Feature-Based Localization Methods for Autonomous Vehicles

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

eingereicht von
Xiuyan Guo

Berlin 2017

Gutachter: Prof. Dr. Raúl Rojas
Freie Universität Berlin

Prof. Dr. Marco Block-Berlitz
Hochschule für Technik und Wirtschaft Dresden

Tag der Disputation: 11.04.2017

Declaration

Hiermit versichere ich, für die Erstellung dieser Arbeit alle Hilfsmittel und Hilfen angegeben, und die Arbeit auf dieser Grundlage selbständig verfasst zu haben.

Weiterhin versichere ich, dass die Arbeit noch nicht in einem früheren Promotionsverfahren in identischer oder ähnlicher Form eingereicht wurde.

Xiuyan Guo
February 2017

Acknowledgements

Four years have passed since began my PhD in Germany. It is full of ups and downs - the culture shock, the switch from Electrical Engineering to computer science. I would like to thank the following individuals and organizations for their sincere help and support, without whom I cannot succeed in this long-term and tough endeavor.

First and foremost, I would like to express my deep appreciation to Prof. Dr. Raúl Rojas for introducing me to the field of Artificial Intelligence and for being my supervisor when I started working in this amazing area, providing me with the priceless technical advice.

Secondly, I am sincerely grateful to all my colleagues at the Free University of Berlin, in particular all participants of AutoNOMOS project, without whom I could not have finished, for their support in my research for the duration of my studies in Germany. Especially, I wish to extend warm thanks to Prof. Dr. Daniel Göhring and Fritz Ulbrich for their insightful suggestions and hours on road tests. I would also like to express my gratitude to Tobias Langner, José Antonio Álvarez Ruiz, Ricardo Carrillo and Zahra Boroujeni for helping me in proofreading and the valuable comments. For proof-reading on this thesis, thanks also to my friends Lalminthang Kipgen, Yi Huang and Ningfei Li.

Furthermore, my special thanks also go to China Scholarship Council (CSC) for providing me the financial support, which helps me to realize my dream to study abroad. I would also like to express my gratitude to the Center of International Cooperation (CIC) at the FU Berlin, particularly our Program Coordinator Ms. Stefanie Kirsch, for helping us get ready for studying from our very first day.

Finally, I would like to thank my parents and my wife for their persistent encouragement and support, which gives me the courage to pass through difficult times and days of struggling.

Abstract

Autonomous vehicles are virtually regarded as the panaceas for the future of road transport due to numerous promising benefits. Thus, they have attracted wide attention from both academia and industry. Although associated technologies have been investigated and developed for decades, several obstacles still need to be overcome. One of the major obstacles is the total cost of the needed sensors. Therefore, it would be a long-term and effective solution to use less expensive sensors that can provide the same or even better performance.

The main focus of this dissertation is to design and implement a feature-based localization system. It aims to substitute the most expensive part of the present autonomous test vehicle, Applanix POS LV 510 which is four times more expensive than the vehicle. To do that, feature maps were first constructed through the log files of the test drive in Berlin. Then an algorithm is proposed to localize the test vehicle within the pre-built maps by using the Velodyne LIDAR and the Electronic Stability Program (ESP) associated sensors. The Velodyne LIDAR is employed to extract pole-like features from the previously mapped environments. Gyroscope and wheel speed sensors from the ESP are utilized to carry out the relative localization. The estimation does not need any GPS information after initialization. The performance of the proposed localization algorithm was evaluated through two datasets and the results indicate that it is comparable to the Applanix system. The real on-road tests also verified its effectiveness and robustness in terms of accuracy and precision. It is even more precise than the Applanix system as it shows higher repeatability.

The main innovations and contributions of this thesis can be summarized in three aspects. First, an innovative two-point localization scheme is proposed. It can greatly mitigate the influence of the wrong feature matching during the data association stage, thus it can get more accurate estimations. Second, an Ackermann constraint based trajectory smoothing method is proposed, which can smooth the trajectories especially during U-turns. Finally, the idea of using the online data to create feature maps is also evaluated and tested on the real roads. It is less accurate than the method of using the log files to create feature maps, but it can create city scale feature maps in a more efficient and convenient way.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

ATE	Along-Track Error
CAN	Controller Area Network
DARPA	Defense Advanced Research Projects Agency
DGPS	Differential Global Positioning System
DMI	Distance Measurement Indicator
DR	Dead Reckoning
DTE	Direct-Track Error
DTW	Dynamic Time Warping
EKF	Extended Kalman Filter
ENU	East-North-Up
ESP	Electronic Stability Program
FIR	Finite Impulse Response
FIS	Fachübergreifendes InformationsSystem
GDP	Gross Domestic Product
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Initiate Measurement Unit
INS	Inertial Navigation System

KF	Kalman Filter
LIDAR	Light Detection And Ranging
MDF	Mission Data File
OROCOS	Open Robot Control Software
RADAR	Radio Detection And Ranging
RANSAC	Random Sample Consensus
RMS	Root Mean Square
RNDF	Route Network Definition File
ROS	Robot Operating System
RTK	Real Time Kinematic
SIR	Sampling Importance Resampling
SIS	Sequential Importance Sampling
SLAM	Simultaneous Localization and Mapping
TOA	Time-Of-Arrival
UKF	Unscented Kalman Filter
UT	Unscented Transformation
WGS84	World Geodetic System 1984
XTE	Cross-Track Error

"mach es nicht so kompliziert, mach es einfach"

Raúl Rojas

1 Introduction

Two decades ago, when it came to autonomous driving, people must think it only existed in the science fiction. But now it has become a reality and self-driving cars are gradually becoming consumer cars in the near feature. Especially after the 2005 DARPA Grand challenge and the 2007 DARPA Urban challenge, autonomous driving has entered the public view and attracted interest from the public and companies a like. With the effort of the Google Self-Driving project, many research groups and other car manufacturers, autonomous driving has already achieved remarkable advances. However, many critical challenges still need to be overcome. From the technical perspective, the autonomous driving related technologies are not so reliable. From the cost perspective, main sensors are still expensive in comparison to the cost of the vehicles. From the legal view, few states in the United States have passed legislation and laws to allow testing autonomous vehicles on public roads. To realize fully autonomous driving, therefore, it still has a long way to go.

This dissertation focuses on addressing the fundamental problem of autonomous driving, localization. Most state-of-the-art autonomous test vehicles are equipped with commercial high accuracy GPS-based Inertial Navigation Systems (INS), such as Applanix POS LV. Such systems are quite expensive in comparison with their platform vehicles. This hinders the mass adoption and the development of autonomous vehicles. Therefore, our purpose here is to employ fewer and cheaper sensors to get the same order of magnitude of accuracy as that of commercial systems. To realize this goal, a feature-map based scheme has been successfully implemented. Thousands of times of tests and real on-road tests verified its effectiveness and robustness.

1.1 Motivation

1.1.1 Why Autonomous

Autonomous vehicles, also known as self-driving cars, belong to outdoor mobile robots. They employ many different types of sensors for localization and perceiving their environments. And they depend on the on-board computers to perform autonomous driving tasks. As a long-term goal, such vehicles will replace human drivers to drive passengers to their destinations. The mass adoption of autonomous vehicles has numerous benefits.

- **Safety**
  The widespread embrace of autonomous vehicles is widely believed to be a solution to reduce the rate of traffic accidents. This belief stems from the statistics data related to car accidents. The statistics results show that the leading cause of about 90% of reported car accidents results from human errors. Such errors are not limited to novice drivers but also including trained and experienced experts [1]. Such driving errors include drunk driving, fatigue driving, slow reaction time in emergencies and so on. According to a World Health Organization study, around 3400 people all over the world lose the lives on roads every day, equaling over 1.2 million deaths each year [2]. Since most of such traffic accidents are induced by human errors, autonomous vehicles could eventually reduce the rate of vehicle related deaths. They will eventually eradicate vehicle related accidents. Another exciting aspect is that physiological conditions will not be problems of driving, irrespective of age or physical disability [3].

- **Less Pollution**
  As many researchers predicted and the media reported [4], autonomous vehicles can increase fuel efficiency and reduce the pollutant emissions. Nowadays the traditional fuel vehicles are still the dominant form of transport around the world. And extensive adoption of electronic and autonomous vehicles still has a long way to go. Under such context, gas emissions from vehicles, especially the greenhouse gas, are the leading source of air pollution. Specialized control algorithms allow autonomous vehicles to drive under more efficient driving patterns, avoiding repeated speeding up and braking, and thus increase the fuel efficiency. The autonomous car sharing program will further reduce the air pollution [4].

- **Freedom of Time**
  For those who abandon public transport but choose to drive, the commute to work could be nothing but drive. And long-distance driving may cause fatigue to drivers, which will be prone to traffic accidents. Autonomous vehicles will release you from such tedious driving. This gives you more freedom to do whatever you would like to do, such as reading news or working on a report. Therefore rather than wasting hours on commuting, such moment would become a time for relaxation, preparation, which will make you more productive.

- **Space**
  Autonomous vehicles can save space in two ways. First, such vehicles powered by intelligent systems can drive much more smoothly and react faster than we human beings because of high update rates. This allows them to travel closer on roads. Second, car sharing can reduce the demand for cars, since people do not have to buy their own cars. The fewer household vehicles, the less parking lots will be needed. After taking customers to their destinations, the vehicles will pick up their nearby customers according to their demands, which will reduce the parking time. As a matter of fact, cars are used one hour per day on average, that means over 95 percent of the time cars are parked [5]. Thus increasing vehicle usage can also reduce parking lots. Saved spaces can be used for public area, which will benefit citizens.

Of course, the widespread adoption of the autonomous vehicles is not a panacea for all existing problems. Instead, it will present disadvantages as well. For instance, if fewer vehicles are in demand, it is a fatal blow to automobile related industries. Those industries contribute a huge percentage of GDP (Gross Domestic Product) in many countries, especially in countries like Germany (several important indicators can prove this [6]). Professional drivers would no longer be needed and probably the first released cars are too expensive for most ordinary consumers. However, the advantages outweigh the drawbacks and are the motivations for developing autonomous vehicles.

1.1.2 Why Localization and This Work

Try to imagine that you walk to the nearest supermarket from your home with eyes closed. How could you make it without other people's help? Having a good impression of the surroundings, deciding which path to go and knowing the location

of every step are the minimum requirements. From this simple everyday example, a conclusion can be drawn that if we want to go somewhere, at least we should continuously localize ourselves in either a mental or a real map. A mobile robot also has to consider those problems and ask similar questions, such as "where am I?", "where do I have to go" and "how could I get there?" [7]. To answer the former question in robotics is involved in the eternal task, localization. It is a procedure that a mobile robot estimates its pose, including position and orientation regarding to a defined global coordinate. As many papers narrated, precise and reliable localization is a prerequisite for mobile robots to perform high level tasks [8]. As a special outdoor mobile robot, having precise knowledge of its ego location is a must for an autonomous vehicle. Without precise localization, safe self-driving cannot be achieved, as localization is the basis of navigation and avoidance of obstacles and pedestrians. Thus, localization is an important topic in the autonomous research.

For laypersons, localization should be a simple task since they may hear a lot about the American Global Positioning System (GPS). Although the GPS is the most widely adopted system for localization, a stand-alone GPS unit cannot provide the required order of magnitude of the localization precision for autonomous vehicles. As a general rule of thumb, safely driving an autonomous vehicle in urban environments requires the localization accuracy within a few centimeters [9]. Due to the technical restrictions, a stand-alone GPS unit cannot satisfy this requirement. Therefore, most state-of-the-art autonomous research vehicles are configured with the GPS-based Inertial Navigation Systems [10, 11, 12]. These systems function well in vast open areas but suffer from multipath propagation and satellite blockage in dense urban environments [13]. Apart from that, such systems are usually considerably expensive compared to the total cost of the whole systems. In this dissertation, an alternative solution is presented with the aim of using cheaper on-board sensors to achieve equivalent localization precision. For the purposes of simplicity and high accuracy, the proposed solution is based on pre-built feature maps. When a map is available, localization becomes a straightforward problem for the robot pose estimation. But in the long run, to achieve fully autonomous driving relied on pre-built maps is not the way to go. Mostly, pre-built maps are not available. And building a map is both a time-consuming and tedious task, especially when manual annotation is required. Under such circumstance, being capable of exploring unknown environments enables the robot to fulfill fully autonomous driving. To give robots such ability, researchers devote their efforts to studying the fantasy topic, simultaneous localization and mapping (SLAM).

1.2 Brief Introduction of Autonomous Driving Projects

By now, the benefits of developing autonomous vehicles and the importance of localization have been discussed. In this section, the significant autonomous driving projects will be introduced, especially the milestone moments. Autonomous project at the Free University of Berlin will also be presented, on which the author works.

1.2.1 Important Autonomous Driving Projects

Human beings have long expected to implement autonomous robots. But in the realm of autonomous vehicles, experiments began in the 1920s. Promising research took place in the 1950s. And a substantial progress was made in the late 1970s. Then researchers realized that vehicles have to perceive their environments and make decisions independently [4]. A lot of pioneering work was carried out by universities and car manufactures. However, in this section, only important autonomous driving projects will be introduced. The DARPA Grand Challenge and the DARPA Urban Challenge were the most important events in the history of autonomous driving. They have been speeding up the development of autonomous driving. And the Google Self-Driving Car Project is the most widely reported project. It is a promising project with the aim of delivering autonomous cars to the masses. A more detailed introduction can be found in [4, 14].

- **2004-2005 DARPA Grand Challenge**
  To promote autonomous technologies, the Defense Advanced Research Projects Agency (DARPA) launched the DARPA Grand Challenge. The winners of participated autonomous ground vehicles were supposed to complete an off-road course without any human intervention with a time limit. The first competition was held in 2004 in the Mojave Desert region of the United States, along a 150-mile route following the Interstate Highway 15 from Barstow, California to Primm, Nevada. The exact route, defined as an ordered list of GPS coordinate waypoints, was kept as a secret until shortly before the race. No participated vehicles finished the route and the best result was 11.78 km of the course. Therefore no winner was declared and the second competition was scheduled for 2005.

