image localization results and the ground-truth capture positions, average localization errors of 0.61 m when database and query image sequences were in the same lane and 3.17 m when they were in different lanes were observed. With the spacing between database images being about 2 m, this demonstrates effective image matching for topological localization. The different-lane results include a lane offset

error (approximately 2 m), so even under significant visual changes that reduce the performance of whole image similarity-based methods, image matching results were maintained.

Chapter 4

Topometric Localization with Feature-Scale Regression

This thesis covers three main research topics within the theme of visual localization for vehicles using monocular vision. Chapter 3 introduced the construction of feature-scale tracklets to perform per-feature image-match voting, with the aim of matching a query image frame to a database image frame for topological localization. This chapter presents the second research topic on performing topometric localization by expanding on the feature-scale tracklet concept. The capture position of database frames is used together with scale information within each tracklet to determine regression coefficients that relate the scale of each feature to metric capture position. This allows topometric localization of query frames to sub-database frame spacing precision. Like the topological localization method presented in Chapter 3, the proposed method allows each individual matched feature to contribute to the visual location prediction. However, rather than each matched feature voting on the closest database match, in this research, they instead produce individual metric estimates of the current location. These are combined to produce a normally distributed visual location measurement, that provides a probability density function for the measurement update step of a Bayes estimator.

This chapter begins with a background of the research in Section 4.1. An introduction to the novel contributions is presented in Section 4.2. The proposed methodology is described in Section 4.3, followed by an explanation of the experiments and experimental results that were used to evaluate the method in Section 4.4. The results are further discussed in Section 4.5 before this chapter is summarized and concluded in Section 4.6.

4.1 Background

In Chapter 1, an introduction to vehicle localization was presented, proposing that solving the problem of ego-localization can be divided into three stages; topological localization, topometric localization, and direct metric localization. This chapter addresses the problem of topometric localization. The aim of the research presented in this chapter is to build on the topological localization presented in Chapter 3 and increase the functionality of feature-scale tracklets to enable a metric position estimate for a query image. This means that the localization can be performed with a metric error that is not as heavily dependent on the database image frame spacing.

Like the topological localization method presented in Chapter 3, the research contained in this chapter is based on scale-invariant features. Again, this method is agnostic to extraction and description types, as long as the features are scale invariant. The explanation of image features and scale invariance is referred to Section 3.1 and Appendix A.

Topometric localization estimates a position in space, which is a continuous quantity (unlike one-to-one image matching, which is a discrete problem). The step from topological localization to topometric localization requires some form of interpolation between database capture locations. In this research, interpolation is performed with regression of feature scale and capture locations. Linear least-squares regression is used to determine the best linear fit between capture location and feature

scale within each feature-scale tracklet. In this case, feature scale is the explanatory variable with capture location along a road segment being the dependent variable.

The maintenance of a current position estimate is aided by a measure of the degree of confidence in the current measurement, so that this can be combined with a prediction based on previous motion to provide more reliable and smooth localization. In this research, this is performed using a Bayes estimator. If a variety of information is available about the system state, a Bayes estimator can combine this information, together with the confidence of each state estimate, to provide the most probable current state. For a moving vehicle, the current state (in this case, location) can be predicted from previous states using a motion model, and then updated using sensor measurements. The per-tracklet position estimate provides a variance estimate and has a Gaussian distribution (see Section 4.5) which makes a Bayes estimator a good choice for including other measurements, control inputs, or a motion model, in order to improve the overall localization performance. In this application, state models are assumed to be approximately linear, which allows the use of a particular kind of Bayesian filter, the Kalman filter. Bayesian estimation and different estimators, and the notation used for them in this thesis, are described in Appendix B.

4.2 Contributed concepts

This section briefly summarizes the novel contributions of the research described in this chapter. The research presented here contains two main contributions; the structuring of feature-scale tracklets to include regression coefficients to interpolate position with feature-scale change, and a Bayes estimator for combining per-tracklet position estimates and providing a final topometric localization result.

4.2.1 Concept 1: Feature-scale tracklet regression

In Chapter 3 the feature-scale tracklet concept was introduced. In this chapter, a method for using scale and capture position regression within tracklets is proposed. The key concept of feature-scale tracklets is that corresponding features between images have a continuous scale, and in the case of a vehicle moving linearly along a road, the feature scales increase with each consecutive image. The assumptions behind feature-scale tracklet regression is that feature scale within a tracklet increases linearly with capture location along the tracklet. This in turn leads to the assumption that the majority of the motion contained within a tracklet is linear. If these two assumptions are true, then because the feature-scale property is continuous, the relationship between scale and capture position can be modeled with linear least squares. The resulting linear regression line can be used to interpolate the capture position of a query feature even between database capture locations. The calculation of regression coefficients for feature-scale tracklets is described in Section 4.3.1.2.

4.2.2 Concept 2: Bayes estimator for localization with feature-scale tracklets

With the feature-scale tracklet regression coefficients, each matched query frame feature provides a current position estimate. This is in contrast to a per-tracklet database image match vote in the topological localization method of Chapter 3. With many matched features, the distribution of position estimates for all the feature points in the query frame forms a normally distributed Probability Density Function (PDF), which is ideal for use in the measurement update of a Bayes estimator, the theory of which is described in more detail in Appendix B. In this research, the measurement and state transition models are assumed linear, allowing the use of a Kalman filter for state estimation. The implementation is presented in Section 4.3.2.1.

4.3 Topometric localization using feature-scale tracklet regression

As with Chapter 3, this research has a database construction stage and a localization stage. The pre-captured visual database contains feature-scale tracklets, but the contents are modified to include regression coefficients that allow topometric localization. The estimation of the continuous position property is required, so the system is designed around a Bayes estimator. An overview of the system flow can be seen in Figure 4.1. The visual localization estimate process is illustrated with example features in Figure 4.2, which shows how the scale of the query features are used to interpolate capture position along the road segment using the database regression coefficients of matched database tracklets. The process for the database construction phase is described below in Section 4.3.1. This is followed by a more detailed explanation of the localization phase in Section 4.3.2.

4.3.1 Database construction

In this research, as with Chapter 3, the database is constructed by capturing a series of images at known locations. The general database construction commences in the same fashion as in Chapter 3; as described in Section 3.3.1 up until and including Section 3.3.1.1. However, the formulation of the feature-scale tracklets themselves is different. The relationship between the continuous scale property and the continuous image capture position is calculated using a linear least-squares regression for each tracklet, and the resulting coefficients are added to the tracklet for use in the localization phase.

4.3.1.1 Feature-scale tracklet structure

SIFT feature points and SURF descriptors are extracted from each database frame and are matched in consecutive frames, then arranged into feature-scale tracklets.

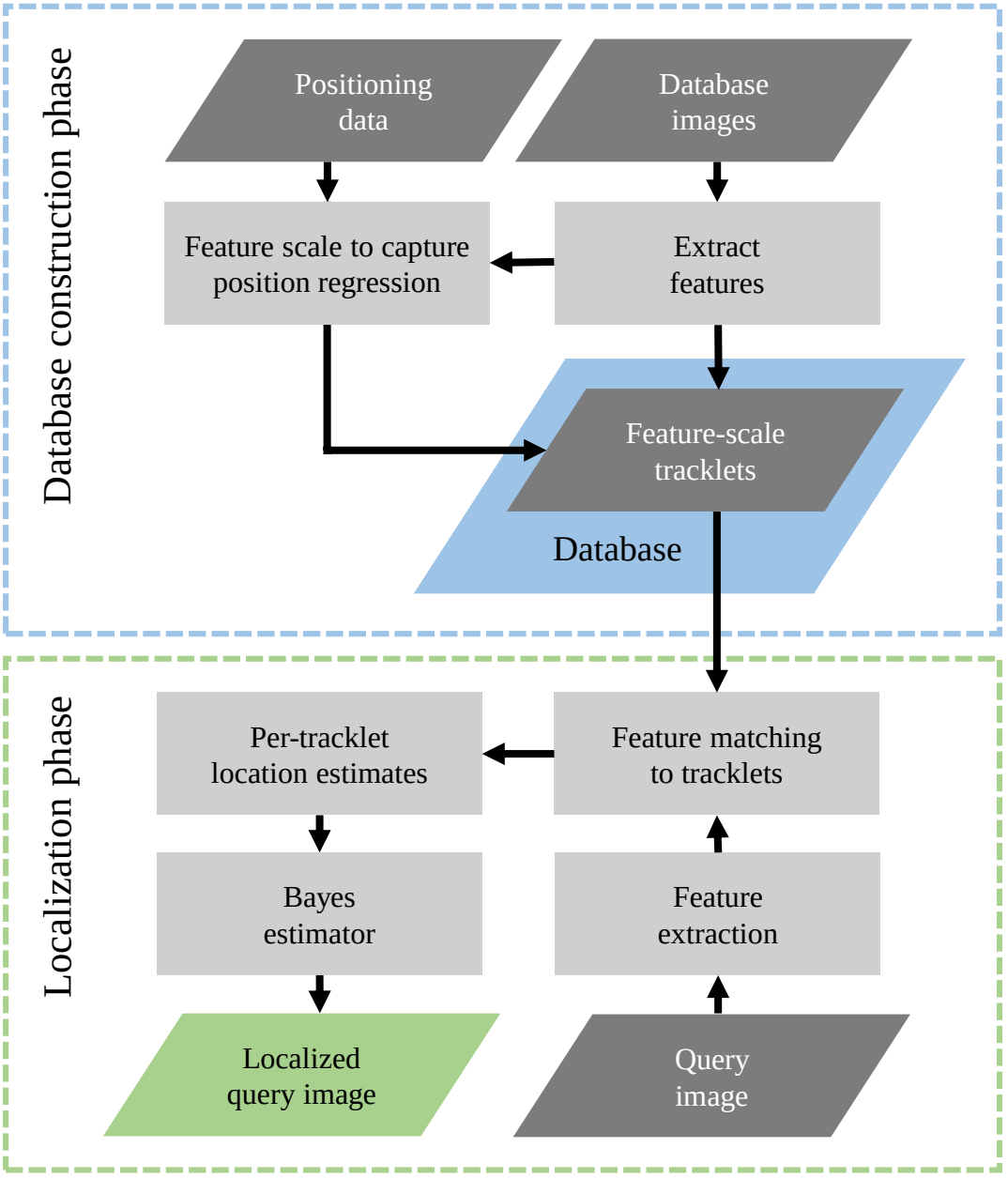


Figure 4.1: System flow diagram of the method described in this chapter.

The notation for describing feature-scale tracklets is adapted from Chapter 3. In Equation (3.2), the nomenclature for the contents of each tracklet was described. With the addition of scale regression, the new feature-scale tracklet structure needs to also include the capture location of each feature, and the calculated regression

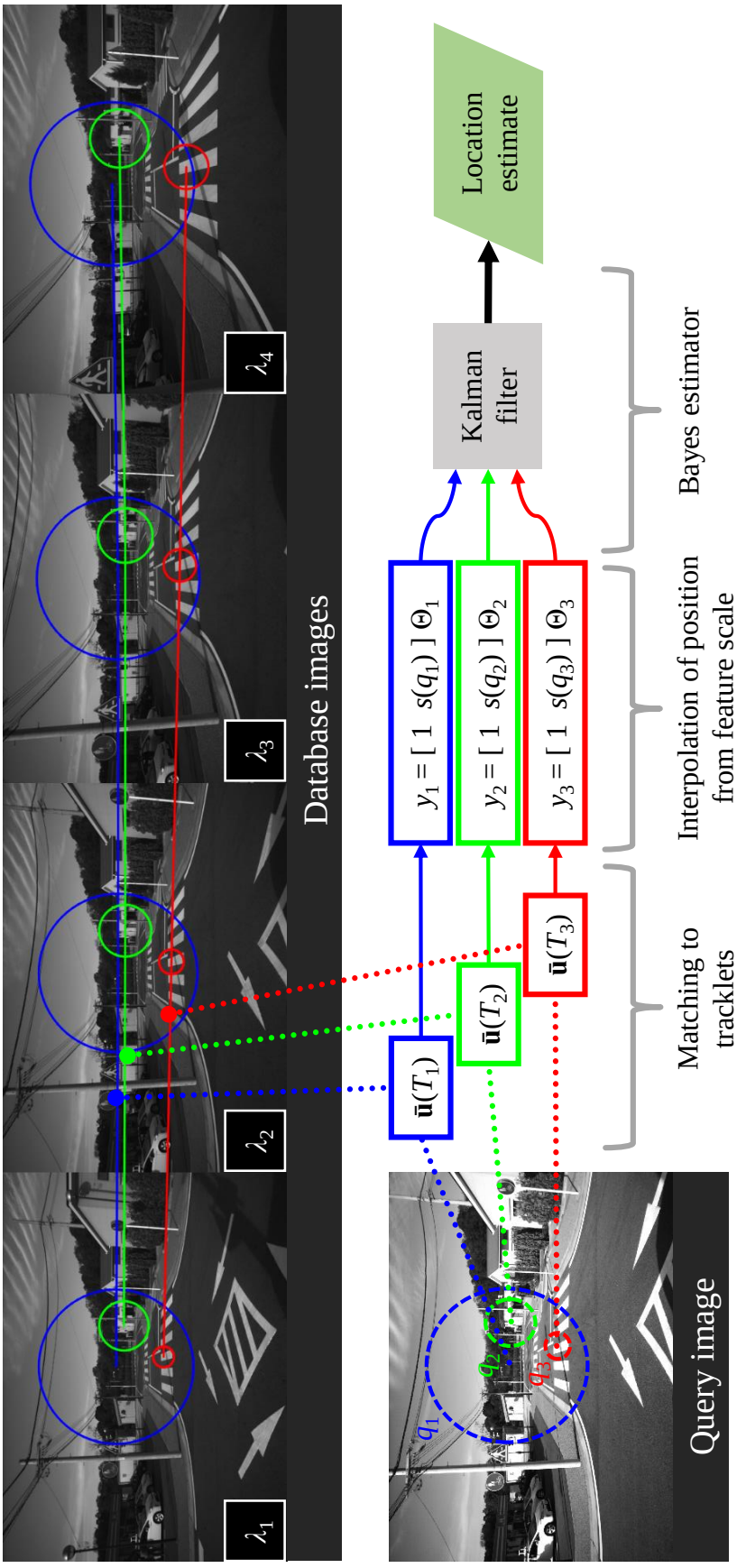


Figure 4.2: Visual localization with feature-scale tracklet regression coefficients overview. Example query image and a series of database images displaying three example tracklets, each highlighted in a different color. The query image features, matched to the tracklets through the average descriptors, are shown matched to the tracklet line based on the interpolated position along the tracklet, calculated from the query feature scale and tracklet regression coefficients.

coefficients. Therefore, the new structure becomes:

$$\begin{aligned} T &= \{g_1, g_2, ..., g_m, ..., g_M, \Theta\}, \\ g_m &= [\lambda_m, a_{\lambda_m}, \mathbf{u}_m, s_m, \mathbf{p}_m, \ell_m], \end{aligned} \tag{4.1}$$

where each tracklet is denoted $T \in \mathcal{T}$ where $\mathcal{T}$ is the full set of database tracklets. A feature tracklet T contains a list of M pre-matched database features from sequential frames, with each $m = 1, 2, ..., M$ as the index along the tracklet. Here λ_m is the database image index which refers to the database image containing the feature g_m, and a_{λ_m} is the feature index within the corresponding λ_m. The descriptor of feature g_m is denoted as $\mathbf{u}_m$, s_m is the feature scale, and $\mathbf{p}_m$ is the vector containing the feature pixel positions. The longitudinal distance along the current database segment is denoted ℓ_m, and defines the capture location of the feature g_m. The scale regression coefficients are denoted Θ, and their construction is described below in Section 4.3.1.2.

Note that the capture location ℓ_m has a scalar value per feature. While the capture locations are actually two-dimensional coordinates, in the research presented in this chapter they are converted into one-dimensional values to represent the length in meters along the current database segment. Where the road splits and new segments form, the segments are annotated appropriately. This notation is suitable because lateral motion of the vehicle is constrained and of less interest than the longitudinal position along the road. While it is also possible to perform a multivariate linear regression to build a per-tracklet model of feature-scale changes with capture position [73], a simpler bivariate least-squares regression is made possible by resolving the capture positions into a single longitudinal distance along the current database segment.

4.3.1.2 Regression coefficient calculation

Feature-scale tracklet regression is performed on each tracklet to calculate the regression coefficients Θ, which describe the linear fit of the relationship between scale and capture position. The regression process allows the inclusion of scale information from all corresponding feature points in the database, giving a more robust position estimate per tracklet and allowing filtering based on the coefficient of determination.

If $\boldsymbol{\ell}$ is the vector of capture coordinates $\ell_1, \ell_2, ..., \ell_M$ of M consecutively matched database features within the tracklet, and $\mathbf{S}$ is the $M \times 2$ design matrix containing the corresponding feature-scale row vectors $\mathbf{s}_m = \begin{bmatrix} 1 & s_m \end{bmatrix}$, then the coefficients for the linear regression Θ can be calculated by ordinary least squares as

$$\Theta = (\mathbf{S}^{\mathrm{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathrm{T}}\boldsymbol{\ell}. \tag{4.2}$$

The coefficients calculated for each tracklet form a parametrization which allows a direct capture position estimate when supplied with a corresponding query feature's scale.

Additionally, the residual sum-of-squares, $\mathrm{SS}_{\mathrm{res}}$ of the set of feature positions with respect to scale can be calculated as

$$\mathrm{SS}_{\mathrm{res}} = \sum_{m=1}^{M} (\ell_m - \mathbf{s}_m\Theta)^2 \,. \tag{4.3}$$

Similarly, the total sum-of-squares $\mathrm{SS}_{\mathrm{tot}}$ can be calculated as

$$\mathrm{SS}_{\mathrm{tot}} = \sum_{m=1}^{M} \left(\ell_m - \bar{\ell}\right)^2 , \tag{4.4}$$

where $\bar{\ell}$ is the mean of the capture locations $\ell_1, \ell_2, ..., \ell_M$. A measure of variability in the scale to position relationship for each tracklet is the ratio of the residual sum-of-squares to the total sum-of-squares, known as the coefficient of determination, R^2.

This can be calculated as

$$R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}}. \tag{4.5}$$

This ratio determines the closeness of the regression fit and can be used to prune tracklets that display highly non-linear scale change with position. The R^2 value is between zero and one, with one representing a perfect linear fit. Tracklets with an R^2 over a chosen threshold can be selected for inclusion in the database. It was found in experiments that a value of 0.8 is an effective threshold for filtering tracklets with mismatches.