  In 2005, the DARPA updated the challenge and doubled the prize money to $ 2 million. It also took place in the Mojave Desert region but with a 132-mile route. This time five teams completed the course and four of them satisfied the rule to finish within 10 hours. The first prize went to Stanley, from Stanford

(a) Stanley Won the 2005 DARPA Grand Challenge[1]

(b) Touted as the most difficult section: Beer Bottle Pass [2]

Figure 1.1: The 2005 DARPA Grand Challenge.

Racing Team led by Sebastian Thrun, which completed the course within 7 hours. The moment was captured in Figure 1.1 (a) when Stanley successfully finished the whole course.

(a) Boss Won the 2007 DARPA Urban Challenge[3]

(b) The final event course map of the Urban Challenge[4]

Figure 1.2: The 2007 DARPA Urban Challenge.

- **2007 DARPA Urban Challenge**
  To further speed up autonomous driving technologies, the DARPA Urban Challenge set the competition in an urban traffic environment (see Figure 1.2). The contest required the participated vehicles to perform complex maneuvers such as merging into traffic, avoiding participants and obstacles, parking and negotiating intersection, while obeying all traffic rules [15]. The competition

[1] http://i.i.cbsi.com/cnwk.1d/i/blog/buzz/StanleyWins_300.gif
[2] https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2005)
[3] http://sportruck.com/news/2007-DARPA-Urban-Challenge-Tahoe/4.jpg
[4] http://archive.darpa.mil/grandchallenge/docs/UC07UCEBrochuretoPrint.pdf

course was chosen at the site of the now closed George Air Force Base in Victorville, California [4]. The 96 km involved urban area course was required to be completed in less than 6 hours. This time six teams managed to finish the entire course and four out of them completed it within 6 hours. The Tartan racing team with its entry "Boss" from Carnegie Mellon University won the $ 2 million prize. Teams from Stanford University and Virginia Tech won the second and third place, respectively.

- **The Google Self-Driving Car Project**
  After DARPA Grand and Urban Challenge, the Google self-driving car project attracts attention from both the Media and the public. Google started the project in 2009. The project attracted many top scientists and engineers from Carnegie Mellon University and Stanford University who won twice the DARPA challenge prize. To carry out research, several different types of cars have been converted into autonomous cars. They developed their own prototype cars as well. Different kinds of sensors are mounted on those cars for the purposes of localization and perception. Well developed software help them making driving decisions. They claimed that they had self-driven over 1 million miles on the streets of Mountain View, California in 2015 [16]. Safety drivers are always required during the testing, who should take over driving if needed and watch over the cars' driving behavior. They still involved in some minor accidents, although they emphasized that no accidents have occurred under the self-driving mode. Besides the self-driving research, they also try to commercialize their cars (see Figure 1.3).

1.2.2 Autonomous Project at the Free University of Berlin

The autonomous driving project at the Free University of Berlin was initiated in 2006, led by Prof. Dr. Raúl Rojas. To carry out research, three autonomous vehicles were developed successively, named as "Spirit of Berlin", "MadeInGermany" (a.k.a MIG) (Figure 1.4) and the autonomous electric vehicle "e-Instein". Spirit of Berlin participated in the 2007 DARPA Urban Challenge and reached the semi-finals of the competition. It demonstrated the ability of safe driving in urban environments. After the Urban Challenge, the second car, MIG was built to test in heavy traffic environments [4]. Since MIG is the present experimental autonomous vehicle in our lab, it will be narrated in the rest of this section. One more thing needed to be mentioned is that in October 2015, this test autonomous car set a 2,414-kilometer autonomous driving record in Mexico without much intervention. It has become

(a) A row of Google Self-driving cars[5]

(b) The prototype of Google self-driving car[6]

Figure 1.3: The Google Self-driving Car project.

the longest autonomous driving ever achieved in Mexico and Latin America [17].

MIG is a modified 2010 Passat Variant 3C with the support of its car maker Volkswagen. The modification consists of the Drive-by-Wire system, localization system, multiple sensors and cameras. The Drive-by-Wire system is a bridge between the vehicle and computers. It allows the computers directly to access the vehicle's actuators via the CAN bus. Thus, the engine, steering, brakes and other actuating units can be controlled by computers. So it is the important interface which makes the autonomous driving possible. The localization system currently utilized is the

[5] http://www.autoblog.com/
[6] https://www.google.com/selfdrivingcar/

Figure 1.4: Spirit of Berlin and MadeInGermany [18].

Applanix POS LV system. The system integrates all the information from the GPS receivers, an Initiate Measurement Unit (IMU) and a Distance Measurement Indicator (DMI). So it can provide more accurate positional estimation than a stand-alone GPS solution, especially during the GPS outages. Therefore it is an ideal localization system for working in urban environments. As for sensors, they are mainly LIDAR (also written as LiDAR or lidar in various publications) and radars. The most important LIDAR sensor is definitely the "Velodyne HDL-64E S2", which is a spinning LIDAR sensor mounted on top of the vehicle. Due to the spinning feature and well distributed 64 lasers in vertical, it provides a 360° horizontal field of view (FOV) and a 27° vertical field of view [19]. It produces over 1.3 million point cloud measurements (bearing, range, intensity) per second and the range can be up to 120 meters with the accuracy of less than 2 cm. Thus, this sensor can provide enough information to build 3D maps of its environment and can be used to correct the vehicle's position. The radar sensors are also installed on the vehicle because they can determine the speed of objects for obstacle avoidance based on the Doppler effect. However, the exact positions of objects are difficult to obtain through this effect. To overcome the constraints of the radar, other LIDAR sensors are mounted around the vehicle. By combining those two kinds of sensor, accurate position and speed information of the surrounding objects can be acquired. Besides, some laser scanners are used to detect street curbs and lane markings. There is still one more thing needed to be noticed, the emergency stop buttons. Since autonomous technologies are still under development, the vehicles cannot handle all unexpected emergencies or system failures caused by software issues. Those emergency stop buttons enable the safety driver to take over control at any moment [18].

1.3 Contributions of the Dissertation

The main innovations and contributions of this dissertation are three aspects.

(**A**) An innovative two-point feature based localization scheme is proposed. It turns out to be a simple and elegant approach. The extended Kalman filter framework is applied to deal with data fusion problem like most of the published works. But this work is different from the classical EKF localization scheme in the way of handling the measurement update stage. Most papers use a single measurement to calculate the measurement update, under the assumption that the data association problem is known. But this is not the case in practice. This work employs the two-point matching method to solve this problem, which can greatly mitigate the influence of the wrong feature matching during the data association stage thus can get more accurate estimations. The performance of the proposed localization algorithm was evaluated through two recorded datasets. And the results indicate that it is comparable to the Applanix system. The real on-road tests also verified the effectiveness and robustness of the proposed localization scheme.

(**B**) In addition to using a second EKF to smooth the trajectory, an Ackermann constraint based trajectory smoothing method was proposed, which can smooth the trajectories especially during U-turns.

(**C**) We are probably the first group to use the online data to create pole-like feature maps. This method was also evaluated in this thesis and tested on the real roads as well. It is less accurate than the method of using the log files to create feature map. But it can create city scale feature maps in a more efficient and convenient way.

1.4 Thesis Structure

This thesis will present a pole-like feature-based localization scheme for autonomous vehicles. In the next chapter, widely used data fusion algorithms will be introduced and compared, including Kalman filter and particle filter. Chapter 3 will discuss the sensors utilized in this scheme. Chapter 4 presents the core work of this thesis. And chapter 5 will evaluate the performance of the presented scheme. The last chapter concludes the whole work, and the outlook for future work will be extended as well.

"Make things as simple as possible but not simpler."

Albert Einstein

2 Related Work

Mobile robot localization is an active and attractive research field and myriads of papers addressing this problem have been published. They can be classified as the Kalman filtering based and the particle filtering based. This chapter will introduce the related algorithms and make a comparison between them. Mapping is also important as it affects the map construction and computational complexity. Thus, some conventional mapping techniques will be introduced in the first section. The related work of mapping and localization will be presented as well.

2.1 Mapping

For mobile robots, navigating from one location to another without a map is not an easy task as this involves in localization and path planning. Localization provides instant location information in some given form. And representing the observed world on a map would be an ideal choice. Path planning needs to know the initial position and the target position, and these two positions are properties on a map. Thus a pre-built map would be a prerequisite for lots of complicated and high-level applications. For autonomous vehicles driving in urban scenarios, they have to comply with traffic rules according to given traffic signs and lane markings. If these rules have not been transformed into map properties, an autonomous vehicle needs to sense them in real-time with a high degree of robustness. But nowadays, computers are still not good at perceiving related tasks. Therefore, if traffic related information is embedded into a prior map, it will greatly improve the robustness and reduce overall complexities. However, a map is not always being available, especially when a robot explores an unknown environment. Under such scenario, the robot has to perform the well-known task, Simultaneous Localization and Mapping.

To create a map, we first need to define a coordinate frame or reference so that the position of a point or feature can be represented unambiguously. Essentially, a map is a set of sensor-measuring records and it consists of a list of objects' locations in the environments or coordinates in a defined reference frame. Thus, a map is an abstract representation of its environment [20]. Since the sensor measurement is prone to uncertainty, probabilistic techniques are widely employed to deal with this problem. Besides, maps can be created in two dimensions and three dimensions. Two-dimensional maps are easy to build and much efficient in computing. And three dimensional maps model environments with more information. So they can provide more accurate localization results than that of two-dimensional maps. For the purposes of simplicity, maps presented in this work are in two dimensions. Finally, the most widely adopted mapping techniques can be classified into three categories, known as location-based, feature-based and topological maps. However, the location-based and feature-based maps can be certainly grouped under the metric maps [21, 22].

1. **Location-Based Maps**
   The occupancy grid map is the famous and widespread location-based map representation. Owing to the simplicity in implementation and application, it has been widely used for environment modeling in mobile robotics. It was first introduced by A. Elfes [23]. It is a volumetric map. It offers information not only about the objects but also the free space in the environment by fine grained metric grids [21]. What is more, it is capable of representing arbitrary features and offering sufficient detail. As each grid cell must be defined as being either occupied or free, the maps consume more memory space than other maps. Since the correlation among grids is not taken into consideration and the robot's pose exists uncertainty, it cannot guarantee the map consistency in large environments [24]. But the beauty of the occupancy grid maps lies in the fact that it is perfect for mobile robots to navigate in unoccupied spaces [21].

2. **Feature-Based Maps**
   This map, just as its name implies, extracts typical geometric features from the environment to model its perceived world. Tree trunks, poles of traffic lights and other pole-like objects in the street can be treated as point features. The straight street curbs and wall-wall intersection or wall-ground lines can be identified as lines. Normally only the location information of the features is utilized to build the map. But other information, such as the features' size, color and LIDAR intensity value can be used during feature matching stage

as well [24]. Feature matching also refers to the data association problem, which is trying to find the correspondence between observed features and features on the map. Data association is both difficult and important. Accurate localization depends on the correct data association. And wrong data association will lead to inconsistent in localization. Since the feature-based map only stores well-chosen features' locations, which makes it smaller in file size and possible to represent a large-scale environment. In robotics, features normally correspond to distinct physical objects. Thus, those objects are known as landmarks [21]. One disadvantage of such maps is that they typically do not provide any information to distinguish between drivable and non-drivable areas. Thus, they are not suitable for the path planning task.

3. **Topological Maps**
   Topological maps choose graphs to describe the connectivity between different locations of features [22, 24]. It is different from the former two fashions which present the environment in Cartesian coordinate frames. The graphs include nodes and arcs which correspond to possible travel paths. Nodes represent important features, such as entries and exits, and arcs in the graph indicate that direct paths exist between the neighbor nodes. Thus, they use graphs rather than specific geographic information to present the environment and they will not consume much storage, which is suitable for fast path planning tasks. One of the main drawbacks of using such maps is wrong recognition in complex environments where two places look alike. Under this circumstance, the logic of the map will be destroyed and the robot cannot estimate its position [24]. The famous topological map format is the Route Network Definition File (RNDF). It was first used in the 2007 DARPA Urban Challenge and later by many autonomous car projects. In the last part of this section, a brief introduction of the RNDF will be presented.

The organizer of the DARPA Urban Challenge, DARPA introduced the RNDF which conveys the road network information of the racecourse. It is a high-level topological map which defines accessible areas for navigation [4, 25]. It uses road segments to represent structured areas and one or more lanes to make up a segment. The free-travel 'zones' are introduced to define unstructured areas, used for like parking lots. Extra information, such as way-points and stop signs, is provided in the RNDF as well. The RNDF also provides the latitude and longitude position information of checkpoints. They are key locations on lanes and utilized for specifying the ordered set of goal locations in the navigation mission file, known as the Mission

Data File (MDF). The MDF also includes the speed limit for each segment. After participating in the DARPA Urban Challenge, our team continues to use the RNDF for navigation. To get rid of its flaws, a new graph map named RNDFGraph was developed [4]. To create our own RNDF from scratch, the first step we manually drive the test vehicle in the expected areas. Then we will resort to the RNDFGraph editor to create maps by playing the recorded log files. Although the graphic editor relieves the burden to create digital maps, it still needs much manual editing work, such as adding speed limits to all road segments. Thus, a smarter and highly automatic editor is in demand in the long run.

2.2 Localization

In robotics, localization is one of the most important and fundamental tasks. For a mobile robot, knowing the precise real-time location is a prerequisite for performing all the high-level tasks [21]. The essence of the localization problem for a mobile robot is to estimate the pose related parameters. Those parameters include position and orientation with regard to a given reference frame. Localization can also be seen as establishing the correspondence between the measurements and features within a map. However, the localization problem is still difficult, though it has been extensively studied in the communities. The reasons are threefold. First, the uncertainty is widely existed in estimation. Second, the perfect motion model is hard to approximate. Last one is the unobserved states in practice. Thus, this section will introduce and compare both the absolute and relative localization scheme. Then the state-of-the-art LIDAR-based localization works will be introduced.

2.2.1 Absolute and Relative Localization

The absolute localization is a global localization solution which relies on the GNSS (Global Navigation Satellite System) constellations or landmarks to restore the position and orientation information with regard to a global reference frame. As one component of the GNSS, although the GPS technology is a terrific engineering achievement, the principle behind the technology is pretty simple. A GPS receiver employs a complex version of mathematical technique, trilateration to determine its location and speed. GPS is a time-of-arrival (TOA) system, which means that it estimates the distance from a receiver to a valid satellite by multiplying the speed of electromagnetic waves in free space by the transit time of a signal [26]. The time is critical as three nanoseconds will lead to the range error up to one meter [27]. Out

of the cost consideration, the clocks used on the user side are inexpensive crystal clocks, which are not so precise as that of used on GPS satellites. Moreover, the received power of GPS signal is so weak that it can be affected by many factors, such as the troposphere and tree canopies. A clear view of the sky is a basic requirement. So upon driving in tunnels, and underground parking lots, GPS is out of work. In inner cities, tall buildings can introduce the multipath effects, further deteriorating the precision. Because of these reasons, the GPS is often outage and the localization accuracy is not so high, on the order of meters. To get better performance by compensating external and systematic influence, Assisted GPS (A-GPS) and Differential GPS (DGPS) were developed. For DGPS, it heavily relies on nearby reference stations to broadcast the corrected information that can be used to improve accuracy of local GPS results. In addition, the update rate of GPS is relatively slow, at a frequency of 1 Hz, which means every 27 meters will produce a value if traveling at a speed of 100 km/h. Therefore, GPS alone cannot provide safety autonomous driving through city streets, as mere centimeters error may cause collisions.

Another absolute localization method is based on landmarks, which is widely employed in communities. Landmarks can be either natural objects [28, 29], such as trees or artificial ones [7, 30, 31], such as reflectors, radio beacons and street-lamps. They also can be classified as either active or passive according to how sensors sense them, specifically speaking emitting energy or not. In this work, pole-like features in the urban environments are selected as landmarks. The localization scheme based on these features is more accurate and robust than the GPS way. This conclusion can be proved in the following chapters.

The relative localization technique is a local localization technique. It uses onboard sensors and kinematic models to estimate the robot's pose relative to its initial pose. The popular dead reckoning belongs to the relative localization. Dead reckoning is relatively precise within a brief period of time but suffers drift in the long term due to the inherent drawbacks. The main drawbacks include the inaccurate kinematic model and the drift accumulation. First, the kinematic model is not always accurate, as it is hard to take external conditions and wheel slippage into consideration. Second, as the estimation integrates measurements over time, the drift accumulates as well. Since the method itself has no other mechanisms to mitigate the accumulated error, it cannot avoid drift over long time scales and the error grows without bound. On the contrary, the absolute localization does not have drift problem and can be used to correct drifts. In practice, therefore, combining these two different techniques to obtain better results is widely employed.

Both absolute and relative localization have advantages and drawbacks, as sum-

marized in Table 2.1. Combining them can obtain more accurate and robust measurements, as they are complementary to each other. Therefore, many research teams including our team employ the Applanix POS LV as the localization system. It combines the strengths of both relative and absolute localization. Therefore it can provide good position results even when GPS is outage. The RMS localization error in X and Y directions are about 40 cm when the GPS signal is not available [32]. However, sudden position discontinuities would happen upon entering or leaving a GPS outage stage or switching DGPS reference stations [33].

Table 2.1: Comparison between relative and absolute localization

	Relative Localization	**Absolute Localization**
Reference	Relative to its initial position	Global reference frame
Requirements	No particular	Conditions must satisfy
Error	Error accumulated Drift over time	No drift Independent of time
Application	Widely used	Possibility of outage

Note: The cells with red background color represent merit of features.

2.2.2 LIDAR-Based Localization

A myriad of papers focus on studying the localization problems. Different papers may use different sensors, such as GPS, LIDAR and Cameras. This work is based on LIDAR. So here we only introduce LIDAR-based works.

Over the last decade, the LIDAR-based localization scheme for autonomous vehicles has gained remarkable achievements, especially after introducing the 3D LIDAR sensors. For instance, in 2004, Scheunert et al. [29] proposed a precise vehicle localization scheme by using multi-sensors and landmarks. The motivation is that DGPS is unreliable in difficult areas. In their work, they also relied on the LIDAR sensor to detect the natural landmarks. By fusing all the measurements, the performance has improved dramatically, especially after utilizing the landmarks. Therefore, the availability of landmarks clearly influences the performance. Weiss et al. [34] used a high accurate feature map, GPS, dead reckoning sensors and an IBEO laser scanner to estimate a vehicle's ego state in urban scenario. They proposed a Triangle-Association algorithm for associating landmarks with the data obtained

from the laser scanner and correcting vehicle's pose. But the authors did not provide any specific value regarding the localization results. In [35], they used GraphSLAM-like off-line relaxation techniques to create a 2D orthographic ground infra-red reflectivity map of flat road surfaces by integrating Velodyne LIDAR data with GPS, IMU and odometry. Then they used a particle filter to align LIDAR measurements with the reference data in the *a priori* map to localize a moving car. Experimentation results showed that a reliable real-time centimeter-accurate localization was achieved. However, it seriously relies on maps. This may lead to failure due to many changes, like road paint changes or in adverse weather, like a snow-covered road. In addition, the map takes much memory, requiring about 10 MB per mile of storage [36] which may limit its adoption. The authors extended this work in [37] by modeling the environment as a probabilistic grid, so that a Gaussian distribution can be assigned to each cell regarding the infra-red remittance. Localization practice in many dynamic environments proved that this extension outperformed previous work. However, it is not absolutely independent of tough weather conditions or changing environments.

Wolcott and Eustice [38] proposed a method that used only one monocular camera to localize a vehicle within an *a priori* 3D map. The map data was collected by a four Velodyne HDL-32E LIDAR equipped vehicle. The essence of the algorithm is to maximize the so-called normalized mutual information between live camera view and many synthetic views. And the synthetic views are generated from the prior 3D map at a guessing pose. Results showed that it got similar localization accuracy as that of [35], but with a much cheaper sensors. In [39], the same authors proposed a scan matching algorithm through multi-resolution Gaussian mixture maps to localization a car under adverse weather or poorly textured roadways.

Many researches have been employing feature patterns for localization in large-scale outdoor environments, which are similar to this work in spirit. Claus Brenner [40, 41, 42] and his team [43, 44] have studied this problem through different methods. In [40], the author explored the method that uses local pole patterns to carry out global localization. First, they created a global map off-line by extracting local descriptors from the raw data recorded by high accuracy and high resolution LIDAR sensors. Then vehicles configured with low-end sensors tried to estimate their global position by recovering part of these descriptors. The major focus was on exploring of the local pole patterns. He also proposed a pole-like feature extraction approach for driver assistance systems [41] and evaluated the performance under simulation when a trajectory was given. The results proved that relative positioning is possible through poles and high accuracies can be expected. [43] presented a

method to localize vehicles by establishing the correspondence of triangulated point patterns. The proposed path-like point pattern matching scheme for localization in spirit is the same as the Delaunay triangulation-based methods for fingerprint verification [45]. The matching step tried to match a set of observed triangles with a set of reference ones in a given map regarding their geometric similarities. The experimental test gave a promising result that the method could be executed in real-time and was robust enough to overcome random errors. Alexander Schlichting [44] also presented a vehicle localization scheme which was based on multi-layer automotive laser scanners for feature extraction and local patterns for matching. The authors employed extracted pole-like features and building facades from laser point cloud to create feature patterns descriptor, unlike previous methods which only utilized poles to build descriptors. It also relied on matching descriptors to localize the vehicles. Results showed that the proposed matching algorithm could dramatically reduce the number of false matching compared with a nearest neighbor matching solution.

Researchers from Oxford University proposed several methods for vehicle localization by using LIDAR. In [46], the authors proposed a low cost LIDAR-based localization method for autonomous vehicles. A local 3D map was created by utilizing the odometry information to estimate the pose of the vehicle and only one 2D push-broom LIDAR to scan the environment. It tried to localize the vehicle by matching the local 3D map with a previously built global 3D map, and gained centimeter-level accuracy. Although this implementation cannot run in real-time, it does indicate that Velodyne can be replaced to some extent. This work was extended in [47] by using two 2D LIDAR systems. They used the horizontally mounted one to infer linear and rotational velocity and the declined one to scan the surroundings. With these two fixed LIDARs, velocity and position can be estimated simultaneously. As an extension to previous work of [48, 49], an experience-based localization scheme was presented in [50]. It is a promising approach for future vehicle localization as it takes changing environments into consideration. A novel mapping and localization approach allows vehicles to have multiple "memories" of the same location under different experiences. This is different from a traditional map which relies on a single static global representation. New driving experience can always be added to an existing map and the best experience can be used to match current driving experience for localization. Over one year test-drive demonstrated that this method could provide real-time centimeter-level localization accuracy.

This work employed the LIDAR for feature detection as well but only used pole-like features to create feature maps. It makes the created maps smaller than other

methods as they just store the features' position and uncertainties. Further, this method is independent of the environment changes. And the main algorithm is based on two-point matching method. The execute time for the feature matching and localization is less than 1 ms, which makes the real-time on-line operation possible. Besides, we are the first group which used the online data to create feature maps and successfully tested this idea on the real road. Although almost every method provided some experimental results, it is difficult to compare them because of a lack of the general benchmark. Therefore, this work used our group's own datasets for performance evaluation and execution of field tests.

2.3 Filtering Foundations

Filtering has become one of the most important tools on signal and information processing, as it is difficult to generate or obtain clean signals in practice. For instance, sensors are imperfect and are prone to picking up noise in measurements, which contributes to the uncertainties in measurements and estimation. Thus, in robotics, researchers prefer to employ probability tools, Bayesian filters, to deal with this problem. Filtering has been widely adopted for dealing with uncertainties and data fusion. In this section, popular filter algorithms, such as Kalman filter and particle filter will be briefly introduced, and some notable features are summarized and compared in Table 2.2.

Table 2.2: Comparison among the mainstream filters

	System/Update Model	Noise Model	Application
KF	Linear	Gaussian distribution	Optimal performance
EKF	Moderate non-linearities		Difficult to implement Difficult to tune
UKF	Highly non-linear		Improved performance over EKF
PF	No particular	No particular	Best performance for non-linear, non-Gaussian

2.3.1 Kalman Filter

Kalman Filter (KF) is a famous and fundamental filtering technique. Kalman filter and its variants are widely used for mapping and localization for mobile robots. It is an ideal filter for linear and Gaussian distributed systems. It is also the optimal filter as it is derived under the criteria of minimizing the mean square error. And it satisfies nearly any kind of relevant criterion of optimality [51]. Both pros and cons of this algorithm are the Gaussian distribution. The beauty of the special Gaussian distribution is that multiplying and adding two Gaussian functions still get another Gaussian function, a consistent and elegant result. Its drawback lies in the fact of narrowing its application within the normal distribution uncertainty.

As derived from the Bayesian equation, the Kalman filter is a recursive filter, which consists of two stages, motion prediction and measurement update. It works recursively in these two stages. Motion prediction is based on the kinematic model or simply dead reckoning, which will increase the uncertainty. This stage is also named as time update stage. Then the filter incorporates measurements to update the estimation, which will reduce the overall uncertainty. For the sake of convenience and intuition, equations will be utilized to express these two stages. The equations used here follow their original formulations in [52] and they are described at discrete time serials. The derivation of the Kalman filter is beyond the scope of this thesis. Refer [21, 51, 53] for a better understanding or an overview of the mathematical derivation of the Kalman filter or the extended Kalman filter equations.

1. **Motion Prediction**
   In general, the Kalman filter updates the state of a system from its previous state at time *t-1* to present t by adding the control inputs. The estimation of the state is expressed as the mean of the Gaussian distribution. The variance of estimation is also estimated. The process can be expressed in the following equations (2.1) and (2.2).

$$\hat{x}_t^- = A_t \hat{x}_{t-1} + B_t u_t \tag{2.1}$$

   where:

   - $\hat{x}_t^-$: is the predicted state vector at time t
   - A_t: is the state transition matrix
   - B_t: is the control input matrix
   - u_t: is the control vector at time t

$$P_t^- = A_t P_{t-1} A_t^T + Q_t \tag{2.2}$$

where:

- P_t^-: is the predicted covariance in state space at time t
- P_{t-1}: is the covariance in measurement space at time $t-1$
- Q_t: is the process noise covariance matrix at time t

After $A_t P_{t-1} A_t^T$, the covariance is transferred from measurement space into state space so that the process noise covariance Q can be added to the uncertainty estimation. This is the reason why this step increases uncertainty of estimation.

2. **Measurement Update**
   The measurement update incorporates measurements to previous estimation. So a better estimation is supposed to get and its uncertainty will be lower as well. The first task during this stage is to compute the Kalman gain, K_t. The Kalman gain is a ratio, showing the uncertainty ratio between the prediction and measurement. If the estimation is more confident about the measurements, it will put more weight on the Kalman gain, and vice versa. After computing the Kalman gain, the posterior state can be estimated with more confidence.

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \tag{2.3}$$

where:

- K_t: is the Kalman gain at time t
- H_t: is the transformation matrix which projects the state vector into the measurement domain at time t
- R_t: is the measurement noise covariance matrix at time t

$$\hat{x}_t = \hat{x}_t^- + K_t (Z_t - H_t \hat{x}_t^-) \tag{2.4}$$

where:

- $\hat{x}_t$: is the state vector after incorporating measurement update t
- Z_t: is the measurement vector in the measurement domain

$$P_t = (I - K_t H_t) P_t^- \tag{2.5}$$

where:

- P_t: is the posterior error covariance estimate t

2.3.2 Extended Kalman Filter

The Kalman filter assumes the motion and measurement equations are linear and the uncertainties of all the random variables follow the Gaussian distribution. In practice, however, the assumptions of optimality are difficult to satisfy due to the nonlinearities of the system model or the measurement model. It is common to apply the Kalman filter extensively by easing the assumptions of optimality. Hence the extended Kalman filter (EKF) was introduced by linearization of systems described by non-linear equations, such as through a first order Taylor series approximation. The extended Kalman filter has almost the same equations as the Kalman filter except some minor difference. In the EKF, the A_t in the equation (2.1) is the Jacobian matrix of partial derivations of process update function with respect to the state vector. And the H_t in the equation (2.4) is the Jacobian matrix of partial derivative of the measurement function with respect to the state vector.

The EKF has become a standard and widely adopted technique to address non-linear estimation. However, the main drawback of the EKF is the approximation by means of the first order truncation. It can introduce error in the posterior estimation and may lead to suboptimal performance or even divergence. This becomes serious when the non-linear models are approximated and the higher order terms in the Taylor series are more important than the first order term. In order to cope with this disadvantage, a new improvement of the extended Kalman filter was proposed, the Unscented Kalman Filter by Julier and Uhlman [54].