4.3.1.3 Feature descriptor averaging

This research also proposes the averaging of feature descriptors within each tracklet to reduce the database size. Once the features have been collected into feature-scale tracklets, they make up a string of matched features of increasing scale and varying image position. However, they all have similar descriptors which led to them being matched to each other. These descriptors can be combined into one descriptor per tracklet to save database space.

From Equation (4.1), each feature-scale tracklet $T \in \mathcal{T}$ contains a string of features, $g_1, g_2, ..., g_m, ..., g_M$. Each feature g_m contains a descriptor $\mathbf{u}_m$. The descriptor averaging process assigns a single descriptor $\bar{\mathbf{u}}(T)$ as

$$\bar{\mathbf{u}}(T) = \frac{1}{M} \sum_{m=1}^{M} \mathbf{u}_m, \tag{4.6}$$

which provides the average descriptor for each tracklet T. When matching query image features to the features in the database, the query image feature only needs to be matched to each tracklet once, rather than multiple times for individual feature descriptor comparisons within the tracklet. Each tracklet containing only one feature descriptor also substantially reduces the database size.

4.3.2 Localization

This section describes how the proposed method achieves localization of input query image frames. As with the topological localization method presented in Chapter 3, before a query image frame is localized relative to the database tracklets, the system must retrieve the relevant sections of the database. The database may be retrieved as a single file with indexed tracklets, or for very large databases, split into many files of a more manageable size for reading or streaming into the localization system. The current vehicle position estimate is used to select candidate tracklets from the database based on the capture positions which they cover. Where no current position estimate exists, to commence localization, coarse positioning information from GPS is used to determine the general region, and tracklets covering close-by locations are included as potential matches to the original query frame features. The approximate localization from GPS could be replaced with a vision-only approach by sequence matching over dimension reduced images [26]. This would remove the dependence on GPS, but would potentially require a few minutes to capture a sequence suitable for general position recognition. The increase in database size would be small as only a few bits are required for sequence template matching.

SIFT features are extracted from the query image, and then matched to the recorded features of the candidate database tracklets. The first few frames are localized using visual measurements only, described below in Section 4.3.2.1. Once the motion of the vehicle is established from the first few visual measurements, the localization process is performed within a Kalman filter implementation of a Bayes estimator (Section 4.3.2.2).

4.3.2.1 Visual location measurement

The Bayes estimator described in Appendix B requires a measurement update. In this research, this is provided by a visual measurement from combining individual location measurements from the query features matched to database tracklets.

When N features from a query image are matched to a subset of database feature-scale tracklets, each matched query feature q_i and corresponding T_i (where $i = 1, 2, ..., N$) create a measurement of the query capture position, y_i, using the database coefficients determined for each tracklet in Equation (4.2) and the query feature scale $s(q_i)$ as

$$y_i = \begin{bmatrix} 1 & s(q_i) \end{bmatrix} \Theta_i. \tag{4.7}$$

The results of the individual measurements can be combined to form a single visual measurement by averaging their values as

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i. \tag{4.8}$$

The variance of the overall visual measurement can be calculated as

$$\sigma_{\bar{y}}^2 = \frac{1}{N} \sum_{i=1}^{N} (\bar{y} - y_i)^2 . \tag{4.9}$$

One of the contributions of the proposed method is the recognition that the multiple measurements form a Gaussian distribution which is suitable for inclusion in a Bayes estimator as a measurement model. When combined, the query location measurements form the following Probability Density Function (PDF):

$$P(\bar{y} \mid x) = \frac{1}{\sqrt{2\pi\sigma_{\bar{y}}^2}} \exp\left(-\frac{(x - \bar{y})^2}{2\sigma_{\bar{y}}^2} \right), \tag{4.10}$$

where x is the actual capture position of the query image, $\bar{y}$ is the mean of the individual visual location measurements y_i, and $\sigma_{\bar{y}}^2$ is the variance of the distribution.

4.3.2.2 Kalman filtering

If the vehicle motion is assumed to follow a linear constant-velocity model, the system has a Gaussian state transition model and measurement model, so a Kalman filter can be used to avoid the difficult calculations of finding the maximum *a posteriori*

estimate of the fundamental Bayes estimator (see Equation (B.1) and Equation (B.2) in Appendix B).

Following the notation used in Appendix B, in this section k denotes the time step for each predict/update cycle. The state transition model predicts the location of the vehicle $\hat{x}_{k|k-1}$ and associated variance $\hat{\sigma}^2_{k|k-1}$ based on previous states as follows:

$$\begin{aligned} \hat{x}_{k|k-1} &= \hat{x}_{k-1|k-1} + \Delta\hat{x}_k + w_k, \\ \hat{\sigma}^2_{k|k-1} &= \hat{\sigma}^2_{k-1|k-1} + \sigma^2_{w_k}, \end{aligned} \tag{4.11}$$

where w_k is the transition model noise, and $\Delta\hat{x}_k$ is the predicted location change determined by the chosen motion model. Assuming that no vehicle odometry information is available, a constant-velocity state transition model is used. The new vehicle location PDF is approximated to be a Gaussian with mean μ and variance σ_w. The previous distance covered between query image frames is calculated, and adjusted for timing differences in the next time step as

$$\Delta\hat{x}_k = \left(\frac{\hat{x}_{k-1|k-1} - \hat{x}_{k-2|k-2}}{t_{k-1} - t_{k-2}}\right)(t_k - t_{k-1}) \tag{4.12}$$

where t denotes the system time. The calculated $\Delta\hat{x}$ is added to the last predicted vehicle location such that $\mu = \hat{x}_{k-1|k-1} + \Delta\hat{x}_k$. This provides the following Gaussian probability density function

$$P(\hat{x}_k \mid x_{k-1}) = \frac{1}{\sqrt{2\pi\sigma^2_w}} \exp\left(-\frac{(x_{k-1} - \mu)^2}{2\sigma^2_w}\right), \tag{4.13}$$

which forms the state transition model for the Kalman filter.

The key component of this localization method is the visual measurement step, which provides the measurement update in the Kalman filter. The candidate database tracklets are selected based on the previous position $\hat{x}_{k-1|k-1}$. A tracklet is selected for matching if the span of capture positions of the tracklet $\ell_1, \ell_2, ..., \ell_M$ intersect with the query region between $\hat{x}_{k-1|k-1} - \rho\Delta\hat{x}_k$ and $\hat{x}_{k-1|k-1} + \rho\Delta\hat{x}_k$, where ρ is a factor to determine the range of the database tracklets to include for potential matching.

Increasing the value of ρ increases the number of tracklets included in the matching process, which can give better position estimates at the expense of an overall increase in matching time. It was found in experimental testing that a ρ value of 2.0 usually encompassed all tracklets which would produce inlier position estimates.

Query images are then matched to the resulting set of tracklets by comparing the tracklet average descriptors to the query image feature descriptors, using the same feature matching technique as employed in the database construction stage. The final set of matches are then used to complete the measurement update, using the distribution of the combination of individual tracklet estimates $\bar{y}_k$ provided by Equation (4.8) and Equation (4.9) as follows:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(\bar{y}_k - \hat{x}_{k|k-1}), \\ \sigma^2_{k|k} &= (1 - K_k)\hat{\sigma}^2_{k|k-1}, \end{aligned} \tag{4.14}$$

where K_k is the Kalman gain, calculated at each step as

$$K_k = \frac{\hat{\sigma}^2_{k|k-1}}{\hat{\sigma}^2_{k|k-1} + \sigma^2_{\bar{y}_k}}. \tag{4.15}$$

Equation (4.14) provides the estimated position and variance for the query image frame, in terms of the distance along the current road segment. The location estimate $\hat{x}_{k|k}$ is then used to start the next prediction stage from Equation (4.11).

4.4 Experiments

To evaluate the performance of the proposed method, the traffic environment dataset used in Section 3.4.3 could not be employed, as it contains no metric localization information for the image frames. Therefore, the driving-school dataset used in Section 3.4.4 was used to evaluate the metric localization performance of the method presented in this chapter.

This section starts with Section 4.4.1 providing a summary of the comparative and proposed methods that were evaluated in these experiments. The process and results of the topometric localization experiments are presented in Section 4.4.2.

4.4.1 Evaluated methods

To evaluate the performance of the proposed system, five localization methods were compared. The tested methods are summarized as follows:

1. **Proposed method 1: Feature-scale tracklet regression and Kalman filtering.** This method uses all of the proposed techniques introduced in this chapter. It makes use of feature-scale regression to create coefficients for per-tracklet localization measurements as described in Section 4.3.1.2 and Section 4.3.2.1. A Bayes estimator in the form of a Kalman filter (Section 4.3.2.2) is employed to give a final location estimate for each query image frame.

2. **Proposed method 2: Feature-scale tracklet regression** This method uses all of the proposed techniques introduced in this chapter up to the Bayes estimator for position estimation with a Kalman filter. The feature-tracklet regression models described in Section 4.3.1.2 are employed to create per-feature interpolated location measurements using the method described in Section 4.3.2.1. Localization is performed without a state transition model and proceeds with the average of the visual location measurements making up the location estimate.

3. **Comparative method 1: Feature-scale tracklet image matching.** This is the method proposed in Chapter 3. The method is used to perform topological localization in the form of image matching of query and database frames, and the metric localization information from the predicted closest match is applied directly to the query frame.

4. **Comparative method 2: Feature-scale DTW.** This is comparative method 1 in Chapter 3. It is based on previous work [65] and uses the average scale

change between matched features of individual images as a cost measure. This method is implemented using DTW to compare image streams. Like the proposed method, it uses matched feature-scale comparisons; however, it does not use the pre-matching and tracklet construction techniques presented in Chapter 3, or the feature-scale regression and Bayes-filter approaches presented in this chapter. Topometric localization is performed by using the localization information from the predicted closest database frame match directly.

5. **Comparative method 3: WISURF-DTW.** This is comparative method 3 in Chapter 3. It is a modified implementation of WISURF [27] for image matching. The WISURF method creates a single SURF descriptor for each image, and chooses a database image match based on descriptor distance. Instead of using a Bayesian filter [27], here image matching is performed using the WISURF image descriptor distance and DTW to remove the dependence on motion estimation for localization.

Table 4.1 provides a comparison of the techniques used in the evaluated methods.

4.4.2 Driving-school dataset experiment

This experiment provided an evaluation of the proposed method, testing the contributed concepts of feature-scale regression and localization with a Bayes filter individually to examine their contribution to topometric localization performance. The driving-school dataset of Chapter 3 was used in this evaluation.

4.4.2.1 Image capture

The vehicle configuration remained the same as for the experiments in Section 3.4.4.1 of Chapter 3, and once again only the front-facing camera images were used. The captured images were downscaled from $2{,}400 \times 2{,}000$ pixels to 600×500 pixels. The localization information supplied by the MMS for each image was used for

Table 4.1: Summary of the techniques used in the evaluated methods.

Method	Feature-scale comparison	DTW	Tracklets	Tracklet regression	Kalman filter
Proposed 1: Feature-scale tracklets, scale regression + Kalman filter	✓	—	✓	✓	✓
Proposed 2: Feature-scale tracklets, scale regression	✓	—	✓	✓	—
Comparative 1: Feature-scale tracklets, image matching only	✓	—	✓	—	—
Comparative 2: Feature-scale DTW, image matching only	✓	✓	—	—	—
Comparative 3: WISURF-DTW, image matching only	—	✓	—	—	—

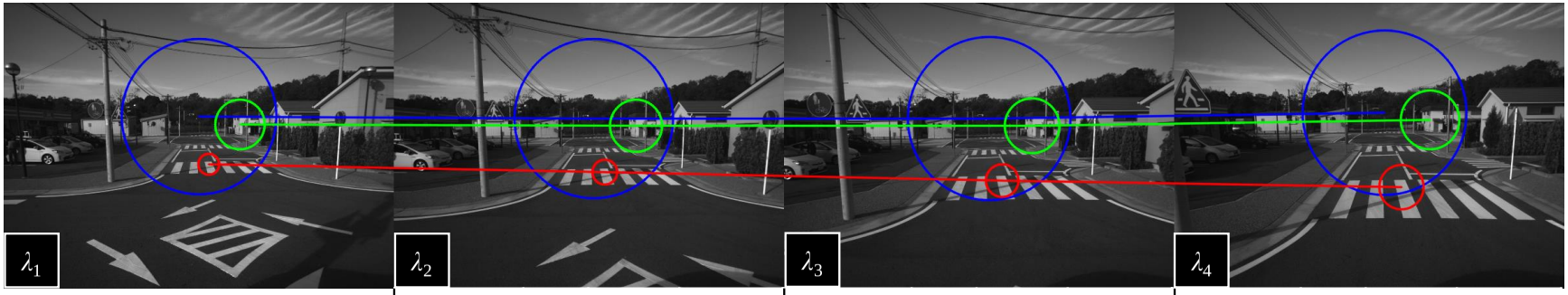
tracklet construction in the database streams, and provided a ground-truth for the query stream images to allow performance evaluation. For this experiment, only the same lane sections of the dataset from Section 3.4.4.1 were employed. The capture specifications are the same as summarized in Table 3.3 of Section 3.4.4.1 in Chapter 3.

4.4.2.2 Localization performance

Databases were constructed for the proposed and comparative methods, and localization was performed through sections of the dataset. The actual contents of the tracklets used as an example in Figure 4.2 are displayed in Figure 4.3 (with the exception of the calculated regression coefficients, pixel locations, and descriptors). The descriptor averaging that was proposed in Section 4.3.1.3 was applied to the two proposed methods, resulting in at least a 50% reduction in the database size when compared to comparative methods 1 and 2 which also use image feature descriptors. The WISURF-DTW method of comparative method 3 uses a single descriptor per image frame so has a significantly smaller database than all other evaluated methods. The database size information for the evaluated methods is shown in Table 4.2.

The same machine used for the experiments in Chapter 3 was used to perform the localization process. Again, without any GPU implementation, multi-threading nor optimization of any kind, localization was performed at approximately 15 Hz. The application of scale regression coefficients for position estimation and the Kalman filter made a negligible difference to the processing time when compared to feature extraction and matching.

Localization accuracy was evaluated using the MMS localization data. For the query image stream, localization information provided the ground-truth. Topometric localization results in terms of the absolute error are presented in Figure 4.4 and Table 4.3. All tested areas were where the query and database image frames came from the left-hand lane. Proposed method 1 achieved an average localization error of 0.33 m, representing a 46% improvement over the feature-scale tracklets image matching



	λ_1	a_1	s_1	ℓ_1		λ_2	a_2	s_2	ℓ_2		λ_3	a_3	s_3	ℓ_3		λ_4	a_4	s_4	ℓ_4
g_1	130	1	6.56	248.80	g_2	131	1	7.84	250.74	g_3	132	1	10.17	252.70	g_4	133	1	13.61	254.73
g_1	130	2	15.55	248.80	g_2	131	2	16.30	250.74	g_3	132	2	16.92	252.70	g_4	133	2	18.13	254.73
g_1	130	3	47.62	248.80	g_2	131	3	48.85	250.74	g_3	132	3	50.07	252.70	g_4	133	3	50.76	254.73

Figure 4.3: Example of the contents of the feature-scale tracklets from Figure 4.2. The calculated regression coefficients, pixel locations and descriptors have been omitted for clarity.

Table 4.2: Database sizes. The best results are shown in bold.

Method	Database size [KB/m]
Proposed 1: Feature-scale tracklets, scale regression + Kalman filter	40.19
Proposed 2: Feature-scale tracklets, scale regression	40.19
Comparative 1: Feature-scale tracklets, image matching only (Chapter 3)	80.96
Comparative 2: Feature-scale DTW, image matching only	191.23
Comparative 3: WISURF-DTW, image matching only	**1.42**

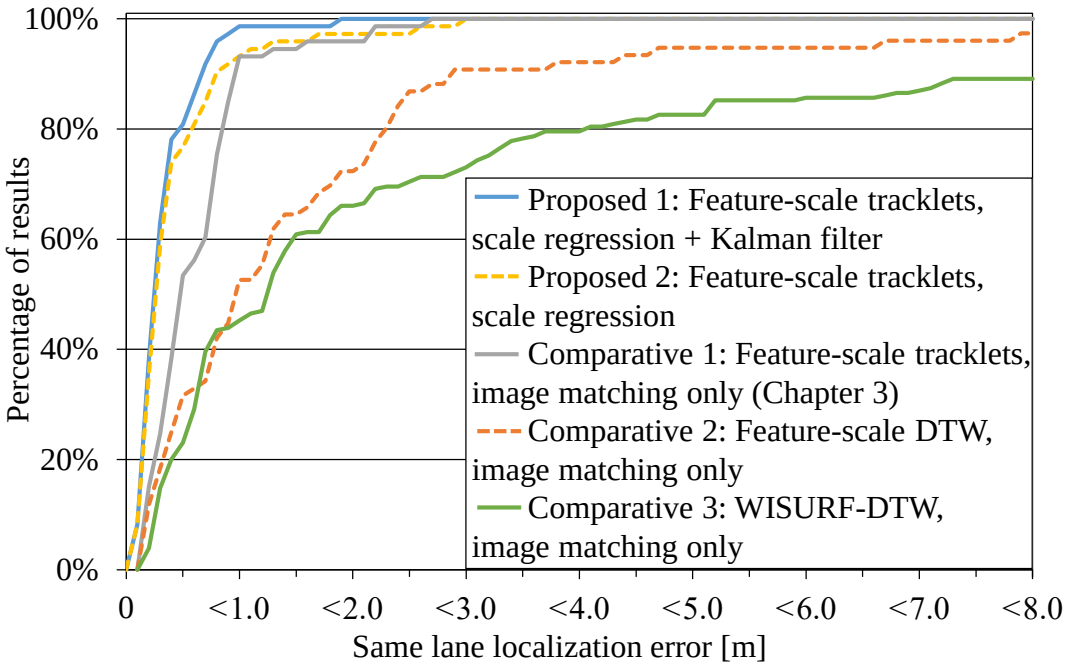


Figure 4.4: Topometric localization results from the driving-school dataset. The graph shows the percentage of localization results within each metric error level.

Table 4.3: Summary of the topometric localization results from the driving-school dataset. The best results are shown in bold.

Method	Average error [m]	Standard deviation [m]	Maximum error [m]
Proposed 1: Feature-scale tracklets, scale regression + Kalman filter	**0.33**	**0.27**	**1.82**
Proposed 2: Feature-scale tracklets, scale regression	0.42	0.49	2.92
Comparative 1: Feature-scale tracklets, image matching only (Chapter 3)	0.61	0.46	2.66
Comparative 2: Feature-scale DTW, image matching only	1.00	1.32	11.04
Comparative 3: WISURF-DTW, image matching only	3.71	1.68	14.33

method alone. Compared to proposed method 2, a 21% improvement was observed from the application of the Bayes filter for localization. The filter also managed to reduce the localization variance and maximum error, as the motion model provides smoothing of the localization results.