2.3.3 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) was introduced to deal with the approximation issues of the EKF, based on the intuition that approximating a Gaussian distribution is much easier and more accurate than approximating a highly non-linear function [55]. Instead of approximating the non-linear process and observation models as the EKF, the sigma points of the UKF directly use the true non-linear models. It works similar to the particle filter, using samples to represent the state

distribution. But the major difference is that the UKF chooses a minimal set of representative sample points to capture the distribution of the state completely, while the particle filter uses as much as possible samples to represent its state distribution. Compared with the EKF, the UKF captures the posterior distribution more accurate than that of the EKF, specifically speaking to the third order of Taylor series expansion for any nonlinearity [56].

The unscented transformation (UT) is the essence of the UKF. It is a method for computing statistics of a random variable which undergoes a non-linear transformation [54]. A couple of equations are employed to elaborate this transformation, following the equation style in [57]. Consider a L-dimensional random variable x with mean $\overline{x}$ and covariance P_x through a non-linear function, $y = g(x)$. $2L+1$ weighted sigma points are utilized to calculate the statistics of y as the flowing equation (2.6).

$$\begin{aligned} x_0 &= \overline{x} & \omega_0 &= \frac{\lambda}{L+\lambda} & i &= 0 \\ x_i &= \overline{x} + (\sqrt{(L+\lambda)P_x})_i & \omega_i &= \frac{1}{2(L+\lambda)} & i &= 1,..,L \\ x_i &= \overline{x} - (\sqrt{(L+\lambda)P_x})_i & \omega_i &= \frac{1}{2(L+\lambda)} & i &= L+1,..,2L \end{aligned} \tag{2.6}$$

where:

- ω_i: is the weight of the ith sigma-point and the sum of all weights equal 1
- λ: is a scaling parameter
- $(\sqrt{(L+\lambda)P_x})_i$: is the ith row (or column) of the matrix square root of the weighted covariance matrix $\sqrt{(L+\lambda)P_x}$.

Then the non-linear function, $y = g(x)$ is applied to transform these sigma points as the equation (2.7). The mean and covariance for y can be calculated using the standard statistics equation.

$$y_i = g(x_i) \qquad i = 0,...,2L \tag{2.7}$$

$$\overline{y} = \sum_{i=0}^{2L} \omega_i y_i \tag{2.8}$$

$$P_y = \sum_{i=0}^{2L} \omega_i (y_i - \bar{y})(y_i - \bar{y})^T \tag{2.9}$$

In comparison with the Monte-Carlo sampling and the EKF linearization technique in a two-dimensional example, the author in [57] used the Figure 2.1 to demonstrate the accuracy of the unscented transformation. The left side of the plot shows the Monte-Carlo sampling results. The middle part of the plot shows the results of the linearization of the EKF. And the right side shows the superior performance of the unscented transformation. Thus, the UKF is more suitable for non-linear application. Although the UKF is introduced to cope with non-linear problem, it also can be used in linear system [58]. When it comes to computational complexity, the UKF and the EKF have the same performance.

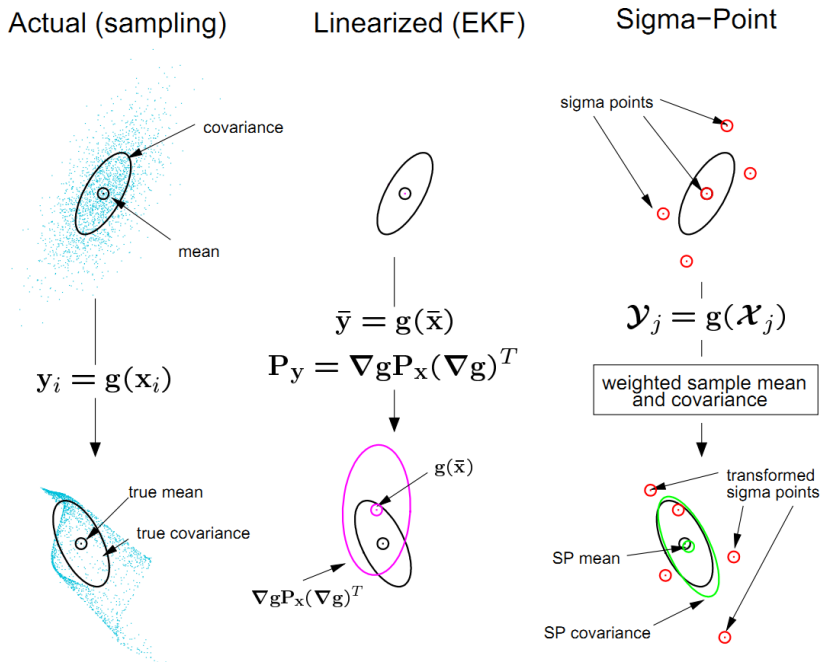


Figure 2.1: Comparison of the accuracy of the UT for mean and covariance with other two approaches [57].

2.3.4 Particle Filter

Thus far, all introduced filters, the KF, the EKF and UKF assume the associated noises follow the Gaussian distribution. What if the noise models do not follow the Gaussian distribution? And what if a filter is required to track more than one target simultaneously? Obviously a new filtering technique, particle filter has to be

introduced to cope with these problems. Particle filter is a popular optimal estimation solution for non-linear and non-Gaussian application. The main advantage of particle solutions over standard approximation methods, such as the EKF, is that they are not dependent on any functional linearization or approximation, but rely on a bunch of random particles with associated weights to represent the posterior distribution. Thus, particle filter can be used in more circumstances and with much flexibility. Due to this flexibility, the particle filter is computationally expensive. However, this constraint can be overcome eventually by the ever-increasing computational power and the application areas will be enlarged. Another bottleneck of the particle filter is the sample impoverishment or depletion of samples because of the sample resampling [59]. It is a step in the filtering process that is employed to duplicate particles with high importance weights and to get rid of the particles with low importance weights like wise. Although the resampling step was introduced to mitigate the influence of the particle degeneracy problem caused by the Sequential Importance Sampling (SIS), this side effect caused by the resampling will ruin the diversity of samples. Since the resampling step can lead to the particle depletion, myriads of the proposed algorithms are trying to handle this issue. The remainder of this subsection will briefly introduce the basic idea of the particle filter, aiming at helping readers to figure out the underlying ideas and concepts.

In a broad sense, particle filters belong to the class of Bayesian filters. They recursively perform the prediction of the state of systems according to the state space models and update through noisy measurement at each time step. In essence, particle filters are based on the sequential Monte Carlo methods, which employ mass samples to represent the probability densities [60]. The key concept of Monte Carlo methods is using average value of samples to substitute the integral calculation which exists in the Bayesian equations. The reason is that the integral calculation is more difficult. However, the target posterior distribution is also difficult to obtain. Thus, the Importance Sampling was introduced to solve this problem and for computational efficiency, the SIS is utilized in a real application. But the sampling step will lead to the degeneration of particles after several iterations, resulting in only a few particles with significant weights and the others with extremely low weights. If further steps are not employed, most computational efforts will be wasted on updating the non significant particles. A widely adopted solution to solve the degeneracy issue is resampling, such as the Sampling Importance Resampling (SIR). As mentioned before, this step will cause the sample impoverishment. For better and further understanding, a couple of papers or tutorials can be used as references [60, 61].

2.4 Summary

Since localization is the main topic of this thesis, this chapter has reviewed the mapping and localization related techniques. As noise is present in almost every state space and measurement, a better estimation performance relies on the filtering. Foundations of widely used filtering algorithms have been introduced as well. Besides, both the merit and the constraints of each filter have been analyzed.

"Sharp tools make good work."

Chinese Proverb

3 Sensor Set-up and Modeling

Sensors serve as eyes and ears to an autonomous vehicle, sensing its environments. A central computer running special programs acts as the brain of the vehicle, making different kinds of decisions and handling many situations. To fully understand how an autonomous vehicle works requires familiarity with its hardware set-up. To work with various types of sensors also requires the knowledge of their features. Thus, the hardware set-up and software framework of MadeInGermany will be depicted in detail in this chapter. Since this work needs to incorporate several sensors, this chapter will introduce these sensors thoroughly to some extent, especially focusing on their disadvantages or the application challenges.

3.1 Overall Sensor Set-up for MadeInGermany

Our present main test platform, "MadeInGermany", has a similar sensor set-up to many other well-known autonomous test vehicles [9]. As briefly mentioned in Chapter 1, section1.2.2, MadeInGermany is an autonomous vehicle transformed from a 2010 Volkswagen Passat Variant 3C. The transformation consists of installing the Drive-by-Wire system and equipping with many different types of sensors. Installing the Drive-by-Wire system is the first step to implement an autonomous vehicle, because it allows computer commands to control the vehicle. Manual operation, however, is not excluded. On the contrary, the safety driver can always take control of the vehicle with a higher priority out of safety considerations. Sensors include LIDAR sensors, Radar sensors, ultrasonic sensors and video cameras. Localization system can be treated as a special sensor as well. The rest of this section will introduce these sensors, respectively. Figure 3.1 illustrates the externally-visible part of

sensor configuration. And Figure 3.2 sketches its sensor coverage.

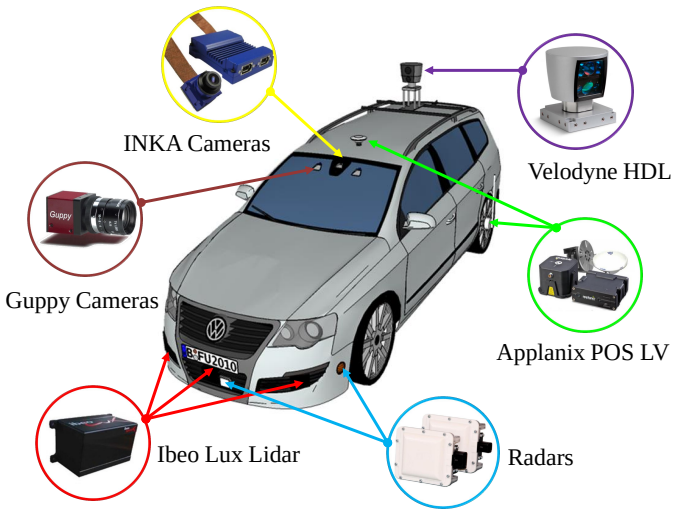


Figure 3.1: Sensor set-up of MadeInGermany[7]. Only the externally-visible part of sensors are illustrated.

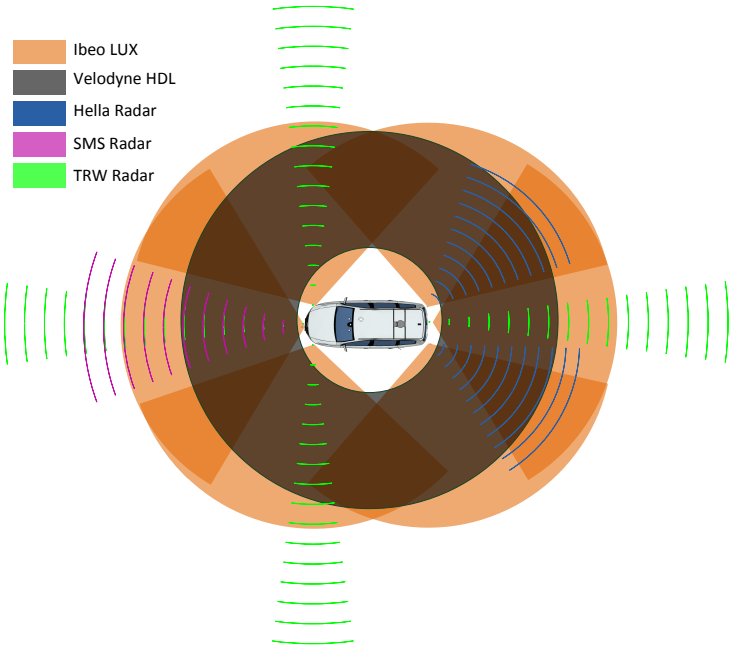


Figure 3.2: Illustration of sensor coverage of MadeInGermany[8].

[7]Inspired by [18].
[8]Inspired by [18].

1. **Control Interface**
   The Drive-by-Wire system is the key element of vehicle control interface, enabling the computer-driven control. It makes the software control over the vehicles possible, by transforming the traditional mechanically actuated throttle, gear shifter and steering system into electronically actuated ones. Thanks to its carmaker, Volkswagen, they factory-opened the Drive-by-Wire interface to us for the purpose of research. Thus, communications between computers and the vehicle's actuator systems can be established directly via Ethernet to the Controller Area Network (CAN) bus. For instance, steering commands are sent directly to the embedded controller system of the steering motor, which is a Maccon hollow shaft motor to manipulate the steering wheel [4]. The brake, however, must be always available for safety drivers because of safety reasons.

2. **Applanix POS LV**
   The estimation of the vehicle's position, orientation and other states is obtained from an Applanix™ POS-LV 510, which provides an inertial GPS navigation solution. This system operates at up to 200 Hz. It combines a position and orientation computing system, an Inertial Measurement Unit (IMU), a Distance Measuring Indicator (DMI) and differential GPS [32]. The IMU and DMI provide the relative position and orientation information while GPS generates absolute information. So the system takes advantage of these two localization schemes and generates stable and highly accurate position and orientation information. This integrated solution has uncomparable advantages over a stand-alone GPS solution, especially during the GPS outages, this scheme can still provide the inertial data alone. This allows the vehicle to safely pass through GPS outage area for a period of time. However, it also suffers from sudden position discontinuities upon switching between GPS coverage and outage [33], or between different GPS reference stations.

3. **Video Cameras**
   There are three kinds of cameras equipped behind the windshield, the Continental™ HDR camera, two monochrome Hella-Aglaia INKA™ stereo cameras and a pair of AVT (Allied Vision) Guppy™ cameras. The Continental HDR camera is an important part of the built-in production-line lane departure warning system [4]. It is employed to carry out robust lane detection. The Hella-Aglaia INKA 2 firewire COMS cameras with the back-end FPGA build up a powerful stereo vision system, generating a real time depth map

for object recognition and obstacle detection [18]. The Guppy cameras can be used in a wide range of application, such as lane detection, traffic signal detection and recognition, stereo vision and so on [4].