4.5 Discussion and analysis

This section provides a discussion about the evaluated methods. The effectiveness of the proposed method is analyzed, and its performance is compared to the comparative methods is discussed. Further exploration of the assumptions upon which the proposed method is based are presented. This includes experiments to provide justification for the use of a normal distribution for the visual location measurement update, and the linearity assumptions in both feature-scale change with capture position and the system model for the Kalman filter. Database size and camera configuration are also discussed.

4.5.1 Evaluated methods

Apart from the comparative methods 2 and 3, all of the evaluated methods used feature-scale tracklets. As was also observed in the experiments in Chapter 3, the methods based on feature-scale tracklets achieved significantly better image-matching performance than the methods without, so comparative method 1 was the most effective of the comparative methods tested. The differences between the three methods based on feature-scale tracklets are more subtle. Using regression within the database (proposed method 2) made a significant 31% improvement in localization error over the best image matching only method (comparative method 1). This illustrates the effectiveness of the use of feature scale to interpolate position within a group of pre-matched features from consecutive database frames. The large number of estimates per query frame makes the visual localization system ideally suited to inclusion in a Bayes estimator, as variance can be easily calculated over the samples.

A Kalman filter is a natural choice for including a simple motion model in the localization estimate, and also allows the inclusion of control inputs and other sensor localization measurements if available. The simple constant-velocity motion model that was applied smoothed the localization results enough to enable a 21% improvement in localization accuracy for proposed method 1 over proposed method 2. It also reduced the variance of the localization error by 70%, as shown in Table 4.3 (where standard deviations are displayed instead of variance for unit consistency. The corresponding drop in standard deviation is 45%).

4.5.2 Feature-scale tracklet regression assumptions

The visual localization method proposed in this chapter relies on two assumptions. The linear regression performed on the tracklets as described in Section 4.3.1.2 assumes that feature scale varies linearly with capture position. The strength of the linear fit can be evaluated using the coefficient of determination, R^2 (Equation (4.5)). In these experiments, it was found that most tracklets containing only correct matches have an R^2 very close to one, suggesting that the linear fit is a good model for feature-scale change with forward motion. The regression lines of the example tracklets from Figure 4.2 and Figure 4.3 are shown in Figure 4.5. The coefficients of each tracklet are displayed in the linear-fit equation for each line, together with the R^2 value. The coefficient of determination R^2 is an important filter measure in the proposed method. By only using tracklets containing an R^2 above a certain threshold, features which do not show stable linear increase in scale with capture position are effectively ignored. This also results in the removal of tracklets containing mis-matched features. Unlike pose-estimation methods that use geometric constraints between matched features, the proposed method can generate a localization estimate even when only a few tracklets are matched within the query image features; in fact, even one matched tracklet will produce an estimate. The coefficient of determination is therefore useful to select a smaller group of feature matches which provide quality location estimates.

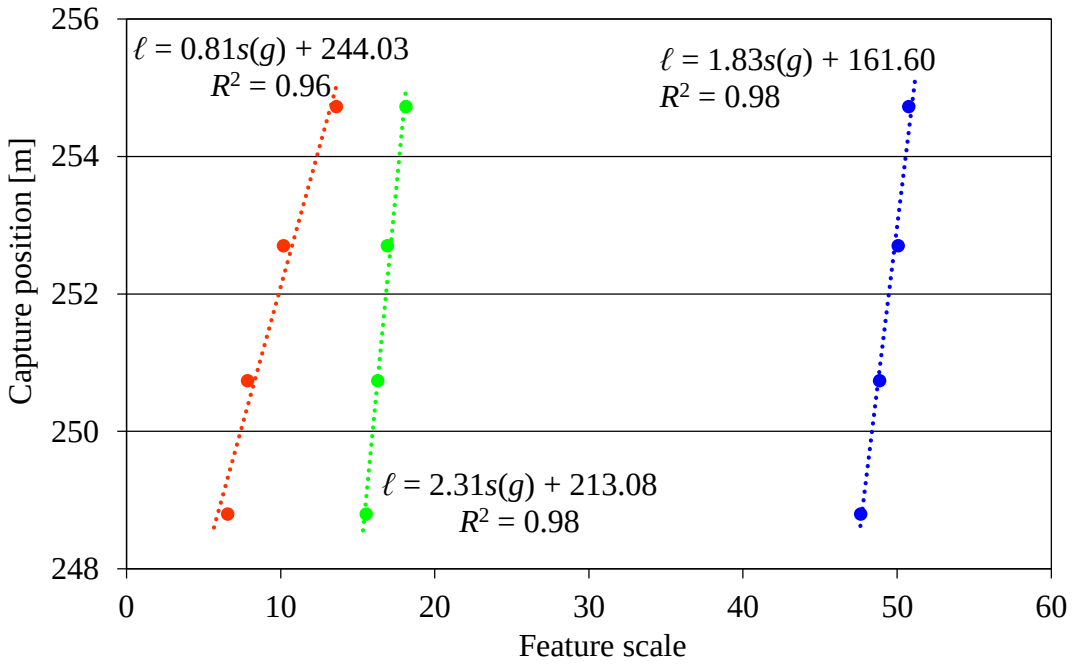


Figure 4.5: Linear regression fit of the example tracklets from Figure 4.2 and Figure 4.3. The resulting regression coefficients and R^2 values are shown alongside the corresponding regression line of each tracklet.

The second assumption is that the distribution of the per-tracklet visual measurements is a zero-mean Guassian. In order to confirm that the noise distribution v_k of y_k is a zero-mean normal distribution such that $v_k \sim \mathcal{N}(0, \sigma^2_{y_k})$, the location estimate error frequencies were plotted over many localization steps. The resulting distribution, shown in Figure 4.6, proves that the noise distribution is indeed closely Gaussian.

The Gaussian nature of the visual position estimate is the reason that a Bayes estimator was chosen for localization. Some of the related works [25, 26, 66] use a DTW framework for localization estimation instead. These methods perform sequence matching between the query and database image streams, minimizing a cost function to determine the most likely location of the vehicle. The proposed method is related in that it creates a position estimate for each database feature by minimizing a difference cost, but does this relative to a regression line constructed for each database tracklet rather than discrete image-to-image matching. By pooling all of the individual feature estimates into a normally distributed PDF, a measurement update

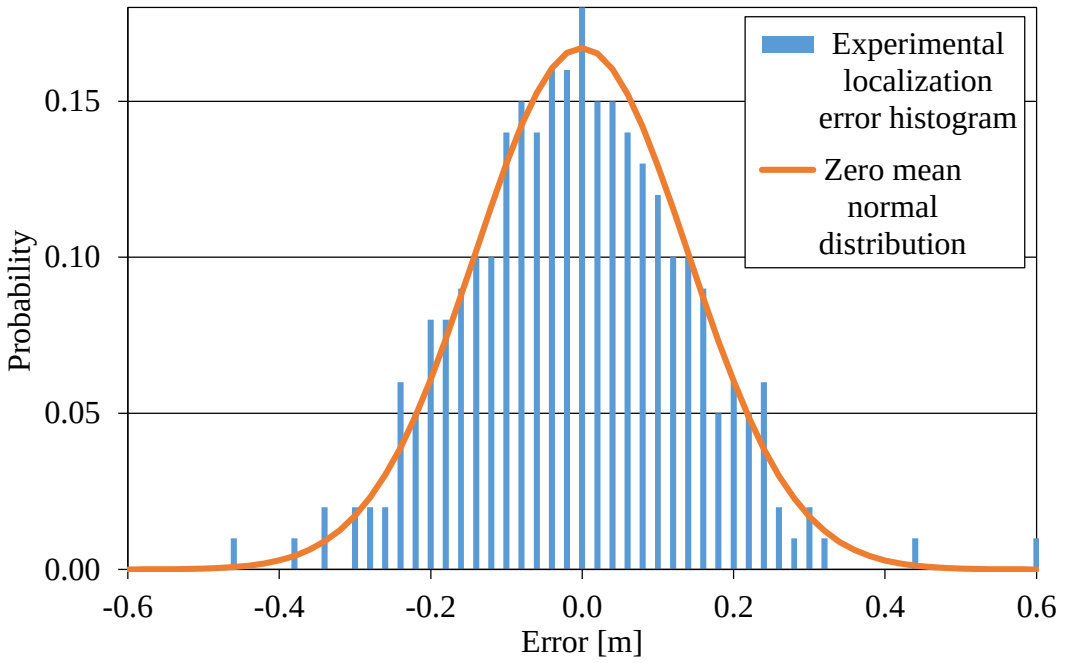


Figure 4.6: Frequency histogram of the individual tracklet localization error v_k compared with the standard, zero-mean normal distribution with variance σ_z^2.

ideally suited for a Bayes filter is created. This allows easy inclusion of a motion model for smoothing and improving the localization results. DTW is a suitable estimator when interpolation between image frames is not used, but for localization precision that is not limited by the database image spacing, some form of regression is required.

If the system is assumed to be linear, a Kalman filter is an optimal estimator. A non-linear estimator, such as an Unscented Kalman Filter (UKF) [102] or particle filter [50] can also be applied to this method but adds complexity, with potentially limited benefit. This is investigated in the following section.

4.5.3 Estimator evaluation

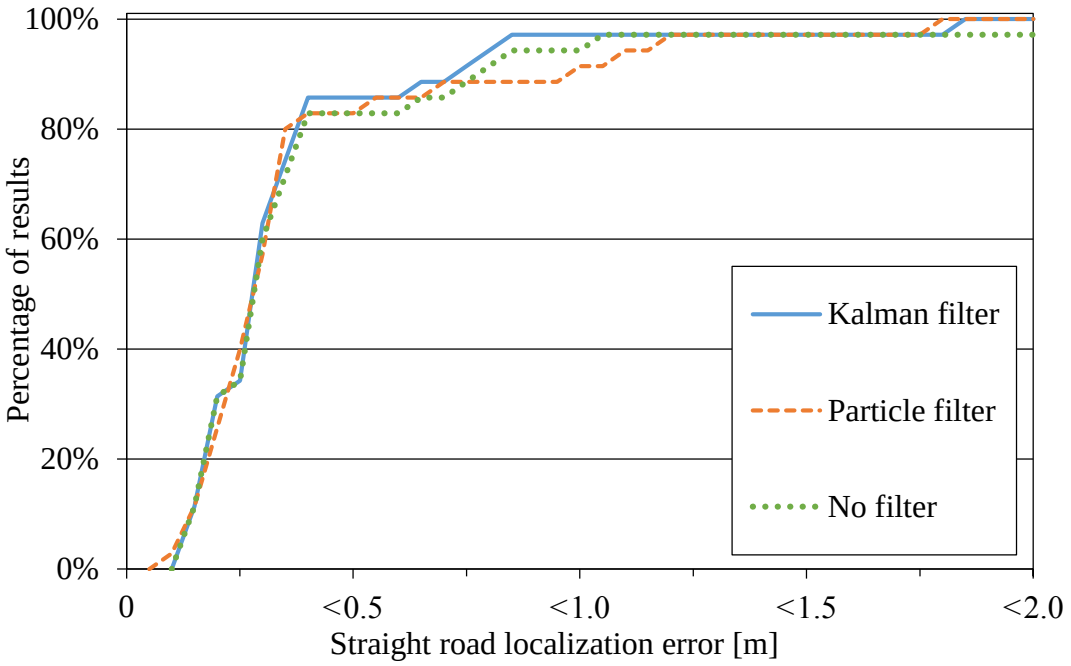
The Kalman filter used in the proposed method is optimal for linear problems with Gaussian noise, but a vehicle may not strictly adhere to these properties. Especially where cornering occurs, the linear system assumption may no longer be appropriate.

Therefore, the suitability of the Kalman estimator was evaluated by implementing a particle filter for comparison. A particle filter is also a Bayes estimator, but since it uses Monte Carlo random sampling for measurement and transition, limited assumptions are made about the underlying system model. This makes it an effective estimator for highly non-linear systems, but it is much more computationally expensive than a Kalman filter and may also suffer from divergence due to the random sampling nature of state propagation. A simple particle filter with Sampling Importance Re-sampling (SIR) [50] was implemented. For this testing, the same motion model as described in Section 4.3.2.2 was employed, with various numbers of particles. It was found that a few thousand particles were sufficient for consistent and reproducible results, and 4,000 particles were used for the results presented here.

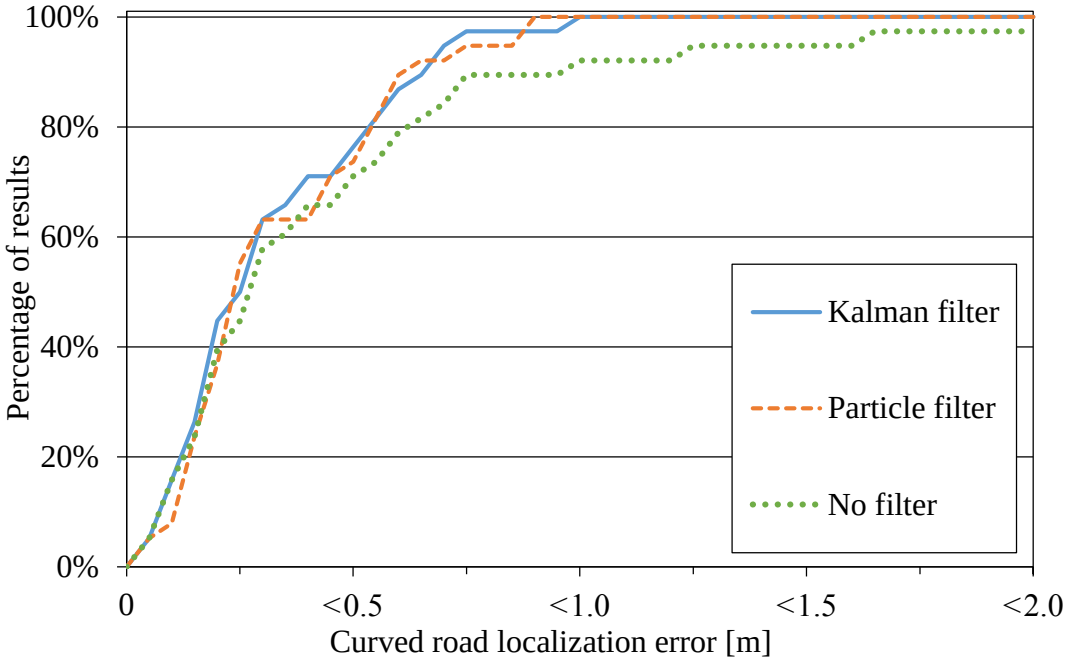
The comparison of the particle filter estimator and proposed the Kalman estimator is presented in Figure 4.7 and Table 4.4. To determine if the particle filter has advantages over the Kalman filter in areas of the dataset where the vehicle maneuvers turns, the evaluation results are split into straight areas of road (Figure 4.7(a)) and areas where a corner was traversed (Figure 4.7(b)). The results show that there was no significant difference in localization accuracy when using a particle filter. The fact that the particle filter did not provide any localization performance improvement, despite being a non-linear estimator, confirms that the computationally simpler Kalman filter is a suitable choice of estimator for this system.

4.5.4 Database size

Another important consideration for an automotive localization system is the database size. Many visual systems that use features have an impractically large visual database even after pre-processing. Although the feature-tracklet database is of respectable compactness when compared to denser feature-point methods [28, 29], the ability to share a single descriptor over consecutively matched features (as described in Section 4.3.1.3) makes a significant difference to the database size. The descriptor component of the database is reduced by at least 60%, which equates to an overall



(a) Straight road areas.



(b) Curved road areas.

Figure 4.7: Localization error rates of the tested estimators, comparing the results of the proposed Kalman filter, comparative particle filter, and no filter on both straight (a) and curved (b) sections of road.

Table 4.4: Summary of the localization results with different estimators. The best results are shown in bold.

Road type	**Estimator**	**Average error [m]**	**Standard deviation [m]**	**Maximum error [m]**
Straight	Kalman filter	**0.31**	**0.23**	0.96
	Particle filter	0.32	**0.23**	**0.88**
	No filter	0.44	0.54	2.92
Curved	Kalman filter	**0.35**	**0.31**	1.82
	Particle filter	0.38	0.36	**1.80**
	No filter	0.39	0.44	2.58

halving of the database size. The resulting database fits approximately 25 m into 1 MB of storage. While it would be possible to compress this database into a much smaller file size —in this implementation, the descriptor values were stored as raw text— even in its current format, downloading the database in real time at 100 km/h would use less bandwidth than streaming a standard 720p movie. This should be obtainable by modern mobile data networks.

4.5.5 Camera considerations

In the experiments presented in this chapter, the camera type and mounting system used for both query and database images were the same. While it would be desirable to use a variety of configurations to test robustness, this was not possible with the hardware available. However, like the method presented in Chapter 3, because the method proposed in this chapter uses only feature scale and does not rely on accurate pixel position like pose-estimation methods do, it should be robust to alternative mounting positions and uncalibrated cameras. Knowledge of the camera's focal length should be sufficient to allow localization using a different camera from the one used in the database construction.

The core of the proposed system is the use of the scale of feature points. Feature scale is calculated by constructing a Gaussian scale pyramid to determine feature points which present a local maximum over multiple scales. Gaussian scaling is analogous to blurring, so blurred images have an affect on the perceived scale of features. Therefore, the method proposed in this research requires sharp images. Blurred images can be caused by vehicle motion or camera shake, so the camera mounting method must be secure. A camera capable of capturing well-lit images with short exposure times is also important for minimizing motion blur.

4.6 Summary

In this thesis, three vehicle localization types using monocular cameras are explored. As introduced in Chapter 1, these are topological localization, topometric localization, and direct metric localization. These localization types provide progressively refined ego-localization, with topological localization giving the current general vehicle position relative to a map, topometric localization positioning the vehicle metrically in a coordinate frame, and direct metric localization performing a full 6DOF position and pose estimate. In this chapter, a topometric methodology for positioning the current vehicle position in a pre-constructed database based on visual similarity was presented. This solves the problem of determining the metric location of a query frame within the continuous plane of a coordinate system.

The methodology, experiments, and analysis provided in this chapter presented an effective system to perform topometric localization of a vehicle using a single camera and a pre-constructed database. The proposed database construction method makes use of the continuous scale property of extracted image features, and performs a linear regression fit for feature-scale change with capture position within each feature-scale tracklet. This allows query images to be metrically localized by determining their estimated capture location along each feature-scale tracklet regression line, based on the scale of individual query features. Combining these measurements over all tracklets provides a visual location measurement in the form of a PDF. This can be incorporated into a Bayes filter, such as the Kalman filter proposed in this research, to provide continuous and robust localization estimation.

The experimental results show a large improvement in localization accuracy when compared to methods that apply capture locations of database frames to matched query frames from topological localization. Even without the proposed Kalman filter, the average localization error decreased by 31 % when compared to the best topological image matching results (using the method described in Chapter 3). In the experiments, the use of a Kalman filter resulted in a further 21 % decrease in average localization error to 0.33 m and a 70 % reduction in variance was observed.

Grouping features into tracklets also allows averaging of descriptors within each tracklet, halving the size of the database without reducing the performance of the system.

Chapter 5

Direct Metric Localization within Sparse Voxel Maps

This thesis covers three main research topics within the theme of visual localization for vehicles using monocular vision. Chapter 3 introduced the use of feature-scale tracklets to perform topological localization for positioning within a map, and more accurate topometric localization within a two dimensional metric coordinate frame was presented in Chapter 4. This chapter presents the third research topic, on performing a full six degree of freedom (6DOF) pose and position estimate of a vehicle using direct metric localization. The database or map required for this type of localization is very different from the ones used in Chapter 3 and Chapter 4. Rather than using visual information at discrete intervals along pre-captured routes, the research presented in this chapter uses a 3D map constructed from LIDAR information. To create a relatively compact map, point clouds captured by LIDAR are combined into a sparse voxel map. Direct metric localization is performed relative to this map by using mutually shared edge information between query images and rendered views of the voxel map. This allows accurate pose estimation and 3D localization of the camera at each query image frame.

This chapter begins with a background of the research in Section 5.1. An introduction to the novel contributions is presented in Section 5.2. The proposed methodology is described in Section 5.3, followed by an explanation of the experiments and experimental results that were used to evaluate the method in Section 5.4. The results are further discussed in Section 5.5 before this chapter is summarized and concluded in Section 5.6.

5.1 Background

In Chapter 1, an introduction to vehicle localization was presented, proposing that solving the problem of ego-localization can be divided into three stages; topological localization, topometric localization, and direct metric localization. This chapter addresses the problem of direct metric localization. The aim of the research presented in this chapter is to perform a full 6DOF pose estimation of a camera from a compact 3D database. The approach is conceptually different from the methods presented in Chapter 3 and Chapter 4, which used databases constructed from features extracted from visual database images. Direct metric localization requires the generation of a database image at any possible position and viewing angle, for comparison with the query image from the camera to be localized. As it is impossible to capture database images at all positions and viewing angles, a 3D database must be constructed that can then be rendered at any virtual camera position and camera pose. Recursively rendering and comparing generated images with the query image through a visual dissimilarity measure allows the calculation of the query camera position and pose, relative to the database coordinate frame.

For photo-realistic rendering for direct metric localization, a large amount of information is required and must be stored in a 3D database. The research introduced in this chapter is based around a compact voxel map which contains limited information. It is shown here that this information is sufficient for creation of a relatively small database, while allowing accurate direct metric localization.

5.2 Contributed concepts

This section briefly summarizes the novel contributions of the research described in this chapter. The research presented here contains two main contributions; the use of a compact 3D voxel map which contains much less information than most of the state-of-the-art methods, and a novel objective function which uses mutual edge information to perform pose optimization and localize the query image.

5.2.1 Concept 1: Sparse voxel map for visual localization

As direct metric localization methods usually require very large databases to enable scene rendering, this research aims to perform direct metric localization with a minimal 3D database. The sparse voxel map used in this research is constructed from LIDAR scans performed in the database construction phase. Individual LIDAR scans captured while the vehicle is moving can be combined using techniques such as the Normal Distributions Transform (NDT) [51], and shifted into a common coordinate frame. The resulting collection of millions of points can then be down-sampled to form a coarse occupancy grid, or voxel map. While this reduces the size of the 3D map to manageable levels, it also reduces the amount of information that can be used for rendering and comparison with query image frames for localization. To help overcome this, reflectance data from the LIDAR points is used to include road markers in the map for rendering, which provide strong visual edges. The voxels of the map can be rendered as depth images, with mutual edge information used to determine the match cost between query images and rendered images within an optimization framework. Figure 5.1 shows an example rendered voxel-map scene, together with its corresponding query image. The voxel map used in this research is described in more detail in Section 5.3.1.

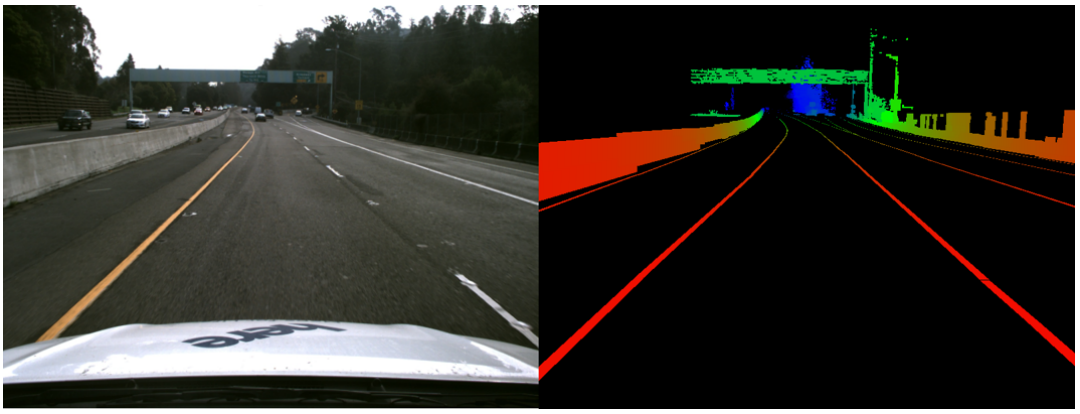


Figure 5.1: Example of a scene rendered from the voxel map, with its corresponding real camera image on the left.

5.2.2 Concept 2: Mutual edge objective function

The voxel map database described above provides no color nor texture information, so common techniques that use joint image entropies can not be easily applied to the image registration problem. Instead, this research applies an image gradient-based objective function that is minimized where mutual edges exist between rendered and real camera images. The 3D information contained in the database is rendered as a depth map to preserve edge information, so that edge areas can be used to bridge the change in modality between the rendered (virtual) images and query (real) images. A non-linear Levenberg-Marquardt [37] optimization is then applied to determine the query camera pose and location within the database coordinate frame.

The proposed system does not use lighting in the map rendering, and only mutual edges are included in the visual dissimilarity measure function, allowing camera localization which is robust to the lighting changes and occlusions typically found in traffic environments. This chapter also describes how the partial derivatives of the proposed objective function can be derived, allowing alignment of rendered voxel-map images to real camera images within a non-linear least-squares optimization framework. The objective-function formulation is explained in Section 5.3.2.2, and the derivation of the objective-function derivatives for use in the optimization process is presented in Section 5.3.2.3.

5.3 Direct metric localization within a sparse voxel map

Like the methods described in Chapter 3 and Chapter 4, the method described in this chapter requires a pre-constructed database, which means that all areas to be mapped must first be traversed with a vehicle equipped with sensors for collecting the required map data. Unlike the previous two chapters, in this research the database capture uses a LIDAR sensor rather than a camera, and the query image localization takes place across a change in modality. An overview of the system can be seen in Figure 5.2. The database construction, which builds a sparse voxel map, is described in Section 5.3.1. This is followed by an explanation of the localization phase in Section 5.3.2.

5.3.1 Database construction

The database capture is performed using a system similar to the MMS that was employed to capture the driving school dataset presented in Chapter 3 and Chapter 4, but a LIDAR scanner is the sensor used to construct the database. Individual LIDAR scans are matched using a point-cloud matching technique such as NDT [51] and combined into a dense 3D point cloud. The point cloud is then sub-sampled by partitioning into a pre-determined 3D grid, with grid areas that contain many points being marked as occupied. This makes up an occupancy grid to form the voxels of the database. Because each voxel is marked by its eight corner coordinates, the number of points that make up the database is drastically reduced. To increase the edge information that is contained in the database, road-marker lines are extracted from the reflectance data of the LIDAR scans, as the reflectance contrast between road and painted road-marker lines is strong and easily detected [48]. The lines are added to the database as a string of coordinates.

In this research, the database used was received as part of a dataset from HERE [103]. The process that was used for construction was not made available with the dataset,

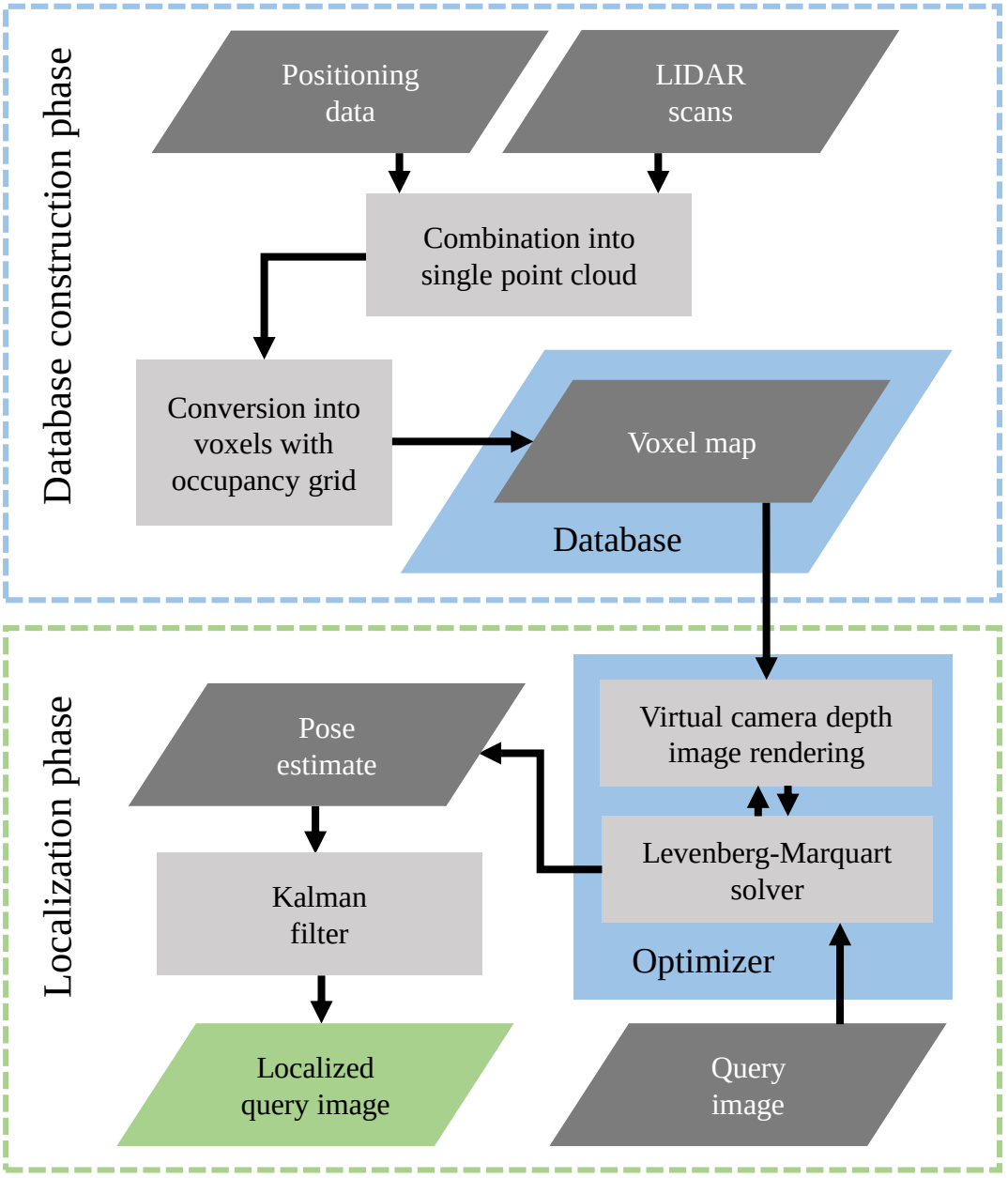


Figure 5.2: System flow diagram of the method described in this chapter.

but it is assumed that it was made using a method similar to the one proposed above. It was initially distributed as part of the University Grand Challenge competition, which featured in the Intelligent Transport Systems World Congress 2016 (ITSWC2016) [104]. More information about the HERE dataset can be found in Section 5.4.2.1 and Section 5.4.2.2

5.3.2 Localization

This section describes how the proposed method determines the camera direction and localization for each input query image. At the core of this method is a least-squares optimization framework, employing an objective function which measures the edge overlap of rendered and real images.

Starting with a rendering of the database from an initial position and pose, an objective function is used to compare the rendered image and query image. By calculating the derivatives of the objective function, the direction of minimization for each parameter in the camera's 6DOF can be determined, providing a new pose estimate for the virtual camera. The process repeats recursively until convergence. The resulting pose estimate is then used as a measurement update to complete localization within a Bayes estimator, which is similar in principle to the one presented in the previous chapter, but includes the parameters for pose estimation over 6DOF.

In the subsequent sections, the main processes of the proposed method are described. which can be summarized as objective-function formulation in Section 5.3.2.2, optimization framework in Section 5.3.2.3, and Kalman filtering in Section 5.3.2.4.

5.3.2.1 Pose optimization formulation

The extrinsic parameters of a virtual camera viewing the voxel-map scene will align with the location and pose of the real camera when the rendered and real scenes share the most overlapping information. Determining these paramters is an $\mathbb{SE}^3$ optimization problem, and changes in Euler angles are directly correlated to any objective function used so they must be included in the optimization process.

The virtual camera pose $\mathbf{r} = [t_X, t_Y, t_Z, \theta_X, \theta_Y, \theta_Z]$ that is equal to the pose of the real camera $\bar{\mathbf{r}}$, can be determined by rendering a view of the map from an initial pose, comparing the rendered image $I(\mathbf{r})$ to the real image $\bar{I}$ through a dissimilarity measure, and then performing a gradient descent or Gauss-Newton optimization to

find the parameters that make up the Euler angle representation of the camera pose that minimize the dissimilarity measure function. The pose estimation becomes:

$$\hat{\mathbf{r}} = \arg\min_{\mathbf{r}} \rho(\bar{I}, I(\mathbf{r})), \tag{5.1}$$

where $\rho(\bar{I}, I(\mathbf{r}))$ defines the dissimilarity measure. Typical dissimilarity measures, such as pixel-wise SSD and NID, are minimized when the real and rendered images are aligned.

The voxel map used in this chapter has no texture information, so the voxels are rendered as a depth map, with the color of each vertex encoding the distance to the camera. The lack of texture and sparse nature of the rendered voxel map, as well as the change in modality between the real and rendered images, make this optimization problem poorly suited to dissimilarity measures such as SSD and NID. The dissimilarity measure adopted for use with voxel map renders is described in the following section, Section 5.3.2.2.

5.3.2.2 Objective function

The objective function must provide robustness to noise and a wide convergence basin. Visual comparison of the real and rendered images such as those in Figure 5.1 suggest that the majority of shared information lies in the mutual edge areas. However, where most objective functions use some kind of pixel-wise subtraction, the predominantly empty nature of the voxel map renders will provide a poor dissimilarity measure. By combining these two observations, the proposed method uses a per-pixel multiplication as a dissimilarity measure between query images and rendered voxel-map images, as shown in Figure 5.3.

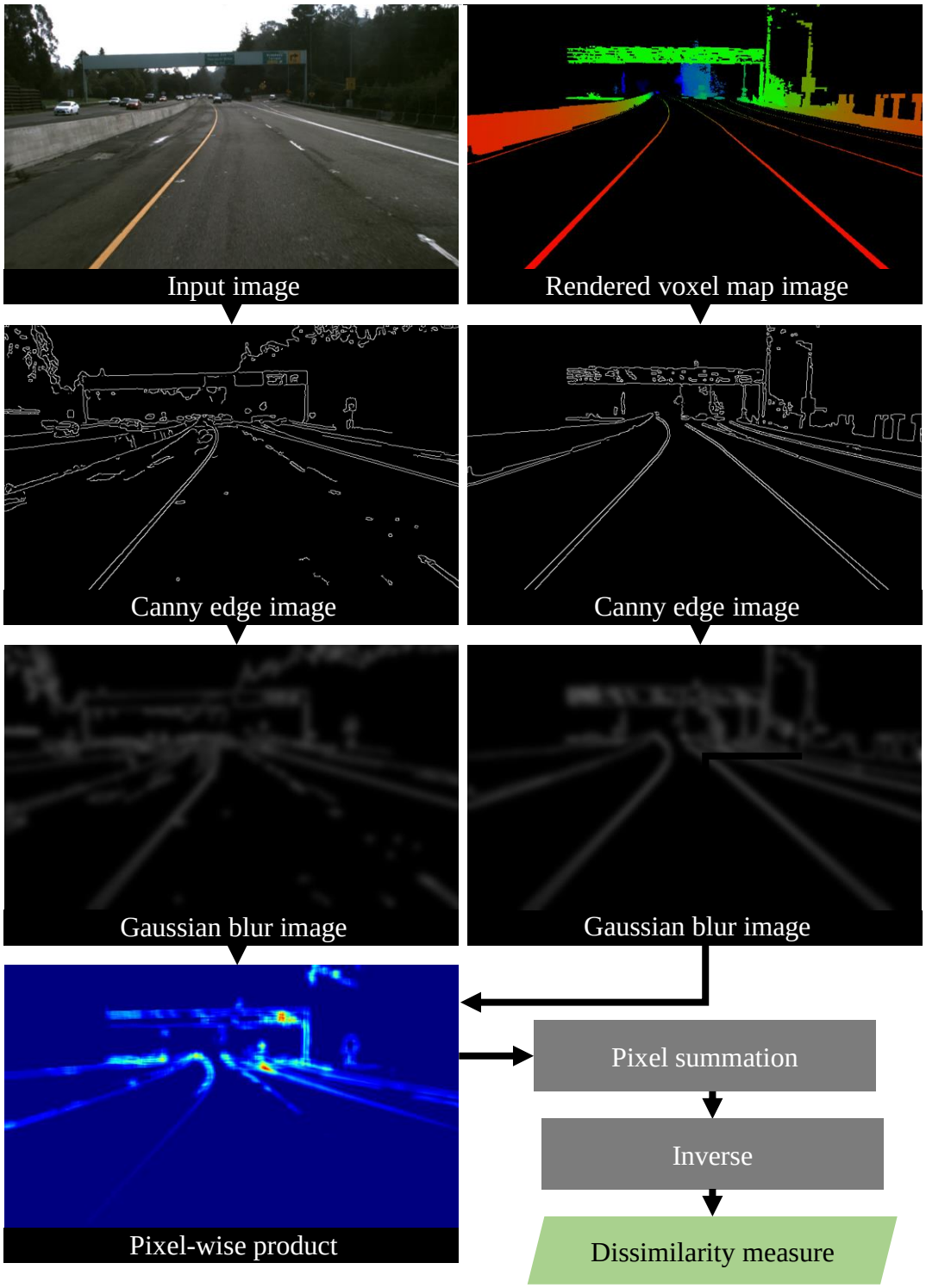


Figure 5.3: Process for determining image dissimilarity between query images and voxel map renders, which is used by the mutual edge area objective function of the proposed method.