4. **LIDAR Sensors**
   LIDAR sensors have an advantage that can provide precise depth information regarding their surroundings, which is more accurate than that provided by stereo cameras. Thus, they are widely equipped in autonomous vehicles. Two types of LIDAR sensors is installed on the MadeInGermany, a Velodyne™ HDL-64E scanner and six Ibeo LUX™ laser scanners. The Velodyne HDL-64E is a very powerful spinning LIDAR sensor on board. It provides a 360° horizontal field of view and a 26.8° vertical field of view. It scans with 64 laser beams and spins at 5-15Hz. Further information about Velodyne will be discussed in next section. Six Ibeo LUX laser scanners, with three built in the front and three integrated in the rear, provide another almost 360° field of view around the car. And each of these scanners is a four-layer scanner with an 110° horizontal field of view and a 3.2° vertical field of view. It is conventionally used for obstacle detection and collision avoidance [62]. Obstacles can be detected up to 200 m by these scanners, which are required for highway tracks [18].

5. **Radar Sensors**
   Unlike LIDAR sensors, accurate distance is difficult to obtain for radar sensors. But the obstacle's speed relative to the vehicle can be easily determined by applying the Doppler effect. Thus, radars are normally used in auto-mobile for automatic cruise control and lane change assistance [18]. Three types of radars installed in our vehicle. The first one is the SMS short-range radar, which is used to merge obstacles detected by the laser scanner [18]. Second one is the Hella Blind-spot radar, which is used to assist lane change by observing the neighbor lanes [18]. The last one is the TRW long-range radar, which can detect obstacles up to 200 m. It is important for high speed driving on highways [18].

6. **Ultrasonic Sensors**
   Ultrasonic sensors are built-in sensors on board, with four in the front and another four in the rear. They are integrated for serving the Park Distance Control and the Park Assistant System. They observe objects within 2.5 m in front and back of the vehicle. They will be turned off automatically when the vehicle's speed exceeds 10 km/h and vice versa [18].

Sensors alone are not powerful enough, but definitely powerful with their respective computing systems. The reasons are two-fold. First, plugging all sensors with different interfaces into a single computer is not an easy task. Second, processing the huge amount of data with different update-rates in real-time is a great challenge even for a single current high-performance mobile computing platform. One Velodyne HDL alone generates over 1.3 million point cloud measurements per second, let alone other sensors. Thus, all the sensor data are processed by several computers on different networks and computing systems normally are installed in the trunk. Take Applanix POS LV as an example, its computing system acquires data from its GPS, IMU and DMI to generate high accuracy real-time vehicle navigation information. A high-performance workstation laptop acts as the "brain" of the vehicle for environment representation, behavioral decision and operational monitor.

3.2 Software Architecture

The software ran on MadeInGermany is quite complex. It is hard to imagine if there is no framework to manage the whole system. A framework would greatly benefit the entire project, such as reducing the programming effort. The software framework used in our project is the Open Robot Control Software (OROCOS) framework. OROCOS is an open source real-time robot control software package organized in the form of C++ libraries [63]. Especially, the OROCOS Real-Time Toolkit (RTT) can be used to create a module-based and data-flow centered control architecture without delving into the code of the whole system [4, 63]. In cooperation with CORBA (Common Object Request Broker Architecture), it can provide strict distributed real-time services for Linux systems [4].

Under this framework, software design obeys the concept of modular design. For instance, a sensor's operation is implemented as an abstract module. A specific task can also be modeled as a module. An XML file, called xmltest, is used to define the connection between modules. Then modules initiate communication via software ports, that is one module writes data into its output port and the connected module can read it from its input port. Thus, data can be exchanged in a lock-free and thread-safe state [4]. A module can be triggered periodically or by external events. Other properties can be configured through Attributes, Properties, and Operations without the need to compile the whole project [4].

Figure 3.3 depicts an abstract high-level schematic of the modules used in the architecture. The architecture can be divided into three levels, sensing, behavior and actuator. The first level involves in the environmental perception, related to

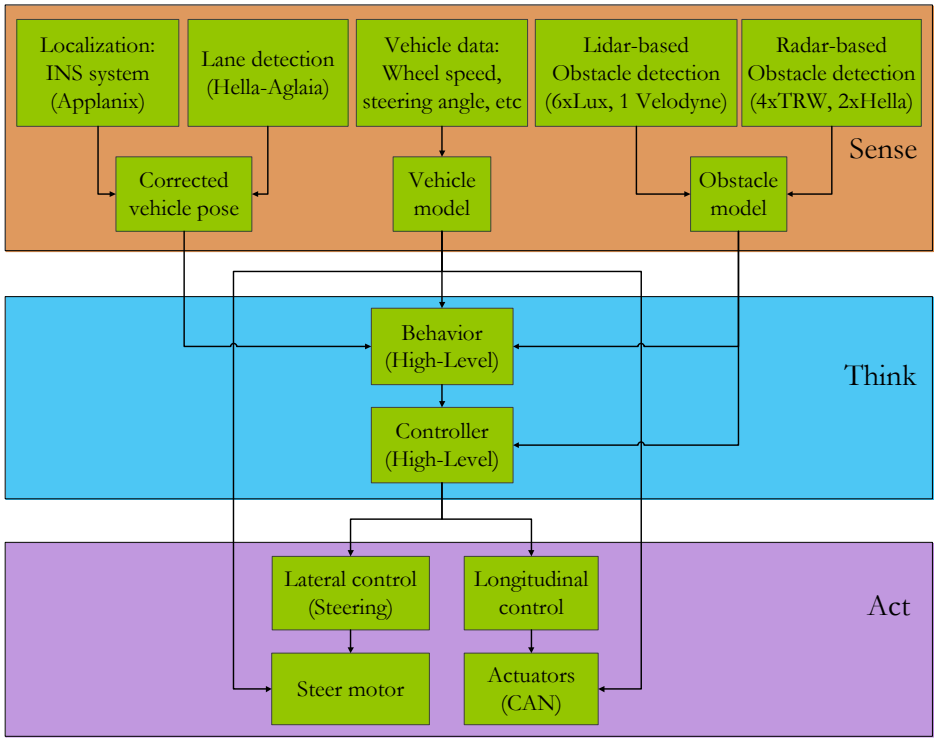


Figure 3.3: Abstract data flow chart of MadeInGermany [18].

cameras, LIDARs and radar sensors. The middle level acts as decision making role in dealing with different situations and translates high-level commands into low-level control commands to drive the actuators. The actuator level receives control commands to execute them. Since the work presented in this thesis is related to the localization problem, it belongs to the first level.

Although this work is still based on the OROCOS, few teams and developers have been maintaining the OROCOS project at present. And ROS (Robot Operating System) is a popular framework in robotic communities. Thus, ROS is currently employed to replace the old framework, OROCOS on our project.

3.3 Sensor Set-up for This Work

A brief introduction to the overall sensor set-up for MadeInGermany was given in previous section. The motivation of this work is to develop an effective localization scheme using fewer, cheaper or onboard sensors. Thus, this section will introduce the main sensors employed in this work, both advantages and disadvantages.

3.3.1 Velodyne for Feature Detection

For autonomous driving, the vehicle has to perceive its environment both correctly and reliably. Better environment perception capability gives the vehicle more confidence to perform more complex tasks like localization and obstacle avoidance. For feature detection, LIDAR sensors are the most reliable sensors among any existing sensors, as they are almost independent of weather and illumination conditions and have best directionality. They can be classified into two categories, 2D and 3D. Conventional 2D LIDAR sensors utilize a rotational mirror to reflect a single laser beam to scan the environment. This mechanism limits them to work in a single plane but they are still widely used in robotics. With multi-beam and also rotational mechanism, such LIDAR sensors are classified into 3D LIDAR sensors. One such example is the well-known Velodyne HDL-64E S2. Its prototype was introduced by the Velodyne TeamDAD in the 2005 DARPA Grand Challenge. And its mature product was adopted by five out of six teams that successfully completed the entire course of the 2007 DARPA Urban Challenge [64]. It will be further introduced in the rest of this section.

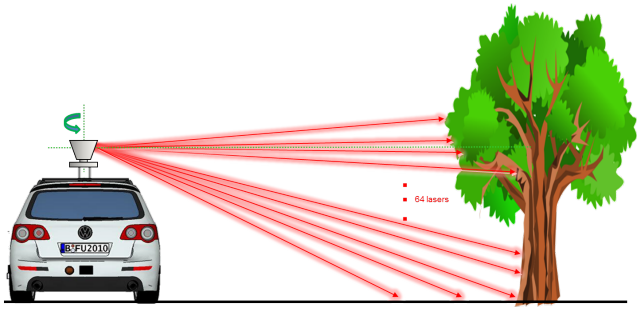


Figure 3.4: 64 vertically arranged lasers embedded inside the Velodyne HDL-64E.

Velodyne™ HDL-64E S2 comprises a vertical array of 64 laser emitters. They are well arranged in a spinning house, as shown in Figure 3.4, rotating at a user selectable rate ranging from 5 to 15 Hz. Change rotating rate does not change the data rate, but does change the horizontal angle resolution. The higher the rotation rate, the lower the angle resolution. It generates point cloud measurements (bearing, range, intensity) at a rate of over 1.3 million per second. And the effective measurement range can be up to 120 meters with an accuracy of less than 2 cm. The rotation feature gives a full 360-degree horizontal field of view. The well arranged 64 laser emitters are spread out over a 27-degree vertical field of view. Thus, combining with precise INS system, this sensor can be employed to create 3D maps of its environments and can be utilized to correct the vehicle's position. One competitive advantage over

other sensors is that the Velodyne LIDAR needs no ambient light to work, so it can robustly provide information for the vehicle at night.

Although the introduction of such novel devices in autonomous driving has dramatically changed the way in which robots perceive their environments, it also presents many challenges. First, like many other sensors, calibration is an indispensable step before any real application. A set of geometric parameters must be determined. They are defined to determine the relative position between each laser beam and the LIDAR sensor. Calibrating only one laser beam is not a daunting task but calibrating multi-beams simultaneously is an overwhelming task [64]. Second, processing large amount of data requires developing powerful algorithms and consumes huge computing power. Even though both sensors and research in computer vision area have made remarkable progress, sensors still cannot directly recognize features like we human beings. In this project, pose-like features should be extracted from the rich 3D point clouds. Finally, mounting such devices on mobile platforms also brings challenges to applications as an entire scan needs some time to complete, as illustrated in Figure 3.5. This is different from a stereo camera which can obtain a single image in one shot. And this issue will be further explained in section 4.3.3.

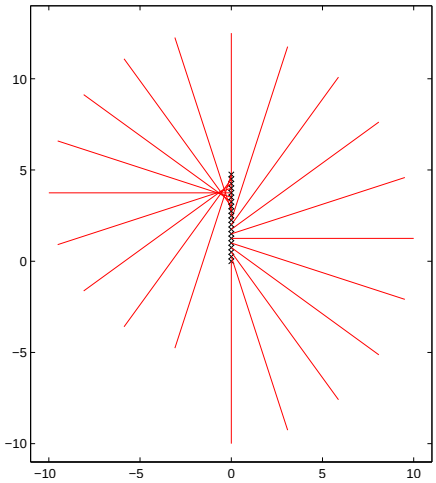


Figure 3.5: Illustration of one scan of Velodyne in the plan view. The red lines represent the laser beams while the black X crosses represent the Vehicle's positions. (For illustrative purposes only, no units involved.)

3.3.2 Odometry for Motion Prediction

Odometry is widely used in robotics by both legged and wheeled robots, which provides a simple and low-cost localization solution. Odometry estimates the relative robot displacement related to a starting position, by counting how many time

the wheel turned over time. Though this method is easy to implement, it suffers errors from the drift and slippage. Without feedback or other sensors for correction, the drift cannot be overcome and grows without bound over time [65]. Therefore, to be used effectively, other sensors and calibration processing are required to cooperate with odometry sensors. Technically speaking, an odometry sensor is a measurement sensor rather than a control sensor. But in practise, it is treated as a control unit for motion prediction in Kalman Filters [21].

Basically, three methods can be employed to do odometry calculation, the model-based, Visual Odometry and probabilistic approaches. The straightforward approach is the model-based calculation. The Ackermann steering model is doubtlessly the representative model [66]. This model exploits geometric relationships between the vehicle and the trajectory under the control law. A common simplified version of the Ackermann steering model is the bicycle model. It simplifies a four-wheel model into a two-wheel model and places an imaginary wheel pair at the centers of the front and rear axles [67]. The bicycle model also assumes that the vehicle moves on a plane. However, non-systematic errors will worsen the estimation results without using other sensors. Thus in this work, the author tries to fuse the wheel speed data with the gyroscope data to obtain a better estimation. Visual Odometry is another method to handle odometry calculation. This method determines the pose of a robot by finding relationships from sequential camera images. Stereo vision is commonly used to estimate the 3D position of features and track them between frames to evaluate displacement [68]. Visual Odometry can be employed to improve navigational accuracy. The last method is based on probabilistic approaches by using Kalman Filters. But tuning uncertainty covariance matrices is a difficult task. Thus it is difficult to improve the accuracy. An automatic tuning of noise parameters approach is proposed in [69].

Regarding this work, the vehicle has already installed the widely adopted Electronic Stability Program (ESP), also referred to as Electronic Stability Control. With the knowledge of ESP, we know ESP is a computerized system that assists the driver to control the vehicle in dangerous situations. To do that, it detects the vehicle's rotation rate and monitors the angle of steering wheel to predict the driver's intentions. Then it applies braking forces to individual wheels and adjusts the engine performance as well [70]. Wheel speed sensors are standard components in ESP, which detect how fast each individual wheel is turning. Thus, here they can be treated as odometric sensors. The derive of the motion model will be presented in the next chapter.

3.3.3 Gyroscope for Heading Estimation

The odometric sensors alone can be utilized to calculate the angle changes from motion equations. But it is difficult to increase the estimation accuracy due to external influences, such as road on uneven terrain, bump and slippage. Moreover, the accuracy of heading estimation influences the quality of Dead Reckoning (DR). The more accurate the heading estimation, the smaller the DR drift [71]. Therefore, to get more accurate heading estimation, we need to add a gyroscope to the system as a gyroscope is a specialized device to measure the angular changes, also known as angular velocity or yaw rate. After the well calibration, it will provide more accurate measurements. Integrating measurements over time will obtain the changes of the robot's heading. Since yaw rate sensor is an essential unit in ESP, extra gyroscope is not needed any more for not so sensitive applications. Therefore, the yaw rate sensor here acts as a gyroscope. Such sensors are more accurate than other counterpart sensors, such as odometric sensors, because it takes all external influences into consideration rather than its physical limits.