The Canny [105] operator is used to generate edge images for both the rendered and real camera images. The resulting objective function is as follows:

$$G(\mathbf{r}) = \left(\sum_{\mathbf{x} \in \bar{I}, I(\mathbf{r})} Q(\bar{I}) \circ Q(I(\mathbf{r})) \right)^{-1}, \tag{5.2}$$

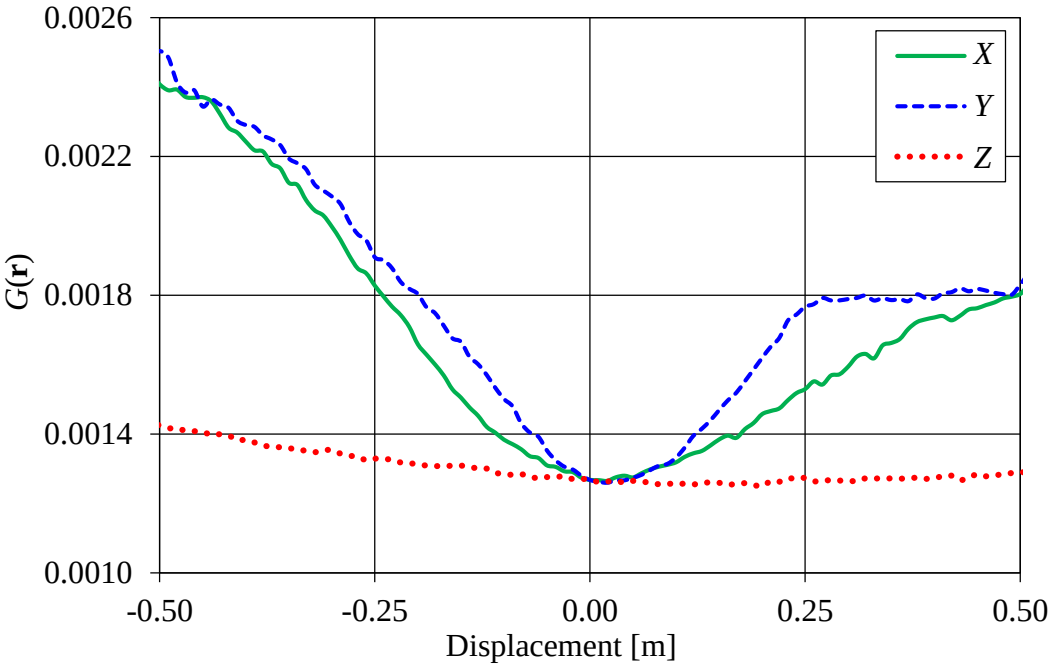
where $\mathbf{x} = [u, v]^{\mathrm{T}}$ denotes pixel location, and $Q(\bullet)$ is the Canny operator followed by a Gaussian blur over the entire image. Note that the inverse of the sum rather than the inverse of individual elements is used to prevent division by zero. Strictly speaking, $G(\mathbf{r})$ could be maximized instead of minimizing $G(\mathbf{r})^{-1}$, but for implementation using optimization libraries, and for comparison with other minimization cost functions, the inverse function is employed. The resulting objective function produces a clear minimum across the pose parameters, as shown in Figure 5.4.

5.3.2.3 Optimization framework

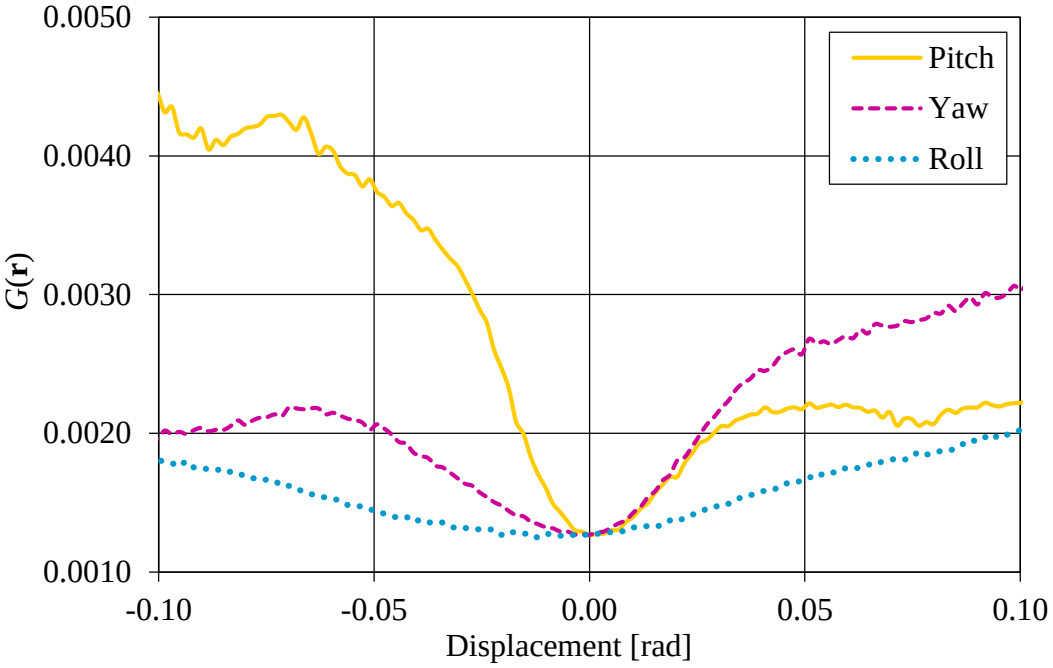
The search space for the optimal parametrization that minimizes Equation (5.2) covers 6DOF, making it a problem that requires a solving with an optimization method. While the proposed objective function could be used within any optimizer, this research uses the Ceres solver [106] implementation of the Levenberg-Marquardt [37] non-linear solver. Optimization methods usually employ either gradient descent or Gauss-Newton to determine the parameters for the next iteration, which require the calculation of the Jacobian matrix. While the Ceres solver provides numeric differentiation, this is error prone and computationally intensive, so analytic derivation of the Jacobian matrix is desirable. The derivative of Equation (5.2) requires differentiation of the dissimilarity measure function, which can be solved using the product rule as follows:

$$\frac{\partial (Q(\bar{I}) Q(I(\mathbf{r})))}{\partial \mathbf{r}} = Q(\bar{I}) \frac{\partial Q(I(\mathbf{r}))}{\partial \mathbf{r}} + Q(I(\mathbf{r})) \frac{\partial Q(\bar{I})}{\partial \mathbf{r}}. \tag{5.3}$$

Here it is noted that the right-hand side of the sum in Equation (5.3) can be removed, since the virtual camera parameters have no affect on $Q(\bar{I})$. Proceeding with the



(a) Translational displacements.



(b) Rotational displacements.

Figure 5.4: Sample objective-function response as the virtual camera is rendered at different (a) translational displacements, and (b) rotational displacements from the true query camera position.

chain rule, Equation (5.3) can be rewritten as:

$$\frac{\partial(Q(\bar{I})Q(I(\mathbf{r})))}{\partial \mathbf{r}} = Q(\bar{I})\frac{\partial Q(I(\mathbf{r}))}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{r}}. \tag{5.4}$$

On the right side of Equation (5.4), $\frac{\partial Q(I(\mathbf{r}))}{\partial \mathbf{x}}$ is the gradient image of $Q(I(\mathbf{r}))$, which can be approximated using a convolution filter. The partial derivative $\frac{\partial \mathbf{x}}{\partial \mathbf{r}}$ is the image-plane velocity of a pixel relative to the velocity of camera motion. This is commonly used in visual servoing applications for robotics, and is known as the *image Jacobian*. The image Jacobian is presented here without derivation, which can be found in relevant literature [107]:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{r}} = \begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u}{Z} & \frac{uv}{\lambda} & \frac{-\lambda^2-u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{Z} & \frac{v}{Z} & \frac{\lambda^2+v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix}, \tag{5.5}$$

where λ is the focal length of the camera in pixels. Note that Equation (5.5) contains Z, which is supplied directly for each pixel from the rendering of $I(\mathbf{r})$ as a depth image. In the rendering process, the depth is provided within the vertex shader and passed to the fragment shader. An alternative would be to use the Z-buffer which is calculated for the purpose of rendering. This would require the replacement of λ in Equation (5.5) with 1.0 since the Z-buffer is in normalized virtual camera space. In either case, there is a problem using the depth image to retrieve Z values for each pixel, as a Gaussian convolution is performed on the edge images. Therefore, the non-negative pixel values of $Q(I(\mathbf{r}))$ bleed past the areas that there are Z values for, resulting in an offset being introduced to the objective-function derivatives. This is overcome by recognizing that convolved pixel areas of $Q(I(\mathbf{r}))$ are at the same depth as the edge itself. By dilating the depth image $I(\mathbf{r})$ using the same kernel size as the Gaussian kernel used by Q, and using the dilated version to determine each pixel's Z value, the offset is nearly completely removed from the objective-function derivatives.

It is now possible to determine the Jacobian matrix of the objective function, $\mathbf{J}$ by summing over all pixels and applying the reciprocal rule to the complete equation,

giving:

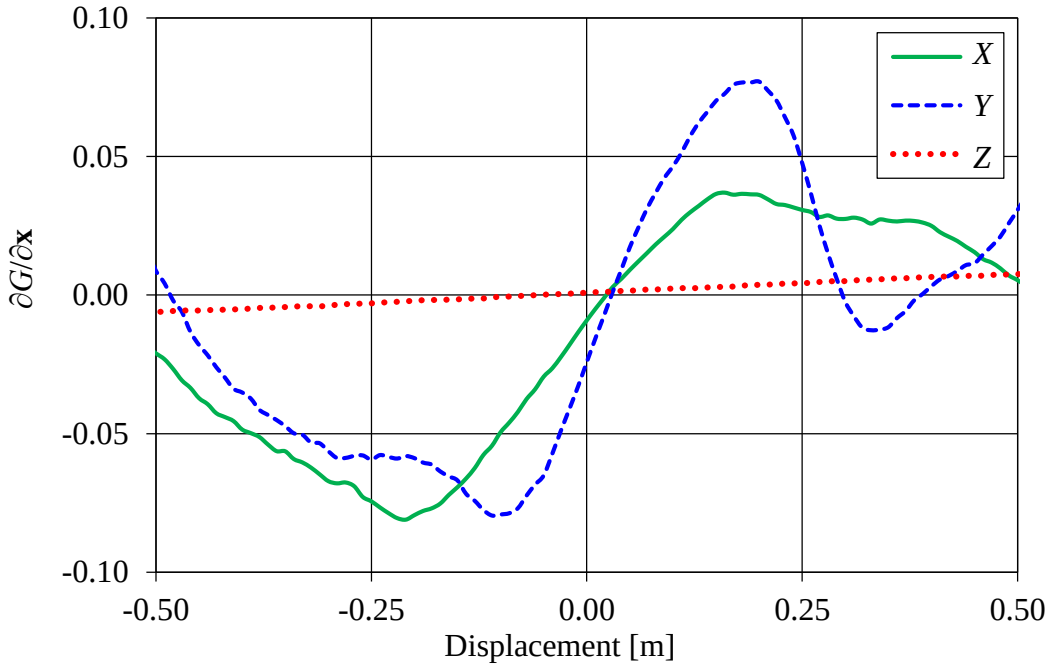
$$\mathbf{J} = \left(\sum_{\mathbf{x} \in \bar{I}, I(\mathbf{r})} Q(\bar{I}) \frac{\partial Q(I(\mathbf{r}))}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{r}} \right) G(\mathbf{r})^2. \tag{5.6}$$

This is in a form that can be directly used by a gradient-based optimizer. Approximations of the gradient images $\frac{\partial Q(I(\mathbf{r}))}{\partial \mathbf{x}}$ are calculated using the Scharr operator [108].

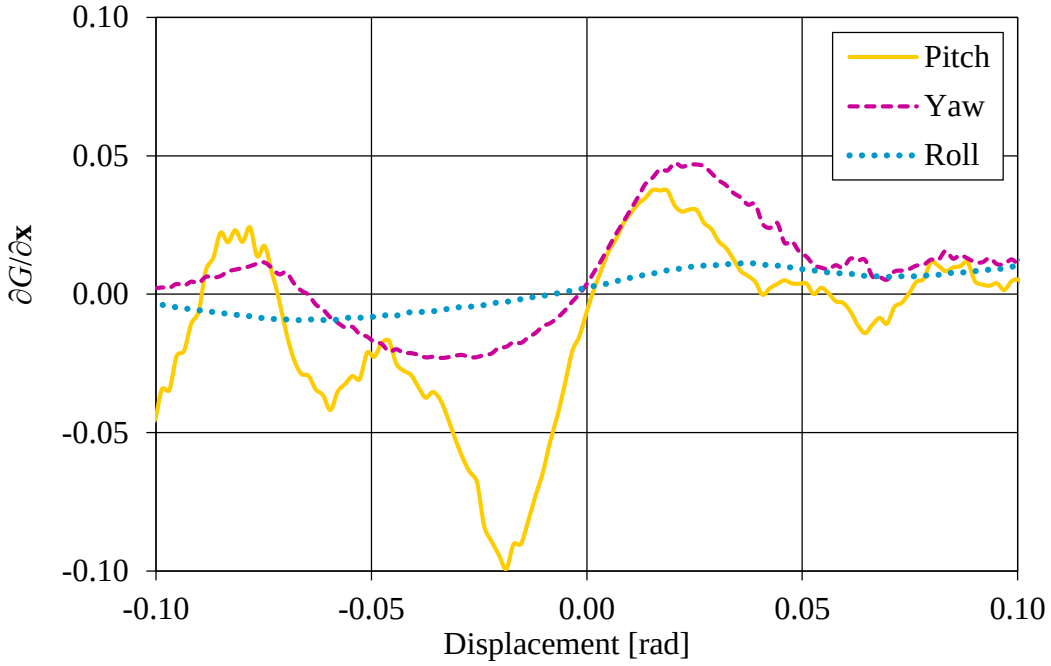
Figure 5.5 presents the objective-function derivatives for a sample image. The graphs show that all of the derived partial derivatives clearly pass close to zero around the ground-truth pose, providing a stable optimization pathway. One point to note is that the Z direction partial derivative and cost function are the least sharply defined. This is the direction of the camera's optical axis, and when we consider typical highway scenes, the reason for this becomes apparent. Most of the scene structure from the voxel map is distant, so transformation in the Z direction does not significantly alter the rendered scene. Closer areas, such as lane markers and road boundaries, are usually in parallel with the direction of motion. This would be less of an issue for city areas, where there are usually more structures close to the camera. In the highway environment of the HERE dataset, it causes a low longitudinal localization accuracy which must be compensated for by using odometry data to supply translation distance.

5.3.2.4 Kalman filtering

The position estimates of the optimization process are used as inputs to a Bayesian localization framework. For the first input image, position is initialized using the GPS data from an inaccurate (smart-phone grade) receiver. This could instead come from a topological or topometric localization as proposed in Chapter 3 or Chapter 4. The camera yaw, pitch, and roll are initialized by finding the road-lane direction at the estimated camera position, based on the road-lane marker data in the map. All six camera parameters are incorporated into a Kalman filter. With each new input image, the prediction stage of the filter estimates the next values of the camera parameters, which are then used as initial values for the visual optimization described above in



(a) Translational displacements.



(b) Rotational displacements.

Figure 5.5: Sample objective-function Jacobian response as the virtual camera is rendered at different (a) translational displacements, and (b) rotational displacements from the true query camera position.

Section 5.3.2.3. The covariances estimated by the Kalman filter are used to construct a bounding region two standard deviations wide for parameter optimization. The output of the optimization step provides the measurement update stage of the filter, with covariance estimates supplied by the Ceres solver. The filter then returns state estimates for all camera parameters of the input image.

The shallow cost function basin of the objective function of Equation (5.2) in the Z direction provides a challenge for the optimizer, causing substantial drift in the position estimate along the direction of travel. An estimate of the scalar vehicle speed can be included into the measurement update to help to overcome this. The speed estimate does not have to be highly accurate; a consumer grade GPS, or odometry hardware is sufficient to reduce drift. While the measurement noise of GPS speed estimates can generally be approximated by a normal distribution, the noise distribution of the sensor used would have to be considered for use with the Kalman filter.

5.4 Experiments

The HERE dataset was used to evaluate the performance of the proposed method. The dataset includes query images with ground-truth capture locations, and the voxel map that was briefly introduced in Section 5.3.1. The ground-truth data provided by the HERE dataset only includes 3D capture locations, so optimization of Euler angles in the pose estimates could not be evaluated except by visual inspection of each frame. However, the location of the camera is of most interest, and angular pose estimation is only necessary for the optimization process.

This section starts with Section 5.4.1 providing a summary of the methods that were evaluated in these experiments. The process and results of the direct metric localization experiments are presented in Section 5.4.2.

5.4.1 Evaluated methods

The unique design of the database as a sparse voxel map for localization means that comparative direct metric localization methods are not really available. Direct metric localization methods usually use a much more detailed database, allowing the use of image entropies [35] to form a cost function. These cost functions fail completely when using the sparse voxel map presented in this research.

As was discussed in Section 5.3.2.3 and Section 5.3.2.4, the optimization process has the highest uncertainty in the Z direction, caused by a shallow optimization basin of the proposed cost function in the direction of motion. Some kind of speed sensor can potentially help to overcome this issue. To determine the effects of adding a speed sensor input to the Kalman filter in the localization process, a variety of speed estimate sources were tested. The evaluated methods can be summarized as follows:

1. **Proposed method 1: Ground-truth speed measurement.** This method calculates a scalar speed measurement from the ground-truth data provided with the query images. Some normally distributed noise is added, to simulate the kind of data that would be obtained from a normal vehicle speedometer. This method is included to demonstrate the potential level of localization accuracy possible when using an accurate speed estimate, for example from wheel sensors. The speed estimate is included in the Kalman filter's measurement update, as described in Section 5.3.2.4.

2. **Proposed method 2: GPS speed measurement.** This method calculates a scalar speed measurement from the noisy (consumer smart-phone grade) GPS that is provided with the query images. Note that the GPS data is different from the ground-truth data; this is explained in more detail below in Section 5.4.2.1. The speed measurement is calculated by averaging inaccurate GPS point motion. If a GPS satellite speed reading is available, this would be a much more suitable input, as GPS speed accuracy tends to be an order of magnitude superior to positional readings, even for inexpensive receivers [43]. The speed

estimate is included in the Kalman filter's measurement update, as described in Section 5.3.2.4.

3. **Proposed method 3: No speed measurement.** This method only uses the measurement update provided by the visual localization process, without any speed measurement from other sensors included.

4. **Comparative method 1: Filtered GPS only.** This method uses the measurements of the consumer-grade, 1 Hz GPS only to produce a localization estimate. The raw GPS values are very inaccurate, but the localization performance was improved by applying Kalman filtering and interpolation between data points.

5.4.2 HERE dataset experiment

The HERE dataset [103] was used to evaluate the effectiveness of the proposed method. It is composed of a voxel map database and a series of camera images for use as query images. The dataset was captured over 10 km on a multi-lane highway in California, USA. Because it was intended to be used for a competition at the Intelligent Transport Systems World Congress 2016 (ITSWC2016) [104], there is not much information about how the data was captured, or what hardware was used. Unlike the previous two chapters, the database construction does not use visual images, so in this chapter the experimental setup is separated into two sections; the first, detailing the query image frames (Section 5.4.2.1), and the second, describing the voxel map (Section 5.4.2.2).

5.4.2.1 Image capture

The sequence of query images supplied by the HERE dataset were captured from a single forward-facing camera, mounted to the capture vehicle. While the calibration parameters are available, the specification of the camera itself is not supplied. The

camera frame rate was 10 fps, and the images have a resolution of $1{,}600 \times 1{,}200$ pixels. However, the supplied intrinsic calibration parameters are for images scaled to 800×600 pixels, so the images were resized appropriately for use in these experiments. Distortion was removed using the supplied calibration parameters, and the images were cropped to remove the vehicle bonnet area that can be seen in a sample image from the dataset in Figure 5.1.

The images are also supplied with a stream of timestamped GPS coordinates. While the GPS device used is unknown, it is stated to be of consumer smart-phone grade quality. The GPS frequency was only 1 Hz, and the accuracy is very low —with errors exceeding 30 m in places. In the proposed methods of this research, the GPS position measurements were not used, but the speed measurement calculated from the distances between GPS points and their timestamps was used in proposed method 2. The GPS frequency was much lower than the camera frame rate, so the GPS speed measurements were interpolated to provide a speed measurement for each query image frame. The interpolated GPS position data was also used for the comparative method, after smoothing with a Kalman filter.

Ground-truth data is also supplied with the query image frames. While the method used to construct this is unknown, it is presented as a set of GPS coordinates and altitude in meters for each query image frame. The capture positions were most likely determined using a sensor suite similar to the MMS system that was presented in Section 3.4.4.1 (in Chapter 3). A summary of the capture specifications is presented in Table 5.1.

5.4.2.2 Voxel-map construction

The map supplied by HERE covers the entire stretch of road where query images were captured. It consists of a set of cuboid voxel GPS coordinates, which define areas of the map close to the road which contain part of a solid object. The size of each voxel is approximately 25 cm square. While the road surface itself is not included in the occupancy grid, the location of painted road lines are provided as a

Table 5.1: Capture specifications for the HERE dataset experiment.

Property	**Specification**
Localization accuracy [m]	Unknown
Imaging sensor	Unknown
Lens field of view	Unknown
Image resolution [pixels]	800×600 (calibrated, scaled)
Image capture rate [fps]	10
Capture route length [m]	~10,000
Auxiliary GPS frequency [Hz]	1.0

string of GPS coordinates. There is also information about road-sign locations and the text that they display, but this part of the dataset was not used in this research. The database is supplied in segments of JSON files, each of which contain the voxel and road-marker coordinates for a section of road approximately 13 m long.

5.4.2.3 Localization performance

Localization was performed using the supplied query images and database files. The same machine used for the experiments in the previous chapters, Chapter 3 and Chapter 4, was used to perform the localization process.

For the first input image, position was initialized using the GPS receiver data supplied with the images. The camera yaw, pitch, and roll were initialized by finding the road-lane direction at the estimated camera position, based on the road-lane marker data in the map. The rendering of the voxel map was performed using OpenGL [109].

The localization results are shown in Figure 5.6 and Table 5.2. While the results show the importance of the speed estimate to control the drift in the direction of

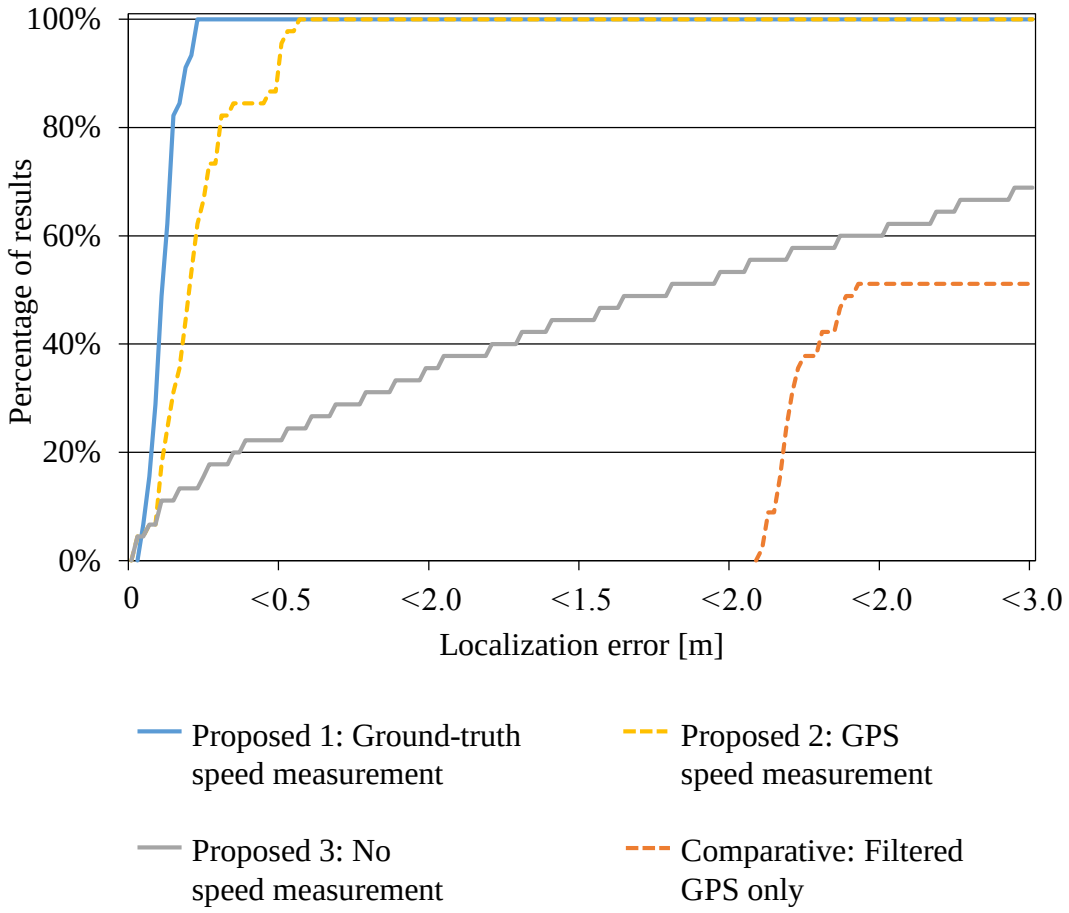


Figure 5.6: Direct metric localization results from the HERE dataset. The graph shows the percentage of localization results within each metric error level.

motion. It can be seen that even a speed measurement based on measurements from a noisy GPS receiver allows significant reduction of the longitudinal positioning drift, allowing decimeter-level positioning accuracy.

5.5 Discussion and analysis

This section provides a discussion about the proposed method. The effectiveness of the proposed method is analyzed and the performance with and without speed measurement input is discussed. A discussion on the outstanding issues of the proposed system and considerations for actual implementation is also presented.

Table 5.2: Summary of the direct metric localization results from the HERE dataset. The best results are shown in bold.

Method	Average error [m]	Standard deviation [m]	Maximum error [m]
Proposed 1: Ground-truth speed measurement	**0.11**	**0.05**	**0.21**
Proposed 2: GPS speed measurement	0.22	0.14	0.55
Proposed 3: No speed measurement	2.08	1.65	5.17
Comparative: Filtered GPS only	5.13	3.05	9.05

5.5.1 Evaluated methods

In these experiments, the proposed method was tested with a variety of supplementary speed measurements to control longitudinal drift. The proposed system could resolve the viewing direction of the camera and lateral offset within the road-lane very well, but struggled with fine-grained longitudinal positioning in areas of the map where image edges were mostly parallel to the direction of motion. This is because most of the scene structure that contributes to the edge images, and therefore the objective function, is fairly distant from the camera. The edges of closer scene structure, which will provide the most information for the Z direction, is nearly all parallel to the direction of motion in highway environments.

As can be seen in the example images from Figure 5.1, the highway scene means that the road markers and barriers close to the edges of the road are very self-similar along the direction of motion. Small longitudinal changes therefore have a small affect on the dissimilarity measure used in optimization. This issue would potentially be much less of a problem in urban environments, where strong edges from buildings and road markings are more often perpendicular to the direction of the road. There is also typically more scene structure closer to the camera when compared to the challenging highway environment.

The ambiguity introduced in the Z direction causes the accumulation of longitudinal drift, as can be seen in the results from proposed method 3 that uses no correction from a speed measurement. Over the relatively short stretch of road used in this experiment, the average error (Table 5.2) was still reasonable for proposed method 3. However, as can be seen from Figure 5.6, the drift will likely increase beyond the point where the recovery becomes impossible, even if some perpendicular scene structure is found.

The results from proposed methods 1 and 2 show that inclusion of a speed estimate, even if noisy as is the case in proposed method 2, are able to control the longitudinal drift. It can be seen that although higher localization accuracy was achieved when using a more precise vehicle speed estimate, even a rough estimate generated from

noisy GPS receiver readings was sufficient to provide decimeter-level localization accuracy.

Although not used in this experiment, the filtered GPS position measurements from comparative method 1 may also be able to be incorporated to reduce drift; however, because of the large biases observed (more than 9 m in places), careful application would be necessary. The raw GPS data sometimes has localization errors exceeding 30 m, so some form of checking that the GPS paths are in possible road areas would also be beneficial if GPS position (rather than speed) measurements were used.

Proposed method 1 uses the ground-truth information to calculate a scalar speed value. Although some noise is added to simulate a real sensor output (i.e, from wheel sensors), it is not an actual sensor reading so is shown here only for evaluation purposes. It was used to test the level of localization accuracy that could be achieved if a more accurate speed measurement than the GPS speed measurement of proposed method 2 were available.

5.5.2 Optimizer considerations

While optimization performance was surprisingly good given the limited information provided by the voxel map renders, there are a few remaining issues with the method. Initialization and parameter filtering before optimization are very important, as local minima are frequent. While convergence into local minima can be reduced by maintaining filtered parameter estimates, as was described in Section 5.3.2.4, the longitudinal drift discussed above in Section 5.5.1 still poses a problem, not just for the localization accuracy, but also for the optimization process. This is because even a relatively small amount of drift increases the chances of the optimizer falling into local minima.

Local minima are a large problem for all direct metric localization methods, as they can cause catastrophic failure of the localization system. Given the sparse nature of the rendered images that this research proposes, local minima may potentially be

more likely than with systems using more information. In the section of road evaluated in this experiment, repeated convergence into local minima did not occur in such a way that localization failed entirely, even when no speed measurement was used. However, to ensure that this does not occur over long localization sequences, combining with other sensors such as GPS may be the simplest solution to this problem. In addition, combining optimization over multiple frames [110] may reduce the risk of falling into local minima and may also slightly improve localization accuracy.

5.5.3 Localization run-time

One important aspect of any localization method is the run-time. The current implementation as presented in this chapter does not run in real-time, with the optimization for each input image taking in the order of 10 sec./frame on the test machine used. This is many times slower than the real-time topological and topometric localization methods described in Chapter 3 and Chapter 4. In these experiments, the machine used for testing had only on-board graphics and the implementation included no particular multi-threading nor optimization. Based on the performance of more complex direct metric localization methods, real-time localization should be achievable with optimization and more powerful dedicated graphics hardware. The main performance bottleneck is in the Jacobian calculations, taking on average 80% of the optimization time. The Jacobian calculations were completely performed on the CPU. The other major computational bottleneck is the voxel-map image rendering process, which again would be significantly faster if a dedicated GPU were employed.

Localization methods that require a larger database may be limited in their operation speed by the bottleneck of constantly streaming and/or loading a large amount of information. The sparse voxel map used by the method described in this chapter is very compact when compared to other direct metric localization methods. The voxel data, stored in JSON format, occupies less than 180 KB/m. This is still larger than the databases created by the topological and topometric localization methods

described in Chapter 3 and Chapter 4. Streaming in real-time may be difficult using modern mobile networks, but further compacting of the dataset should be possible by removing voxel information from voxels that are completely surrounded by other voxels; these voxels make up a large part of the voxel map and do not contribute to the rendering result.

5.6 Summary

In this thesis, three vehicle localization types using monocular cameras are explored. As introduced in Chapter 1, these are topological localization, topometric localization, and direct metric localization. These localization types provide progressively refined ego-localization, with topological localization giving the current general vehicle position relative to a map, topometric localization positioning the vehicle metrically in a coordinate frame, and direct metric localization performing a full 6DOF position and pose estimate. In this chapter, a direct metric localization method for precisely determining the current vehicle position and pose within a sparse voxel map was presented. This solves the problem of ego-localization by using direct visual dissimilarity between rendered voxel-map scenes and query camera image frames.

The methodology, experiments, and analysis provided in this chapter presented an effective system to perform direct metric localization of a vehicle using a single camera and a pre-constructed database. The proposed database uses combined LIDAR scans to create a sparse voxel map, which can be rendered virtually as a depth map at any point. The objective function for pose and position optimization of the virtual camera uses a mutual edge information dissimilarity measure to bridge the change in modality between the rendered depth maps scenes and the visual query image. The resulting position and pose measurements can be used within a Kalman filter to maintain a vehicle state estimate. The use of a speed estimate, either from vehicle wheel sensors or a consumer GPS unit, is required to control drift in the longitudinal direction.

The experimental results show that even a compact voxel map, that contains only edge information, can be used for precise direct metric localization. For the highway environment used in the evaluation experiment of this chapter, most close-by structure such as road lines and barriers had edges parallel to the direction of motion. While this caused ambiguity and potential drift of the position estimate in the direction of motion, the application of a scalar speed measurement was sufficient to overcome this problem. Even an inaccurate, consumer-grade GPS unit could be used to calculate a speed measurement sufficient to achieve an average localization error to 0.22 m. A more accurate speed estimate created from the ground-truth data to simulate a wheel sensor gave a further 50% reduction in average localization error to 0.11 m.

Chapter 6

Conclusion

This chapter concludes the thesis. In Section 6.1, the research presented within this thesis is summarized, and Section 6.2 discusses the challenges that still must be addressed for more robust and accurate visual localization. Section 6.3 presents potential future directions of research to extend the proposed methods. Finally, Section 6.4 completes the thesis with some closing remarks.

6.1 Summary

The research described in this thesis aims to achieve vehicle ego-localization using a single camera. Automated driving technologies require a localization system with a very challenging set of requirements. For production vehicles, sensors must be inexpensive and reliable. Meanwhile, a database or map must be compact enough to either be stored by the system or streamed to the vehicle over mobile networks. While there are many sensor technologies available, a single camera is an inexpensive option that can easily be installed in production vehicles, offering many advantages for production systems. The level of localization required depends on how the positioning information is used, and therefore in this thesis, three types of localization were

addressed. These were topological localization, topometric localization, and direct metric localization.

The first research topic, presented in Chapter 3, uses an image matching method to perform topological localization. In topological localization, the vehicle position is estimated as a discrete area within a map. The proposed method matches a sequence of query images from an in-vehicle camera to a visual database. The structure of the database enables this topological localization method to overcome a number of problems often faced by visual localization techniques. Rather than using whole image similarity, feature points matched between the database and query image sequences allow robustness against scene changes from occlusion and temporal objects. A key part of this method is the arrangement of database feature points into feature-scale tracklets. This allows image matching to be performed by each feature matched between a query image frame and the feature-scale tracklets to vote on the closest database image. Per-feature voting means that geometric constraints between matched features do not need to be used to determine the essential matrix. This, in turn, relieves the need for many, well-distributed features for well-conditioned matching. The result is a real-time topological localization method that solves a number of the challenges for visual localization. The system is robust to occlusion, operates in real-time, and can perform localization with a low number of matched features. This allows general topological localization within a map. It is suitable for intelligent vehicle systems which require a geographic ego-localization estimate, for example, environment-adaptive classifiers for object detection.

The second research topic, presented in Chapter 4, expands on the use of feature-scale tracklets to perform topometric localization. In topometric localization, the vehicle position is estimated as a metric point within the continuous 2D-coordinate frame of the map. The proposed method performs further pre-processing of the captured database to determine the regression coefficients that describe the linear relationship between feature-point scale and capture location along the road route. Localization can then be performed by using the scale of features matched between query and database image frames to provide individual position measurements. These

are combined using a Kalman filter. This allows the method to maintain the advantages of the topological method presented in Chapter 3, but determine the position of the query image within a continuous coordinate frame, therefore performing topometric localization. It is suitable for intelligent vehicle systems that require a metric position estimate, for example, path planning algorithms that require the distance to a corner or pedestrian crossing.

The third research topic, presented in Chapter 5, uses a sparse voxel map to perform direct metric localization. In direct metric localization, the vehicle position and pose are estimated to a high level of precision by comparing a query image with rendered database images. In general, a 3D database for direct metric localization must contain a large amount of information to allow visual rendering of the map at any location. Instead, the proposed method uses a compact 3D sparse voxel map, constructed using LIDAR scans. Direct metric localization can then be performed by comparing the edge images of rendered depth maps of the voxel map and each query image frame. This method is robust to occlusions as it uses a per-pixel product within the objective function, and optimization is performed within a non-linear Levenberg-Marquardt solver. It requires a vehicle speed estimate to overcome optimization ambiguity in the longitudinal direction. Although it does not currently operate in real-time, it provides decimeter-level ego-localization and also the pose of the in-vehicle camera. The estimated pose of the camera can be used to determine the vehicle orientation, suitable for intelligent vehicle systems that require a precise position estimate in 6DOF, for example, accurate lane localization and path planning around fixed road obstacles.