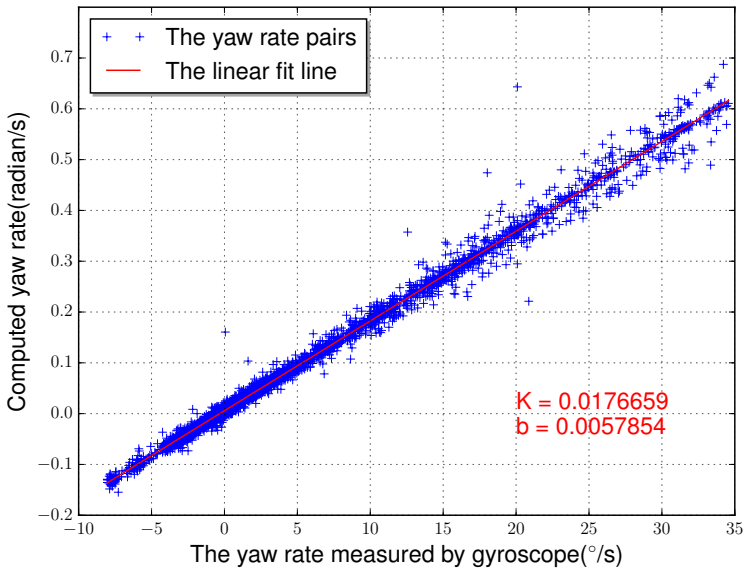


Figure 3.6: Scatter plot showing strong linear relationship.

Gyroscopes normally suffer from the influence of zero-rate level [72] (also known as drift). Zero-rate level is a term introduced to describe the phenomena that gyroscope still has reading even if there is no angular rate present, caused by temperature and other factors. Thus, it changes slightly over time and temperature as well. For most common applications, the impact can be ignored due to the factory tested and trimmed operation. So no further calibration is required. However, in order to

achieve best results, the yaw rate value still needs to be calibrated. The calibration is conducted off-line by comparing the calculated yaw rate with the reading of yaw rate sensor. The data related to those two parameters is extracted from the log files of test driving of MadeInGermany in Berlin. The calculated yaw rate is based on the ground truth heading provided by the Applanix system. To represent the correlation graphically, a scatter plot (3.6) was constructed. The variable on the horizontal axis represents the yaw rate value read from the gyroscope, while the variable in another axis represents the calculated yaw rate. It can be clearly seen that they present a strong linear correlation. Ideally, these two variables should be equal only with a fixed coefficient due to the difference between units, specifically speaking radian over degree. And the fixed coefficient should be equal to $\frac{\pi}{180}$ or (0.01745). It also presents the additive bias of yaw estimation, owing to the effect of zero-rate level. These two parameters can be determined through linear regression or random sample consensus (RANSAC).

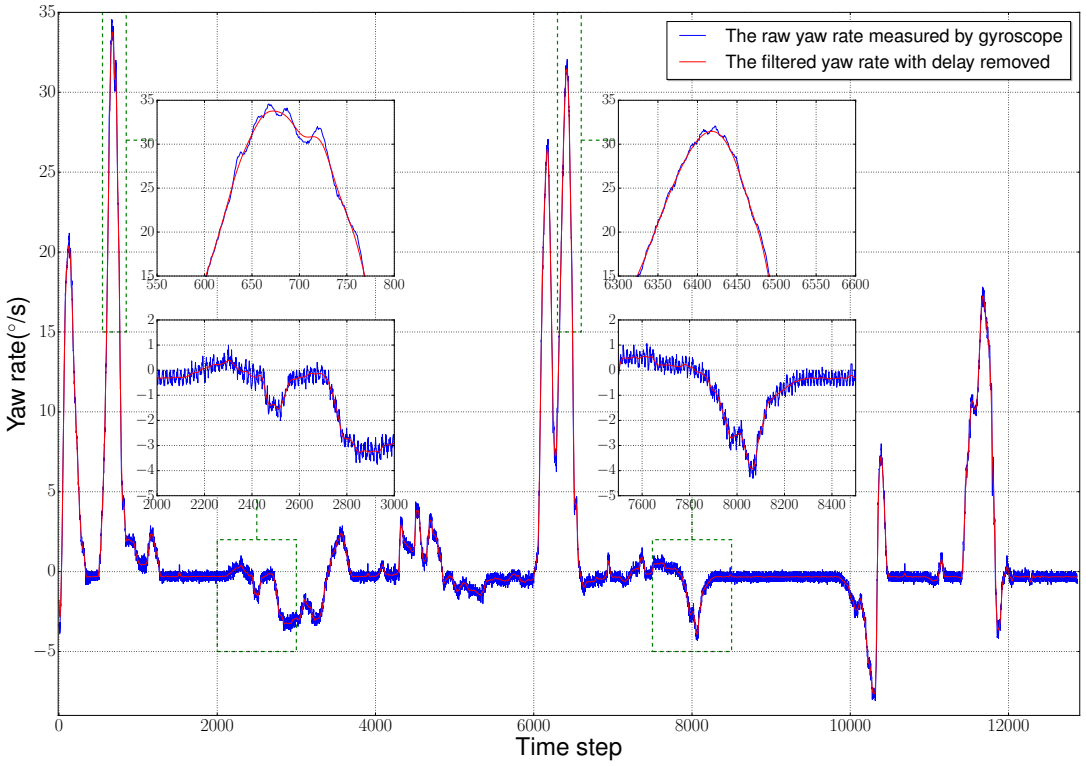


Figure 3.7: The raw data and filtered data with delay removed.

Considering the raw yaw rate data are polluted by noise, an FIR (Finite Impulse Response) filter is applied to the raw data, as Figure 3.7 shown. The blue line represents the raw data and the red line represents the processed data. From the magnified local plots, it can be clearly seen that the chosen filter has greatly smoothed the raw

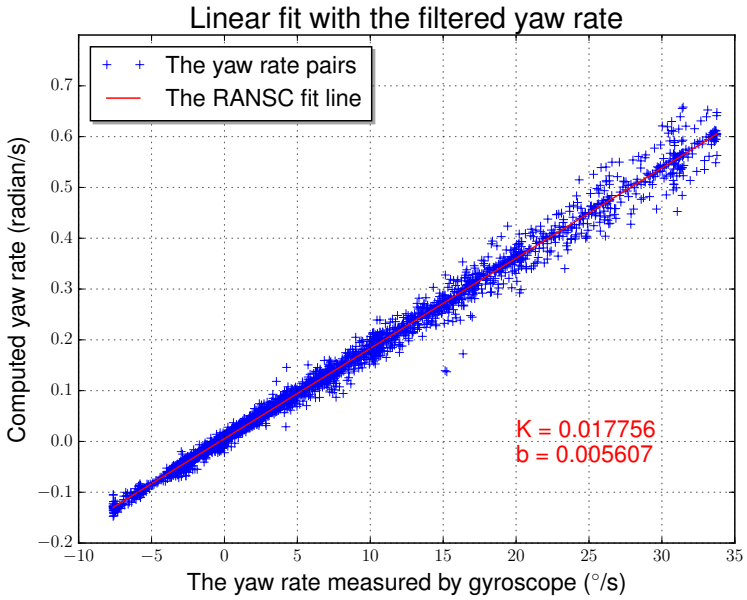


Figure 3.8: Scatter plot of the filtered yaw rate versus computed yaw rate.

data. Then we applied linear regression to the processed yaw rate and calculated yaw rate as shown in Figure 3.8. The scatter about the line is still big. The slope of the line is almost equal to the expected value, which confirms the correctness of the estimation. The biased value means the zero-rate level cannot be completely filtered. So the filtering solution does not dramatically improve the estimation result. And filtering introduces delay, which would worsen the estimation. Therefore, if better results are needed, better gyroscope should be installed.

Although the bias value cannot be filtered, it still can be calibrated online. The calibration can be carried out by averaging the reading of the gyroscope with absolute value when the vehicle is stationary. The typical value of the sensor bias for our test vehicle is around $0.33°/s$.

Besides, gyroscopes are also influenced by the lag problem. Readings of gyroscopes are normally delayed, lagging behind the current value. The influence of this phenomenon is significant, especially during a dramatic angular changing period, which leads to worse estimations. According to our knowledge, there is not much work in the community solving this problem.

3.3.4 Multi-Sensor Output Synchronization

As introduced in previous sections, multiple sensors are employed in this work. They have different output rates, as illustrated in the timing diagram 3.9. The Velodyne LIDAR rotates at constant frequency at 15 Hz. The yaw rate sensor runs

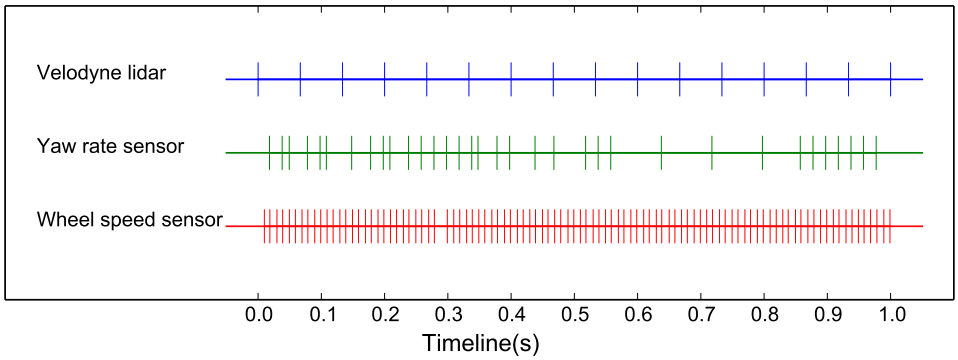


Figure 3.9: The multi-sensor output timing diagram in one second.

mainly on 50 Hz, while the wheel speed sensor almost at 100 Hz. Except the Velodyne LIDAR, the update rate of other two sensors does not run at consistent frequencies. And time is critical to the estimation of the both yaw angle and distance. Thus, the output of multi-sensor must be carefully aligned. In this work, the output port of vehicle speed from the on-board ESP sensor is chosen as the event trigger port. The main reason is that it can provide a 100 Hz update rate as higher update rate means much safer for driving. The output of other two sensors will be aligned to that of the speed sensor.

3.4 Summary

Sensors serve as eyes and ears to an autonomous vehicle, sensing its environments. To fully understand how an autonomous vehicle works, one needs to be familiar with its hardware set-up. To use a sensor properly needs to know its features. This chapter, therefore, thoroughly introduced the installed sensors to some extent, especially their disadvantages or the application challenges. It is a good preparation for the next chapter, as it provides basic information for deriving motion model used in the following chapter.

"Measure what can be measured, and make measurable what cannot be measured."

Galileo Galilei

4

Feature-Based Localization

For an autonomous vehicle, localization is an essential task of estimating the vehicle's pose relative to a given map. Maps are used to register the features' locations and other necessary information in coordinate systems. Maps, however, are independent of the vehicle's poses. Therefore, localization can be regarded as establishing correspondences between the coordinate of the map and the coordinate of the vehicle. Unfortunately, such correspondences cannot directly be measured by sensors. Thus, the estimated poses have to be inferred from the sensed data.

As introduced in the previous chapters, one motivation of this work is to employ low cost sensors to take the place of the expensive DGPS-based inertial navigation system, Applanix POS LV 510. In this work, data for constructing feature maps is still acquired by the high precision inertial navigation system and Velodyne LIDAR due to their relatively low uncertainties. Then a feature map based localization scheme is proposed by only using Velodyne LIDAR and other lower accuracy on-board sensors. The essence of the localization scheme is a novel two-point based localization method, which provides a simple and elegant solution. In this chapter, a localization solution based on the extended Kalman filter will be presented. The extended Kalman filter will be employed to fuse the odometric position estimation and measurements to the pole-like features extracted from Velodyne LIDAR point cloud. This chapter is organized as follows. First, several important coordinates will be defined. Then, localization related subjects will be introduced, including map construction, motion prediction and measurement update. Finally, two trajectory smoothing methods are proposed.

4.1 Coordinate Systems

Coordinate systems are of crucial importance for localizing the vehicle and presenting sensors' data. For this work, several coordinate systems are adopted. First for localizing the vehicle in a map, a global coordinate system should be either adopted or defined in the beginning. With a global coordinate system, features can be easily added to a map and vehicles' poses can be conveniently described with regard to the coordinate system. Besides, the vehicle and its carried sensors all have their own coordinate systems. Thus, this section will introduce the coordinate systems utilized in this work.

Four coordinate systems are adopted in this work, which include:

- the vehicle related coordinate system,
- the local East-North-Up (ENU) coordinate system,
- the Velodyne LIDAR related polar coordinate system,
- the World Geodetic System 1984 (also known as WGS84) coordinate system.

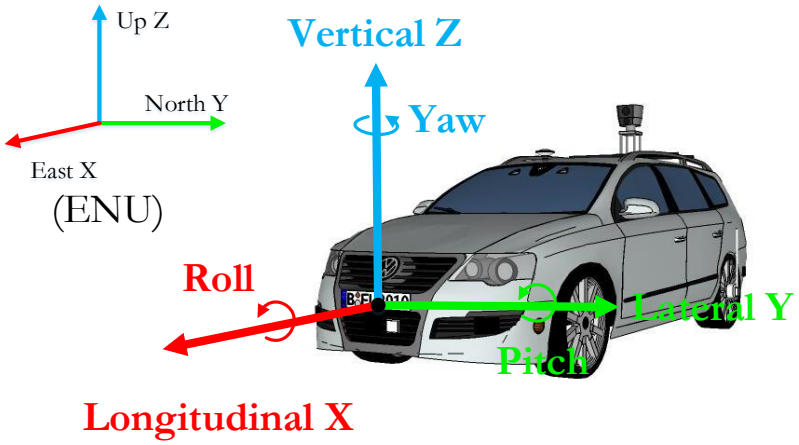


Figure 4.1: Vehicle related coordinate system. It is defined as a right hand orthogonal coordinate system. The origin of this coordinate system is placed at the center of the front axle of the vehicle. The x-axis is along the longitudinal axis of the vehicle, while y-axis is along its lateral axis.