The three research topics summarized above provide a contribution to the challenging field of vehicle ego-localization using monocular cameras. They form a survey of three localization levels, from topological map-relative localization, through topometric position localization, to full 6DOF decimeter-level direct metric localization. These three methods have the potential to form a useful basis for visual localization systems for a wide variety of intelligent vehicle and advanced driver assistance systems.

6.2 Remaining challenges

While this thesis describes three effective visual localization methods, the problem of visual vehicle localization is far from solved. The topological and topometric localization methods described in Chapter 3 and Chapter 4 both rely on feature matching. Even the most modern feature extraction and description techniques have weaknesses which makes their use in all environments difficult. Drastic changes in lighting, in turn, change the visual appearance of the scene enough to cause a difference in extracted feature points and their descriptors. Therefore localization at night, or in severe weather conditions, is very difficult to perform with a database constructed using only day-time images. An obvious alternative is to use a dynamic database [111] which contains adaptive features, or feature descriptors which change depending on the lighting and weather conditions. This poses a data collection issue, which may only be solved with crowd-sourced imagery as localization is performed on-line.

Direct metric localization using the method described in Chapter 5 is less affected by lighting and weather changes, as the sparse voxel map contains no light-dependent information. However, it is still not a real-time solution and depends on extra sensors to provide a speed estimate in order to overcome localization ambiguity in the longitudinal direction.

There are conditions which will render images captured from an in-vehicle camera to be unusable, causing any potential visual localization method to fail. These include drastic occlusion, extreme weather, harsh light reflections, and whiteout of the camera sensor from direct light. Therefore, mission-critical localization will always have to rely on a suite of sensors rather than just one localization method. Even if a camera is used as the primary sensor for localization, at a bare minimum dead-reckoning using an IMU and wheel odometry would be required for situations where camera images are unusable. In reality, a robust localization method will use all the sensor information available to it at all times. What still remains an interesting research question, is what combination of inexpensive sensors is sufficient to perform reliable

localization in all situations? Considering this question, localization research using vision only is still very relevant. If a monocular camera localization system can be made very reliable, some auxiliary sensors (such as GPS, IMU, odometry, and perhaps RADAR or single-line LIDAR) may be required at some times. By being able to avoid a full scanning LIDAR, autonomous driving technologies that rely on localization will advance much more quickly and be less expensive for consumers to uptake. This is significant for the future of autonomous systems in intelligent vehicle technology. However, considering that human drivers use nearly entirely visual cues for localization and navigation, the goal of pure vision-only localization should not be abandoned. While commercialization of localization technologies for use in consumer autonomous driving systems is necessary, the original aim of this thesis remains unchanged. To contribute to the realization of a reliable localization and mapping system using computer vision is important, because this kind of research helps us to understand more about computer vision and human perception. It helps us to imagine what is possible beyond what can be implemented with modern sensor technologies.

6.3 Future research

There is potential for research in two main areas that would improve the localization methods presented in this thesis. The first relates to database construction for the feature-based methods for topological and topometric localization. The second considers the combination of topometric and direct metric localization techniques.

As was mentioned in the previous section, the main drawback of visual databases is the fact that the scenes change drastically under different lighting and weather conditions. The databases of the topological and topometric localization methods, presented in Chapter 3 and Chapter 4 respectively, store extracted feature points. A potential method to overcome the issues of lighting and weather changes is to use images captured by vehicles being localized to update the database dynamically. This

would create multiple feature-scale tracklet sets that vary depending on the lighting and weather conditions, and would also provide a method for keeping the database up to date with changes in road layout and so on. This concept also provides a method to retain only database features which are frequently matched to query image features.

Another potential change to the system would be to move from relatively low-level feature points to higher-level object tracklets. As object detection and classification algorithms improve, largely thanks to the advances in deep learning, semantic scene segmentation and object extraction are now potentially usable for localization. The properties and locations of objects viewed from each capture location could be stored in the database and used in a similar way to the feature-scale tracklets presented in this research.

In the research on direct metric localization using sparse voxel maps, presented in Chapter 5, there are two main drawbacks. The first is the computation time. Future research would definitely include ways to optimize the system to provide real-time performance. Standard methods for improving run speed involve implementation changes, such as multi-threading and the use of GPUs for rendering and threaded computation. In addition, the basic computational requirements could be reduced by altering the 3D voxel map to instead be a 3D edge map. This would reduce the size of the database, as only a set of points describing edges would be required rather than sets of evenly spaced voxel coordinates. It would also be faster to render and result in an edge image that could be directly used in the objective function without using an edge detection convolution filter for every rendered frame. This change, however, would not overcome the problem of localization inaccuracy and drift in the direction of motion. While this was somewhat mitigated by the use of a vehicle speed measurement, another interesting observation is that longitudinal accuracy is precisely where the topometric localization method of Chapter 4 excels. However, it is not able to perform the lateral positioning and camera-pose calculations that direct metric localization provide. Therefore, by embedding feature-scale tracklets into the 3D voxel (or edge) map, and performing topometric and direct metric localization in unison, a precise and purely visual combination could potentially be achieved.

6.4 Closing

The goal of achieving fully autonomous driving has been a strong force behind the large body of research on vehicle ego-localization. This has resulted in a variety of systems that use many kinds of sensors. As there is still no clear consensus on which combination of sensors is the most effective for reliable localization in all environments, the use of cameras as the primary source of localization measurements is an important topic. While this thesis presents some novel localization methods that use a single camera, there are still many areas for investigation and research in order to understand more about what camera-based vehicle localization is capable of.

In Chapter 1, the stated aim of this thesis was to contribute to the realization of a reliable and accurate mapping and ego-localization system that can be cheaply deployed on production vehicles, as a central component of autonomous driving systems. The research presented has demonstrated some reliable systems in the fields of topological, topometric, and direct metric localization. Some issues common in visual localization have been tackled, including database sizes for visual map construction, the use of feature-point scale comparisons to prevent the reliance on essential matrix calculation, and methods which are robust to partial scene changes and occlusion. However, there are still shortcomings in visual localization that remain to be solved for a system that can be used in production vehicles. These include the reliability of the localization system under drastic lighting and environment changes, continued performance under complete occlusion, and real-time performance of direct metric localization methods.

In the future, cameras will nearly certainly form a major part of autonomous driving systems, and very likely a central part of the localization and navigation methods employed to achieve this. As technology develops, autonomous driving may be able to operate with fewer and fewer sensors, for affordable systems that are obtainable for everyday use. As such, the importance of research in the field of computer vision for vehicle localization remains of central importance to the aim of this thesis.

Beyond autonomous driving systems, the methods presented in this thesis could also be a basis for other areas of research. The use of feature-point scale for comparing capture positions that was introduced in Chapter 3 and Chapter 4 is a concept that could be adapted for use in general place recognition, robotic ego-localization, and potentially many other tasks. Direct metric localization with a mutual edge objective function, introduced in Chapter 5, has other applications in on-line LIDAR to camera calibration and object pose estimation in robotics.

In the future of automation, image processing and scene understanding will without doubt have very important roles. As technology advances, processing hardware becomes more powerful and less expensive, opening up opportunities for computer vision to be applied to more and more everyday applications. It is very likely that visual localization will one day become a technology used daily by everyone, in much the same way that GPS is used now.

Appendix A

Scale-Invariant Image Features

Scale invariance is an important property for feature extraction techniques. The motivation behind scale invariance for feature extraction is to find repeatable image keypoints that can be extracted and matched between images of the same scene captured at different viewpoints. In this appendix, the Scale Invariant Feature Transform (SIFT) [86, 87] multi-scale extraction methodology is explained. This introduces the concept of feature scale and how it is used in this research.

The resolution of an object in an image depends on the distance of the camera to the object. Therefore extracting features which are invariant to resolution changes that occur as the camera moves require searching for features over all possible image scales. This involves the construction of a *scale space*. Scaling for a scale space can be achieved using Gaussian filtering. A Gaussian convolution blurs the image, effectively lowering the resolution and giving the same result as a change in scale of the image. The resulting scale space L is given by:

$$L(u, v, \sigma) = G(u, v, \sigma) * I(u, v), \tag{A.1}$$

where $I(u, v)$ is the image intensity at pixel location (u, v), and $G(u, v, \sigma)$ is a Gaussian with the scale parameter σ as follows:

$$G(u, v, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{\frac{-(u^2+v^2)}{2\sigma^2}} . \tag{A.2}$$

The operator $*$ in Equation (A.1) is the convolution operator, which more generally is as follows:

$$L(u, v, \sigma) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(u - \tau, v - t, \sigma)\, I(\tau, t)\, d\tau\, dt. \tag{A.3}$$

In the above equations, σ^2 is the variance of the Gaussian blur operation. While the Laplacian operator performed on Equation (A.1) creates a blob detector, which will give repeatable feature points, Lowe [87] proposed the use of Difference of Gaussians (DoG) to efficiently detect feature keypoint locations. The DoG process approximates the costly Laplacian operator. At a single scale, the DoG is simply a subtraction of two nearby scale space images, separated by a factor k as follows:

$$D(u, v, \sigma) = L(u, v, k\sigma) - L(u, v, \sigma). \tag{A.4}$$

The scale space is created by making a scale pyramid at discrete values of σ, and the DoG determined between each layer of the scale pyramid. Note that each doubling of σ is equivalent to down-sampling the image by a factor of two. Therefore, the scale space is divided into octaves, with the Gaussian image being down-sampled by a factor of two at the end of each octave and then used to start the next octave. Local maximum points in the DoG images are determined by testing each pixel in the DoG stack. Any pixel which is a local maximum within both its eight surrounding pixels, and also the corresponding 18 pixels in the scale layer directly above and below, is extracted for further testing.

The SIFT method performs more operations to interpolate the local maximum point positions in both pixel and scale space. Additional techniques are used to filter strong feature keypoints, but the scale representation as employed by the methods of Chapter 3 and Chapter 4 have now been introduced.

The extracted features represent local maxima in all parameters of the DoG space $D(u, v, \sigma)$. The maxima in the scale space indicates the size of the feature, and is used in the construction of the descriptor for reliable matching over different scales. To fully understand the use of feature keypoint scale in this research, consider a feature extracted with the DoG maximum at scale $\sigma = t$. If the camera moves forward and captures another image, the observation of the same feature is at a higher resolution because the feature is now closer to the camera. Considering the previous image at scale layer $\sigma = t$, the operation required to generate an image of equivalent resolution will require more blur, so the resulting DoG maximum in the new image will be at a higher scale $\sigma = t + \Delta$. Therefore, as is proposed in this thesis, the change in scale of a feature can be used for measuring the difference in capture position between two image frames.

In this research, the SIFT algorithm is used for feature extraction. Most scale-invariant feature extraction methods use a similar scale pyramid for multi-scale operation. The well-known Speeded Up Robust Features (SURF) method [88] uses box filters of different sizes instead of a Gaussian filter at different scales. This allows faster computation, and still involves the construction of a scale space, but the discrete steps in the scale space are much larger. Therefore, interpolation between scale layers is more important, and the lack of resolution in scale is noticeable for the methods proposed in Chapter 3 and Chapter 4 of this thesis.

Appendix B

Bayesian Estimation and Kalman Filtering

A Bayes estimator maximizes the *a posteriori* likelihood of the state of a dynamic system. If a variety of information is available about the system state, a Bayes estimator can combine this information, together with the confidence of each state estimate, to provide the most probable current state. For a moving vehicle, the current state can be predicted from previous states using a motion model and then updated from sensor measurements. The Bayes estimator combines the prediction and update estimates by recursively predicting the Probability Density Functions (PDFs) that describe the prediction estimate (state transition model) and the measurement estimate (measurement model). This appendix presents the formulation of a standard Bayes estimator and introduces the notation used throughout this thesis.

The state vector of a vehicle at the current time step k (with associated system time t_k) is denoted as $\mathbf{x}_k$. The state may include the position and pose parameters of the vehicle, as well as velocity and acceleration parameters if they are used in the state transition and/or measurement models of the system. Assuming a Markov model, the probability density function of the vehicle state $\mathbf{x}_k$ is conditioned on the previous state $\mathbf{x}_{k-1}$ and the measurement update $\mathbf{y}_k$ provided by sensors which measure the vehicle state. The measurement vector $\mathbf{y}_k$ may contain measurements of a single or

multiple parameters corresponding to the vehicle state $\mathbf{x}_k$, as captured at time t_k. The state transition model is denoted $P(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ and the measurement model $P(\mathbf{y}_k \mid \mathbf{x}_k)$. First, the transition model is used to predict the location of the vehicle as

$$P(\mathbf{x}_k \mid \mathbf{y}_{k-1}) = \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}) P(\mathbf{x}_{k-1} \mid \mathbf{y}_{k-1}) \mathrm{d}\mathbf{x}_{k-1}, \tag{B.1}$$

followed by the update state incorporating new measurements from the visual localization system within the measurement model as

$$P(\mathbf{x}_k \mid \mathbf{y}_k) = \alpha P(\mathbf{y}_k \mid \mathbf{x}_k) P(\mathbf{x}_k \mid \mathbf{y}_{k-1}), \tag{B.2}$$

where α is a scalar to ensure that the result integrates to one. In absence of any control input or odometry information, the transition model can be a simple motion model such as a constant velocity motion model.

In practice, calculating the maximum *a posteriori* estimate from Equation (B.1) and Equation (B.2) is not possible because the integrals are intractable. However, for some restrictive cases, a calculable optimal solution does exist. If it is appropriate to assume a unimodal system with linear transition and measurement models, and if the process noise in both transition and measurement steps are Gaussian, the Kalman filter can be used.

A Kalman filter formulates the state system evolution as follows:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{w}_k, \tag{B.3}$$

where $\mathbf{F}_k$ is the state transition matrix, and $\mathbf{w}_k$ is the process noise vector. The state transition matrix applies the effect of each system-state parameter at time $k-1$ to the current system state at time step k. The process-noise vector $\mathbf{w}_k$ gives the noise terms for each element of the state vector $\mathbf{x}_k$. It is clear here that the transition matrix $\mathbf{F}$ represents a linear transformation, and if $\mathbf{w}_k$ is drawn from a zero-mean Gaussian distribution, the Kalman filter will provide an optimal solution.

Based on Equation (B.3), the Kalman transition, or prediction stage, can be formulated as follows:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \tag{B.4}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{Q}_k, \tag{B.5}$$

where $\mathbf{P}_k$ is the covariance matrix for the parameters in $\mathbf{x}$, and $\mathbf{Q}_k$ is the covariance matrix of the process noise $\mathbf{w}_k$.

Before formulating the Kalman measurement update, the measurement model must also be considered. This maps the response of the sensor into the state vector parameters as follows:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \tag{B.6}$$

where $\mathbf{H}_k$ is the matrix that describes the transformation from sensor measurement outputs to the state vector, and $\mathbf{v}_k$ is the measurement noise vector. Again, for a Kalman filter, $\mathbf{H}_k$ describes a linear mapping and $\mathbf{v}_k$ must follow a zero-mean, Gaussian distribution. In the research contained in this thesis, $\mathbf{H}_k$ is simply the identity matrix. This is because the visual position measurements and speed measurements are direct measurements of state vector parameters.

Using Equation (B.6) to construct a measurement update based on Equation (B.2), the Kalman measurement update can be formulated as follows:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \tag{B.7}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}. \tag{B.8}$$

Here $\mathbf{K}_k$ is the Kalman gain, which is calculated at each time step as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k)^{-1}, \tag{B.9}$$

where $\mathbf{R}_k$ is the covariance of the measurement noise $\mathbf{v}_k$. Equation (B.9) shows how

the Kalman gain forms a variance ratio, which changes the influence of the measurement update on the state estimate at each time step based on the relative variance of the prediction and measurement update in each parameter of the state vector. The final state estimate, $\hat{\mathbf{x}}_{k|k}$ at time-step k is given by Equation (B.7), completing the Kalman filter formulation of a Bayes estimator.

Bibliography

[1] *The Motor Industry of Japan 2016*. Japan Automobile Manufacturers Association, Inc., May 2016.

[2] Organization for Economic Co-operation and Development (OECD), 2017. Road accidents (indicator), Accessed: 2017-11-20. URL: https://data.oecd.org/transport/road-accidents.htm.

[3] Kazumoto Morita and Michiaki Sekine. Analysis of accidents by older drivers in Japan. In *Proc. 13th Pacific Conf. on Automotive Engineering, Gyeongju, Korea*, pages 719–724, Aug. 2005.

[4] Yasushi Nishida. Analyzing accidents and developing elderly driver-targeted measures based on accident and violation records. *IATSS Research*, 39(1): 26–35, July 2015.

[5] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications*. Artech House, Norwood, MA, USA, 2nd edition, 2005.

[6] Greg Milner. *Pinpoint: How GPS is Changing Technology, Culture, and Our Minds*. W. W. Norton & Company, New York, NY, USA, 1st edition, 2016.

[7] Muharrem Keskin and Sait M. Say. Feasibility of low-cost GPS receivers for ground speed measurement. *Computers and Electronics in Agriculture*, 54(1): 36–43, Oct. 2006.

[8] Thomas Julg. Evaluation of multipath error and signal propagation in a complex scenario for GPS multipath identification. In *Proc. 4th IEEE Int. Symposium on Spread Spectrum Techniques and Applications (ISSSTA1996), Mainz, Germany*, volume 2, pages 872–876, Sept. 1996.

[9] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google Street View: Capturing the world at street level. *Computer*, 43(6):32–38, June 2010.

[10] Daichi Suzuo, Daisuke Deguchi, Ichiro Ide, Hiroshi Murase, Hiroyuki Ishida, and Yoshiko Kojima. Environment adaptive pedestrian detection using in-vehicle camera and GPS. In *Proc. 9th Int. Conf. on Computer Vision Theory and Applications (VISAPP2014), Lisbon, Portugal*, pages 354–361, Jan. 2014.

[11] Jeffrey Hawke, Corina Gurău, Chi Hay Tong, and Ingmar Posner. Wrong today, right tomorrow: Experience-based classification for robot perception. In *Proc. 10th Conf. on Field and Service Robotics (FSR2015), Toronto, ON, Canada*, pages 173–186, June 2016.

[12] Emilio Remolina and Benjamin Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104, Jan. 2004.

[13] Benjamin Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2(2): 129–153, April 1978.

[14] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1):47–63, Nov. 1991.

[15] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robotic exploration as graph construction. *IEEE Trans. on Robotics and Automation*, 7(6):859–865, Dec. 1991.

[16] Maja J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation*, 8(3):304–312, June 1992.

[17] Hernán Badino, Daniel F. Huber, and Takeo Kanade. Visual topometric localization. In *Proc. 2011 IEEE Intelligent Vehicles Symposium (IV2011), Baden-Baden, Germany*, pages 794–799, June 2011.

[18] Itaï Ben Yaacov. Topometric spaces and perturbations of metric structures. *Logic and Analysis*, 1(3):235–272, July 2008.

[19] Sebastian Thrun, Jens-Steffen Gutmann, Dieter Fox, Wolfram Burgard, and Benjamin Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proc. 15th AAAI Nat. Conf. on Artificial Intelligence (AAAI1998), Madison, WI, USA*, pages 989–995, July 1998.

[20] Benjamin Kuipers, Joseph Modayil, Patrick Beeson, Matt MacMahon, and Francesco Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proc. 2004 IEEE Int. Conf. on Robotics and Automation (ICRA2004), New Orleans, LA, USA*, volume 5, pages 4845–4851, April 2004.

[21] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, Dec. 1992.

[22] Suma Dawn, Vikas Saxena, and Bhudev Sharma. Remote sensing image registration techniques: A survey. In *Proc. 4th Int. Conf. on Image and Signal Processing, Québec City, QC, Canada*, volume 6134 of *Lecture Notes on Computer Science*, pages 103–112, June 2010.

[23] Medha V. Wyawahare, Pradeep M. Patil, and Hemant K. Abhyankar. Image registration techniques: An overview. *Int. J. Signal Processing, Image Processing and Pattern Recognition*, 2(3):11–28, Sept. 2009.

[24] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. 2000 IEEE Int. Conf. on Robotics and Automation (ICRA2000), San Francisco, CA, USA*, volume 2, pages 1023–1029, April 2000.

[25] Hiroyuki Uchiyama, Daisuke Deguchi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase. Ego-localization using streetscape image sequences from in-vehicle cameras. In *Proc. 2009 IEEE Intelligent Vehicles Symposium (IV2009), Xi'an, China*, pages 185–190, June 2009.

[26] Michael Milford. Visual route recognition with a handful of bits. In *Proc. 2012 Robotics: Science and Systems Conf., Sydney, NSW, Australia*, pages 297–304, July 2012.

[27] Hernán Badino, Daniel F. Huber, and Takeo Kanade. Real-time topometric localization. In *Proc. 2012 IEEE Int. Conf. on Robotics and Automation (ICRA2012), St. Paul, MN, USA*, pages 1635–1642, May 2012.

[28] Henning Lategahn and Christoph Stiller. Vision-only localization. *IEEE Trans. Intelligent Transportation Systems*, 15(3):1246–1257, June 2014.

[29] Xiaozhi Qu, Bahman Soheilian, and Nicolas Paparoditis. Vehicle localization using mono-camera and geo-referenced traffic signs. In *Proc. 2015 IEEE Intelligent Vehicles Symposium (IV2015), Seoul, Korea*, pages 605–610, June 2015.

[30] Hugh Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, Sept. 1981.

[31] Philip H. S. Torr, Andrew Zisserman, and Stephen J. Maybank. Robust detection of degenerate configurations while estimating the fundamental matrix. *Computer Vision and Image Understanding*, 71(3):312–333, Sept. 1998.

[32] Bill Triggs. Joint feature distributions for image correspondence. In *Proc. 2001 IEEE Int. Conf. on Computer Vision (ICCV2001), Vancouver, BC, Canada*, volume 2, pages 201–208, July 2001.

[33] Geoffrey Pascoe, Will Maddern, and Paul Newman. Robust direct visual localisation using normalised information distance. In *Proc. 26th British Machine Vision Conference (BMVC2015), Swansea, Wales, UK*, volume 3, pages 70.1–70.13, Sept. 2015.

[34] Ryan W. Wolcott and Ryan M. Eustice. Visual localization within LIDAR maps for automated urban driving. In *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2014), Chicago, IL, USA*, pages 176–183, Sept. 2014.

[35] Guillaume Caron, Amaury Dame, and Eric Marchand. Direct model based visual tracking and pose estimation using mutual information. *Image and Vision Computing*, 32(1):54–63, Jan. 2014.

[36] Daisuke Deguchi, Kensaku Mori, Marco Feuerstein, Takayuki Kitasaka, Calvin R. Maurer, Yasuhito Suenaga, Hirotsugu Takabatake, Masaki Mori, and Hiroshi Natori. Selective image similarity measure for bronchoscope tracking based on image registration. *Medical Image Analysis*, 13(4):621–633, Aug. 2009.

[37] Jorge J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In *Proc. 1977 Biennial Conf. on Numerical Analysis, Dundee, Scotland, UK*, pages 105–116, June 1978.

[38] Hugh F. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Mag.*, 13(2):99–110, June 2006.

[39] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proc. 2001 IEEE Int. Conf. on Robotics and Automation (ICRA2001), Seoul, Korea*, pages 2051–2058, May 2001.

[40] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Monocular vision based SLAM for mobile robots. In *Proc. 18th IAPR Int. Conf. on Pattern Recognition (ICPR2006), Hong Kong, China*, volume 3, pages 1027–1031, Aug. 2006.

[41] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[42] Tom Botterill, Steven Mills, and Richard D. Green. Bag-of-words-driven, single-camera simultaneous localization and mapping. *J. Field Robotics*, 28 (2):204–226, March 2011.

[43] Marko Modsching, Ronny Kramer, and Klaus ten Hagen. Field trial on GPS accuracy in a medium size city: The influence of built-up. In *Proc. 3rd Workshop on Positioning, Navigation and Communication (WPNC2006), Hannover, Germany*, pages 209–218, March 2006.

[44] Vincent Drevelle and Philippe Bonnifait. Global positioning in urban areas with 3-D maps. In *Proc. 2011 IEEE Intelligent Vehicles Symposium (IV2011), Baden-Baden, Germany*, pages 764–769, June 2011.

[45] Vincent Drevelle and Philippe Bonnifait. Reliable positioning domain computation for urban navigation. *IEEE Intelligent Transportation Systems Mag.*, 5(3):21–29, Fall 2013.

[46] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Proc. 2012 Robotics: Science and Systems Conf., Atlanta, GA, USA*, June 2007.

[47] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Proc. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA2010), Anchorage, AK, USA*, pages 4372–4378, May 2010.

[48] Naoki Akai, Luis Yoichi Morales, Eijiro Takeuchi, Yuki Yoshihara, and Yoshiki Ninomiya. Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching. In *Proc. 2017 IEEE Intelligent Vehicles Symposium (IV2017), Los Angeles, CA, USA*, pages 1356–1363, June 2017.

[49] Pragya Agrawal, Asif Iqbal, Brittney Russell, Mehrnaz Kh. Hazrati, Vinay Kashyap, and Farshad Akhbari. PCE-SLAM: A real-time simultaneous localization and mapping using LIDAR data. In *Proc. 2017 IEEE Intelligent Vehicles Symposium (IV2017), Los Angeles, CA, USA*, pages 1752–1757, June 2017.

[50] Luca Marchetti, Giorgio Grisetti, and Luca Iocchi. A comparative analysis of particle filter based localization methods. In *RoboCup 2006: Robot Soccer World Cup X, Bremen, Germany*, volume 4434 of *Lecture Notes on Computer Science*, pages 442–449, June 2006.

[51] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proc. 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2003), Las Vegas, NV, USA*, pages 2743–2748, Oct. 2003.

[52] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J. Lilienthal. Normal distributions transform Monte-Carlo localization (NDT-MCL). In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2013), Tokyo, Japan*, pages 382–389, Nov. 2013.

[53] Ryan W. Wolcott and Ryan M. Eustice. Fast LIDAR localization using multiresolution Gaussian mixture maps. In *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA2015), Seattle, WA, USA*, pages 2814–2821, May 2015.

[54] Zhuang Jie Chong, Baoxing Qin, Tirthankar Bandyopadhyay, Marcelo H. Ang, Emilio Frazzoli, and Daniela Rus. Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA2013), Karlsruhe, Germany*, pages 1554–1559, May 2013.

[55] Liang Li, Ming Yang, Lindong Guo, Chunxiang Wang, and Bing Wang. Hierarchical neighborhood based precise localization for intelligent vehicles in

urban environments. *IEEE Trans. Intelligent Vehicles*, 1(3):220–229, Sept. 2016.

[56] Malin Lundgren, Erik Stenborg, Lennart Svensson, and Lars Hammarstrand. Vehicle self-localization using off-the-shelf sensors and a detailed map. In *Proc. 2014 IEEE Intelligent Vehicles Symposium (IV2014), Detroit, MI, USA*, pages 522–528, June 2014.

[57] Malin Lundgren, Lennart Svensson, and Lars Hammarstrand. Variational Bayesian expectation maximization for RADAR map estimation. *IEEE Trans. Signal Processing*, 64(6):1391–1404, March 2016.

[58] Matthias Rapp, Markus Hahn, Markus Thom, Jürgen Dickmann, and Klaus Dietmayer. Semi-Markov process based localization using RADAR in dynamic environments. In *Proc. 18th IEEE Int. Conf. on Intelligent Transportation Systems (ITSC2015), Las Palmas de Gran Canaria, Spain*, pages 423–429, Sept. 2015.

[59] Erik Ward and John Folkesson. Vehicle localization with low cost RADAR sensors. In *Proc. 2016 IEEE Intelligent Vehicles Symposium (IV2016), Gothenburg, Sweden*, pages 864–870, June 2016.

[60] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Soc. Series B (Methodological)*, 39(1):1–38, 1977.

[61] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust Euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, March 2005.

[62] David B. Wagner. Dynamic programming. *The Mathematica Journal*, 5(4): 42–51, Fall 1995.

[63] Meinard Muller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer-Verlag Berlin Heidelberg, Germany, Sept. 2007.

[64] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Single camera vehicle localization using SURF scale and dynamic time warping. In *Proc. 2014 IEEE Intelligent Vehicles Symposium (IV2014), Detroit, MI, USA*, pages 681–686, June 2014.

[65] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Vision-based vehicle localization using a visual street map with embedded SURF scale. In *Computer Vision —ECCV 2014 Workshops Proc., Part I, Zurich, Switzerland*, volume 8925 of *Lecture Notes on Computer Science*, pages 167–179, Sept. 2014.

[66] Haruya Kyutoku, Tomokazu Takahashi, Yoshito Mekada, Ichiro Ide, and Hiroshi Murase. On-road obstacle detection by comparing present and past in-vehicle camera images. In *Proc. 12th IAPR Conf. on Machine Vision Applications (MVA2011), Nara, Japan*, pages 357–360, June 2011.

[67] Ignacio Parra, Miguel Ángel Sotelo, David Fernández Llorca, and Manuel Ocaña. Robust visual odometry for vehicle localization in urban environments. *Robotica*, 28(3):441–452, May 2010.

[68] Dixiao Cui, Jianru Xue, and Nanning Zheng. Real-time global localization of robotic cars in lane level via lane marking detection and shape registration. *IEEE Trans. Intelligent Transportation Systems*, 17(4):1039–1050, April 2016.

[69] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Map-based probabilistic visual self-localization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 38(4):652–665, April 2016.

[70] Danfei Xu, Hernán Badino, and Daniel F. Huber. Topometric localization on a road network. In *Proc. IEEE/RSJ 2014 Int. Conf. on Intelligent Robots and Systems (IROS2014), Chicago, IL, USA*, pages 3448–3455, Sept. 2014.

[71] Henning Lategahn, Markus Schreiber, Julius Ziegler, and Christoph Stiller. Urban localization with camera and inertial measurement unit. In *Proc. 2013*

IEEE Intelligent Vehicles Symposium (IV2013), Gold Coast, QLD, Australia, pages 719–724, June 2013.

[72] Pratik Agarwal, Wolfram Burgard, and Luciano Spinello. Metric localization using Google Street View. In *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA2015), Seattle, WA, USA*, pages 3111–3118, May 2015.

[73] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Position interpolation using feature point scale for decimeter visual localization. In *Proc. 2015 IEEE Int. Conf. on Computer Vision (ICCV2015) Workshops, Santiago, Chile*, pages 90–97, Dec. 2015.

[74] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Single camera vehicle localization using feature scale tracklets. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E100-A(2): 702–713, Feb. 2017.

[75] Marc Sons, Martin Lauer, Christoph G. Keller, and Christoph Stiller. Mapping and localization using surround view. In *Proc. 2017 IEEE Intelligent Vehicles Symposium (IV2017), Los Angeles, CA, USA*, pages 1158–1163, June 2017.

[76] Hideyuki Kume, Arne Suppé, and Takeo Kanade. Vehicle localization along a previously driven route using image database. In *Proc. 13th IAPR Conf. on Machine Vision Applications (MVA2013), Kyoto, Japan*, pages 177–180, May 2013.

[77] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, June 1981.

[78] Peter Decker, Dietrich Paulus, and Tobias Feldmann. Dealing with degeneracy in essential matrix estimation. In *Proc. 15th IEEE Int. Conf. on Image Processing (ICIP 2008), San Diego, CA, USA*, pages 1964–1967, Oct. 2008.

[79] Gonzalo López-Nicolás, Carlos Sagüés, and José Jesús Guerrero. Parking with the essential matrix without short baseline degeneracies. In *Proc. 2009*

IEEE Int. Conf. on Robotics and Automation (ICRA2009), Kobe, Japan, pages 1098–1103, May 2009.

[80] Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Medical image registration using mutual information. *Proc. of the IEEE*, 91(10):1699–1722, Oct. 2003.

[81] F. Maes, D. Loeckx, D. Vandermeulen, and P. Suetens. *Image registration using mutual information*, pages 295–308. Springer US, Boston, MA, USA, 2015.

[82] Kensaku Mori, Daisuke Deguchi, Jun Sugiyama, Yasuhito Suenaga, Junichiro Toriwaki, Calvin R. Maurer, Hirotsugu Takabatake, and Hiroshi Natori. Tracking of a bronchoscope using epipolar geometry analysis and intensity-based image registration of real and virtual endoscopic images. *Medical Image Analysis*, 6(3):321–336, Sept. 2002.

[83] Daisuke Deguchi, Kensaku Mori, Yasuhito Suenaga, Jun-ichi Hasegawa, Junichiro Toriwaki, Hiroshi Natori, and Hirotsugu Takabatake. New calculation method of image similarity for endoscope tracking based on image registration in endoscope navigation. In *International Congress Series*, volume 1256, pages 460–466, June 2003.

[84] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *Int. J. Computer Vision*, 56(3):221–255, Feb. 2004.

[85] Paul Viola and William M. Wells III. Alignment by maximization of mutual information. *Int. J. Computer Vision*, 24(2):137–154, Sept. 1997.

[86] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. 1999 IEEE Int. Conf. on Computer Vision (ICCV1999) Workshops, Kerkyra, Greece*, pages 1150–1157, Sept. 1999.

[87] David Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, Nov. 2004.

[88] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110 (3):346–359, June 2008.

[89] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proc. 9th European Conf. on Computer Vision (ECCV2006), Part I, Graz, Austria*, volume 3951 of *Lecture Notes on Computer Science*, pages 440–443, May 2006.

[90] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. KAZE features. In *Proc. 12th European Conf. on Computer Vision (ECCV2012), Part VI, Florence, Italy*, volume 7577 of *Lecture Notes on Computer Science*, pages 214–227, Oct. 2012.

[91] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proc. 11th European Conf. on Computer Vision (ECCV2010), Part IV, Crete, Greece*, volume 6314 of *Lecture Notes on Computer Science*, pages 778–792, Sept. 2010.

[92] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. 2011 IEEE Int. Conf. on Computer Vision (ICCV2011), Barcelona, Spain*, pages 2564–2571, Nov. 2011.

[93] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *Proc. 2011 IEEE Int. Conf. on Computer Vision (ICCV2011), Barcelona, Spain*, pages 2548–2555, Nov. 2011.

[94] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. FREAK: Fast Retina Keypoint. In *Proc. 2012 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2012), Providence, RI, USA*, pages 510–517, June 2012.

[95] Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *Proc. 2005 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2005), San Diego, CA, USA*, volume 1, pages 772–779, June 2005.

[96] Point Grey Research. Ladybug3 12 MP FireWire 1394b, Accessed: 2017-10-16. URL: https://www.ptgrey.com/ladybug3-360-degree-firewire-spherical-camera-systems.

[97] Point Grey Research. Flycapture SDK, Accessed: 2017-10-16. URL: https://www.ptgrey.com/flycapture-sdk.

[98] Haruya Kyutoku. *A study on general obstacle detection using a forward-facing in-vehicle camera*. PhD thesis, Nagoya University, Nagoya, Japan, March 2016. [In Japanese].

[99] Gary Bradski. The OpenCV library. *Dr. Dobb's J. of Software Tools*, 25(11): 120–126, 2000.

[100] OpenCV Team. Open source computer vision library, Accessed: 2017-11-20. URL: https://opencv.org/.

[101] Mitsubishi Electic. Mitsubishi Electric Mobile Mapping Systems, Accessed: 2017-10-16. URL: https://www.mitsubishielectric.co.jp/mms/spec.html.

[102] Eric A. Wan and Rudolph van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proc. 2000 IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC2000), Lake Louise, AB, Canada*, pages 153–158, Oct. 2000.

[103] HERE International B.V. HERE, Accessed: 2016-11-29. URL: https://here.com/en.

[104] The University of Melbourne. ITS World Congress 2016 Grand Challenge, Accessed: 2016-11-29. URL: http://conference.eng.unimelb.edu.au/its-gc/.

[105] John Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[106] Sameer Agarwal and Keir Mierle. Ceres solver, Accessed: 2016-11-29. URL: http://ceres-solver.org/.

[107] Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–670, Oct. 1996.

[108] Hanno Scharr. Optimal filters for extended optical flow. In *Complex Motion —IWCM 2004 First Int. Workshop, Günzburg, Germany*, volume 3417 of *Lecture Notes on Computer Science*, pages 14–29, Oct. 2004.

[109] Khronos Group. OpenGL - The Industry's Foundation for High Performance Graphics, Accessed: 2017-11-20. URL: https://www.opengl.org/.

[110] Geoffrey Pascoe, William Maddern, and Paul Newman. Direct visual localisation and calibration for road vehicles in changing city environments. In *Proc. 2015 IEEE Int. Conf. on Computer Vision (ICCV2015) Workshops, Santiago, Chile*, pages 98–105, Dec. 2015.

[111] Laurent Delobel, Romuald Aufrere, Roland Chapuis, Christophe Debain, and Thierry Chateau. Towards automated map updating for mobile robot localization. In *Proc. 2017 IEEE Intelligent Vehicles Symposium (IV2017), Los Angeles, CA, USA*, pages 1342–1347, June 2017.