Figure 4.1 illustrates the vehicle related coordinate system. The origin of this coordinate system is placed at the center of the front axle of the vehicle. It is also the reference point to the vehicle's position, which is critical for modeling the vehicle's

motion, as different points of the vehicle have different trajectories. Choosing different reference points, thus, leads to different motion models. The x-axis is along the longitudinal axis of the vehicle, while y-axis is along its lateral axis.

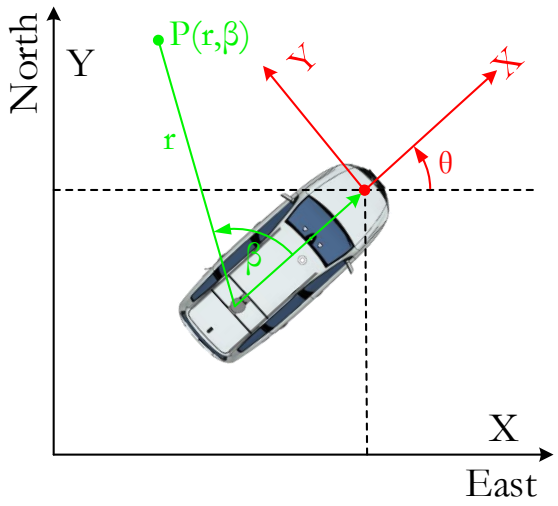


Figure 4.2: Top view of coordinate systems to demonstrate their relationships. The local ENU coordinate system is shown in black. The vehicle rated coordinate system is highlighted in red. And the Velodyne LIDAR ralated polar coordinate system is colored green. θ is defined as the yaw angle. (r, β) are the range and bearing to feature P, respectively.

For the localization problem addressed in this work, the vehicle's pose s_t in two-dimensional space (the Special Euclidean 2, or SE2 for short) will be considered. It consists of Cartesian position (x, y) and orientation θ related to the local ENU coordinate system, as illustrated in the top view in Figure 4.2. The X axis in the ENU coordinate points towards the true east and the Y axis points in the direction of the true north, which follow the conventional definition. The orientation θ is also known as the heading angle of a land vehicle or yaw angle. It is defined between the vehicle's x-axis and the X axis of the local ENU coordinate. Measurements of the features are carried out in the Velodyne LIDAR related polar coordinate system, as highlighted in green in the figure. Its origin is placed at the center of the Velodyne LIDAR. Its polar axis is along the longitudinal axis of the vehicle. Thus, (r, β) are the range and bearing to the feature P, respectively.

For realizing the global localization, the world geodetic system 1984 (a.k.a WGS84) is adopted. It can provide a single feature with a unique latitude and longitude position information in a global coordinate frame, which is useful during mapping stage

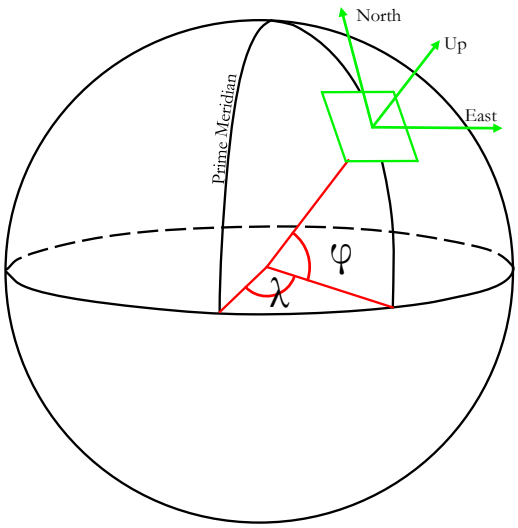


Figure 4.3: Relationship between the geodetic and a local ENU coordinate systems. φ and λ are represent latitude and longitude of the geodetic coordinate system, respectively. The ENU frame is colored green.

due to the unique property. However, latitude and longitude are not convenient for computation and querying maps. Therefore, for map matching and localization, a local ENU coordinate system is employed. The relationship between the geodetic and a local ENU coordinate system are illustrated in Figure 4.3. For the geodetic frame, φ and λ represent the latitude and longitude respectively to the local ENU frame which is colored green. However, the coordinate transformations among all these coordinate system are not presented here. For converting the WGS84 to a local ENU, please refer to [73] for more detailed introduction.

4.2 Overall System Structure

Figure 4.4 shows the localization module in the overall system structure. The whole system has three layers, the sensor layer, the localization layer and the framework layer. In the sensor layer, it includes a Velodyne HDL for pole-like feature detection, ESP related sensors for dead reckoning and for creating Ego state. The localization layer consists of two Extended Kalman filters, the localization EKF and the smoothing EKF. The localization EKF is utilized to fuse estimation from dead reckoning and measurements from pole-like feature extraction. It is the main unit of the localization module. As its name implied, the smoothing EKF takes the output of the localization EKF and dead reckoning as input to make the trajectory much

smoother in real time. To create an Ego state module that can be used by other modules in the system, the vehicle's speed and accelerations from ESP related sensors are also employed besides the output of the smoothing EKF. If treating the whole localization module as a stand-alone system, it is an alternative to the Applanix POS LV 510.

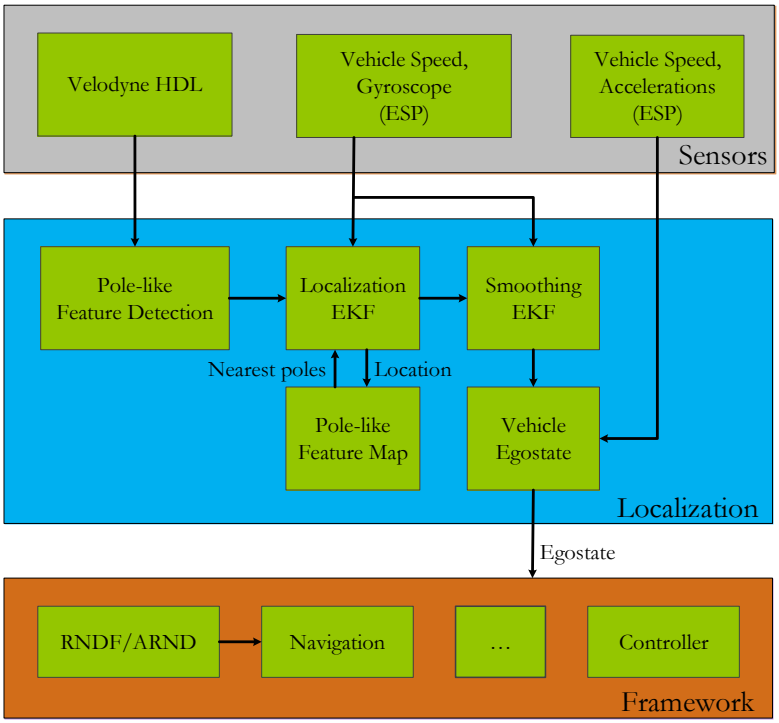


Figure 4.4: Localization module in the overall system structure. The localization module consists of two Extended Kalman Filters. The localization EKF is the main unit of estimation. The smoothing EKF is employed to smooth the estimated trajectory in real time. Then an Egostate is created to provide the vehicle state information to other modules.

4.3 Localization EKF

Localization is the task of estimating a robot's pose in a known or unknown environment. In a known environment, the pre-built map is available for localization which makes the task easier. In an unexplored environment, such as dangerous disaster sites, outer space planets, maps may not be easy to get previously. Then SLAM related techniques are required. Here, to simplify the localization problem and improve the performance, we assume that an *a priori* map is available for applications. Since the inherent uncertainties and bias are unavoidable for estimation,

it is of importance to have the knowledge of the uncertainties, in addition to estimation itself. Therefore, this problem is suitable to be solved in a probabilistic way. The whole estimation is under the frame of the EKF since the state and the measurement are non-linear. In the following subsections, several related aspects, for example map building, motion prediction and measurement update will be presented, respectively. Figure 4.5 illustrates the overview of the frame. First, an *a priori* map was created previously. Second, the readings of wheel speed sensors and gyroscope are used to calculate the motion prediction. Then, features are extracted from the Velodyne point cloud and matched with the features stored in the *a priori* map. After data association, the estimation is updated by fusing the measurement results.

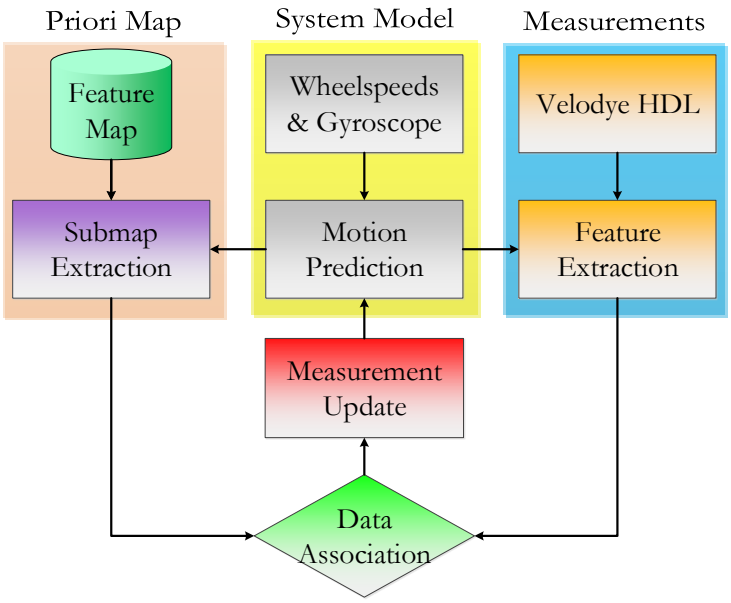


Figure 4.5: Overview of the localization EKF module. First, an *a priori* map was created previously, which named as Feature Map and illustrated as a green cylinder. Second, the readings of wheel speed sensors and gyroscope are used to compute the motion prediction. Then, features are extracted from the Velodyne point cloud and matched with the features stored in the *a priori* map. After data association, the estimation is updated by fusing the measurement results.

4.3.1 Map Building

As introduced in the previous chapter, several kinds of maps can be adopted to build the maps for this work. Here feature maps are chosen for convenience. For simplicity's sake, only pole-like features, such as tree trunks, lamp post and poles

of traffic signs, are chosen to build maps. The reason is that they are upright with homogeneous shape and can be easily identified in almost all environments. They are different from the road surface and lane markings which can be easily affected by many situations, ambient light, snow and fallen leaves. Of course, other features, such as corners and straight street curbs, can also be used. But pole-like features are more reliable and representative than other features because they normally remain in the same place for a longer time. Another reason why pole-like objects are chosen is that they can be regarded as point features in a two-dimensional map. For a single feature, much information, like the feature's size, position and LIDAR intensity, can be stored in the map and used during the feature matching stage. In this work, only the features' position and the associated uncertainties are utilized to build the map.

For the purposes of testing and debugging, two kinds of feature maps are created. One map is created through the log files of the test drive in Berlin, hereafter dubbed as the LOG map. The other one is created through the online data obtained from the FIS-Broker[1], hereafter dubbed as the FIS map. Basically, the LOG map is more accurate than the FIS map. But the FIS map can be created in a much easier way and on a large scale.

4.3.1.1 LOG Map

The LOG maps are created through the log files of the test drive by combining the vehicle's ego state information and the extracted features' information, as shown in Figure 4.6. The vehicle's ego state information was obtained from the Applanix POS LV and the features' information was extracted from the Velodyne point cloud. The feature extraction method and algorithm which were contributed by a former colleague [74], however, are beyond the scope of this work, so they will not be discussed here. To record the test logs, two testing sites were chosen in the vicinity of our lab. One is just in front of the main building of the campus (Thielallee in the western part of Berlin), which is often chosen by our group as the autonomous driving demonstration course. The street is covered with trees on road medians and roadsides. The other testing site is in Englerallee, which is a neighboring street near our lab. The test logs recorded at this site usually consist of several laps to test the repeatability.

After choosing the right map format and acquiring the test log files, maps can be constructed. To create a local map based on a test log file, the origin of the map should be set at first. To use this map in a large area or provide consistent localiza-

[1]FIS stands for "Fachübergreifendes InformationsSystem" in german, which means "Interdisciplinary Information system".

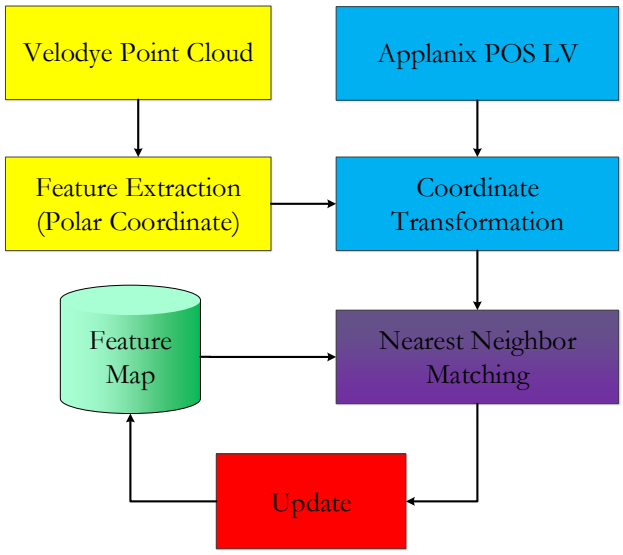


Figure 4.6: The mapping date flow diagram. Pole-like features are first extracted from the Velodyne point cloud and represented in polar coordinates. Combining the Applanix reported location information, the features can be expressed in a global coordinate. Then features are identified through the nearest neighbor matching. New features will be added to the map. Existing features will be used to update the map.

tion information, it is convenient to employ the WGS84 coordinates to denote its origin. Then all the features can be put in the map by adding their coordinates to the map. Due to the uncertainty problem, exact position for each feature is difficult to obtain. The same single feature can be observed by Velodyne LIDAR multiple times, when it locates within the operational range of Velodyne LIDAR. But the feature's position calculated at different moments is different. So the feature's position presents uncertainty. To get better results, the position will be estimated by averaging all the single estimation. Thus, the *a priori* map of the environment $\mathcal{M}$ can be defined as an origin of GPS coordinates in WGS84 and a set of features:

$$\mathcal{M} = \begin{bmatrix} O_{GPS} \\ [L_1,\ L_2,\ \ldots,\ Ln]^T \end{bmatrix}.$$

Each feature describes its coordinate position and measurements' uncertainties as:

$$L_n = [\overline{x_n},\quad \overline{y_n},\quad Var_n]^T$$

Since the number of times of a feature being detected is different from feature to feature, here we use simple iteration equations to update the positions in real time. In other words, the equations update the previous estimation whenever there is a new measurement coming. Although the basic equations are obvious and can be found on many textbooks, deriving their iteration equations here is just for ease of programming. Due to page restrictions, many intermediate steps are omitted. the equation (4.1) shows the position update as follows:

$$\begin{aligned} \overline{X_{n+1}} &= \frac{1}{n+1}\left(n\overline{X_x} + x_{n+1}\right) \\ &= \frac{n}{n+1}\overline{X_n} + \frac{1}{n+1}x_{n+1} \end{aligned} \tag{4.1}$$

where:

- x_{n+1}: is the new coming data for the $(n+1)th$ time
- $\overline{X_n}$: is the n times averaged position
- $\overline{X_{n+1}}$: is the $(n+1)$ times averaged position

The uncertainties are estimated, including variance and covariance as shown in following Equations form (4.2) to (4.5).

$$\begin{aligned} \sigma_n^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{X_n}\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(x_i^2 - 2x_i\overline{X_n} + \overline{X_n}^2\right) \\ &= \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \overline{X_n}^2 \end{aligned} \tag{4.2}$$

where:

- σ_n^2: is the variance through n times iteration
- $\overline{X_n}$: is the n times averaged position

$$\begin{aligned}
\sigma_{n+1}^2 &= \frac{1}{n+1}\sum_{i=1}^{n+1} x_i^2 - \overline{X_{n+1}}^2 \\
&= \frac{n}{n+1}\cdot\frac{1}{n}\sum_{i=1}^{n+1} x_i^2 - \left(\frac{n}{n+1}\overline{X_n} + \frac{1}{n+1}x_{n+1}\right)^2 \\
&= \frac{n}{n+1}\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \frac{1}{n}x_{n+1}^2\right) - \left(\frac{1}{n+1}\right)^2\left(n\overline{X_n} + x_{n+1}\right) \\
&= \cdots \\
&= \frac{n}{n+1}\sigma_n^2 - \frac{n}{(n+1)^2}\left(\overline{X_n} - x_{n+1}\right)^2
\end{aligned} \tag{4.3}$$

where:

- σ_n^2: is the variance through n times iteration
- σ_{n+1}^2: is the variance through $n+1$ times iteration
- x_{n+1}: is the new coming data for the $(n+1)th$ time
- $\overline{X_n}$: is the n times averaged position

$$\begin{aligned}
\mathrm{Cov}(X_n, Y_n) &= \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{X_n}\right)\left(y_i - \overline{Y_n}\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{X_n}\cdot\overline{Y_n}
\end{aligned} \tag{4.4}$$

where:

- $\mathrm{Cov}(X_n, Y_n)$: is the covariance between two variables X and Y
- $\overline{X_n}$: is the n times averaged value in X axis
- $\overline{Y_n}$: is the n times averaged value in Y axis

$$\begin{aligned}\mathrm{Cov}(X_{n+1}, Y_{n+1}) &= \frac{1}{n+1}\sum_{i=1}^{n+1} x_i y_i - \overline{X_{n+1}} \cdot \overline{Y_{n+1}} \\ &= \cdots \\ &= \frac{n}{n+1}\left(\mathrm{Cov}(X_n, Y_n) + \overline{X_n} \cdot \overline{Y_i}\right) + \frac{1}{n+1} x_{n+1} y_{n+1} - \overline{X_{n+1}} \cdot \overline{Y_{n+1}}\end{aligned} \tag{4.5}$$

where:

- $\mathrm{Cov}(X_{n+1}, Y_{n+1})$: is the covariance between two variables X and Y
- $\overline{X_{n+1}}$: is the $n+1$ times averaged value in X axis
- $\overline{Y_{n+1}}$: is the $n+1$ times averaged value in Y axis

4.3.1.2 FIS Map

Creating the FIS maps is another way to create feature maps by using the available online data. A huge set of data related to many aspects of the city of Berlin, including but not limited to nature, environment, traffic and so on, is published online via the FIS-Broker. As for our application, the position information of several pole-like features, such as trees and lamp posts lining Berlin's streets, can be used to create the feature map. First, the number of such features is pretty large. Street lining trees alone account for around 438,000 [75], that means there is a tree in every 12 meters on Berlin's streets. Besides, over 210,000 lamp posts present in the streets as well [76]. Second, the data is trustworthy as the data is managed by the Senate Department for Urban Development. We were told by one technical staff from the department that the typical position accuracy of a tree is from 50 to 100 centimeters, some areas even 10 to 15 cm. Although the accuracy is lower than the log file way, it is much easier and more convenient to create a city scale feature map. In contrast, it is extremely expensive to create the same scale map through the log file way. Thus, constructing maps in this way is quite promising.

To create maps, converting coordinates is a fundamental requirement as the features' positions are recorded in the EPSG 25833 coordinate. Despite the EPSG 25833 is a Cartesian coordinate, it is also a projected coordinate. A projected coordinate must bring distortions to the features' positions [77]. To convert such coordinate to the local coordinate, first need to convert EPSG 25833 to WGS84, then from WGS84 to ECEF, and finally from ECEF to ENU coordinate.

4.3.2 Motion Prediction

Odometric sensor and gyroscope are the main sensors for motion prediction, whose measurements can be used as the basis to calculate the vehicle's motion over time. This subsection will derive the motion model and its error model.

4.3.2.1 State Vector

Here an assumption that the vehicle drives in planar environments is made. The vehicle's pose (or state vector) is denoted as s_i, and as expressed in the equation (4.6). Its' position and orientation are further denoted as (x_i, y_i) and θ_i respectively, related to a local ENU coordinate frame g_o, as illustrated in Figure 4.7.

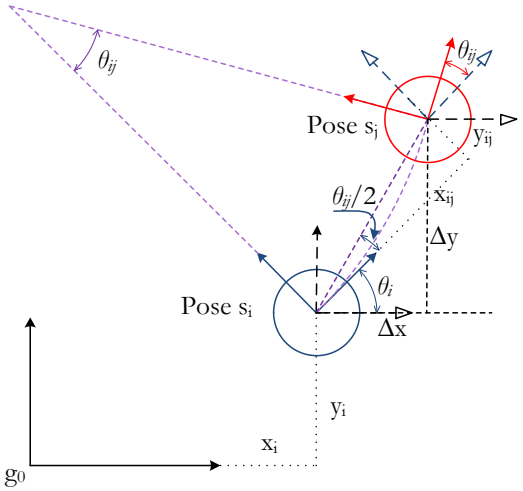


Figure 4.7: Geometry of the odometry process. From pose s_i to pose s_j, the vehicle drives in a circular trajectory. θ_{ij} is the yaw increment. x_{ij} and y_{ij} are the displacement in the local coordinate of pose s_i. The Δx and Δy are the displacement in the local ENU coordinate. Figure is inspired by [78].

$$s_i = [x_i,\ y_i,\ \theta_i]^T \tag{4.6}$$

The odometry is used for the relative localization, such that it cannot estimate the vehicle's pose directly, unlike GPS. It generally estimates the robot pose related to an initial pose by integrating the displacement over time. If the robot starts at pose i and moves to pose j the resulting local displacement measurement with respect to pose i is denoted as s_{ij}.

$$s_{ij} = [x_{ij},\ y_{ij},\ \theta_{ij}]^T$$

The elements in s_{ij} are random variables calculated from odometry by integrating wheel speeds $\Delta\vartheta$ and other dead reckoning related methods over time. The heading change θ_{ij} can be calculated by using the method introduced in the previous chapter. Here we assume the trajectory of the vehicle in this period of time closes to the circular path. The length of the arc or the distance traveled by the vehicle is calculated by averaging the distance covered by two wheels. And each distance is calculated by integrating the wheel speed over time. Here the length of the arc is denoted as $\Delta l = \Delta\vartheta \cdot \Delta_t$. The line segment from pose s_i to pose s_j is denoted as $\mathscr{D}$. Based on the knowledge of Euclidean geometry, the length of $\mathscr{D}$ can be calculated as the equation (4.7).

$$\mathscr{D} = 2\sin\left(\frac{\theta_{ij}}{2}\right) \cdot \frac{\Delta l}{\theta_{ij}} \quad \text{where } \theta_{ij} \text{ in radians} \tag{4.7}$$

Then x_{ij} and y_{ij} can be calculated through the trigonometric functions. as shown in the equation (4.8). The results show that the length of line segment is almost equal to that of the arc when the vehicle travels in a brief period of time.

$$\begin{cases} x_{ij} = \mathscr{D}\cos\left(\dfrac{\theta_{ij}}{2}\right) = 2\sin\left(\dfrac{\theta_{ij}}{2}\right) \cdot \dfrac{\Delta l}{\theta_{ij}} \cdot \cos\left(\dfrac{\theta_{ij}}{2}\right) \approx \Delta l \cdot \cos\left(\dfrac{\theta_{ij}}{2}\right) \\ y_{ij} = \mathscr{D}\sin\left(\dfrac{\theta_{ij}}{2}\right) = 2\sin^2\left(\dfrac{\theta_{ij}}{2}\right) \cdot \dfrac{\Delta l}{\theta_{ij}} \approx \Delta l \cdot \sin\left(\dfrac{\theta_{ij}}{2}\right) \end{cases} \tag{4.8}$$

where: if $\theta_{ij} \ll 0.0873$ radians (or 5°), the small-angle approximation is satisfied.

4.3.2.2 System Model

This part is going to derive the kinematics model which is utilized to predict how the state updates from one pose to another. Given an initial pose s_i in the ENU frame and a measured displacement s_{ij} in a frame local to s_i, the current pose estimate s_j can be calculated by integrating the control signal $u_i = \left(\Delta l, \theta_{ij}\right)^T$ in the ENU coordinate frame as the equation (4.9) shown. Although u_i represents incremental measurement which is computed from wheel speed sensors and gyroscope,

it is treated as control signals as presented in the previous chapter.

$$
\begin{aligned}
f\left(x_j, y_j, \theta_j\right) &= f\left(\left(x_i, y_i, \theta_i\right), u_i\right) \\
s_j = \begin{bmatrix} x_j \\ y_j \\ \theta_j \end{bmatrix} &= \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \theta_{ij} \end{bmatrix} \\
&= \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} \Delta l \cdot \cos\left(\theta_i + \dfrac{\theta_{ij}}{2}\right) \\ \Delta l \cdot \sin\left(\theta_i + \dfrac{\theta_{ij}}{2}\right) \\ \theta_{ij} \end{bmatrix} \\
&= \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ij} \\ y_{ij} \\ \theta_{ij} \end{bmatrix}
\end{aligned}
\tag{4.9}
$$

For some applications, the relative displacement s_{ij} between two given poses s_i and s_j in the ENU frame can be solved as shown in the equation (4.10).

$$
s_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ \theta_{ij} \end{bmatrix} = \begin{bmatrix} \cos(-\theta_i) & -\sin(-\theta_i) & 0 \\ \sin(-\theta_i) & \cos(-\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j - x_i \\ y_j - y_i \\ \theta_j - \theta_i \end{bmatrix} \tag{4.10}
$$

The pose covariance is a fundamental parameter in the EKF. Thus, obtaining this parameter is required. First, recalling the KF equation (2.2) in chapter 2, P_j is the predicted covariance and A_j is the state transition matrix. For the Extended Kalman Filter, A_j is the Jacobian matrix of partial derivations of process update function with respect to the state vector. This equation will be used in later and here detailed coefficient will be presented below (4.11).

$$
P_j^- = A_j P_i A_j^T + Q_j
$$

where:

$$
\begin{aligned}
A_j = \frac{\partial s_j}{\partial s_i} &= \begin{bmatrix} \frac{\partial x_j}{\partial x_i} & \frac{\partial x_j}{\partial y_i} & \frac{\partial x_j}{\partial \theta_i} \\ \frac{\partial y_j}{\partial x_i} & \frac{\partial y_j}{\partial y_i} & \frac{\partial y_j}{\partial \theta_i} \\ \frac{\partial \theta_j}{\partial x_i} & \frac{\partial \theta_j}{\partial y_i} & \frac{\partial \theta_j}{\partial \theta_i} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & -\Delta l \sin\left(\theta_i + \frac{\theta_{ij}}{2}\right) \\ 0 & 1 & \Delta l \cos\left(\theta_i + \frac{\theta_{ij}}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & -x_{ij}\sin(\theta_i) - y_{ij}\cos(\theta_i) \\ 0 & 1 & x_{ij}\cos(\theta_i) - y_{ij}\sin(\theta_i) \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{4.11}
$$

The input Jacobian B_j is the Jacobian matrix of partial derivations of process update function s_j with respect to the input vector u_i, as represented in the equation (4.12).

$$
\begin{aligned}
B_j = \frac{\partial s_j}{\partial u_i} &= \begin{bmatrix} \frac{\partial x_j}{\partial \Delta l} & \frac{\partial x_j}{\partial \theta_{ij}} \\ \frac{\partial y_j}{\partial \Delta l} & \frac{\partial y_j}{\partial \theta_{ij}} \\ \frac{\partial \theta_j}{\partial \Delta l} & \frac{\partial \theta_j}{\partial \theta_{ij}} \end{bmatrix} \\
&= \begin{bmatrix} \cos\left(\theta_i + \frac{\theta_{ij}}{2}\right) & -\frac{1}{2}\Delta l \sin\left(\theta_i + \frac{\theta_{ij}}{2}\right) \\ \sin\left(\theta_i + \frac{\theta_{ij}}{2}\right) & \frac{1}{2}\Delta l \cos\left(\theta_i + \frac{\theta_{ij}}{2}\right) \\ 0 & 1 \end{bmatrix}
\end{aligned}
\tag{4.12}
$$

4.3.2.3 Noise Model

The EKF framework assumes that the process noise follows a Gaussian distribution. The general form of its covariance matrix can be defined as

$$Q_j = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{x\theta}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{y\theta}^2 \\ \sigma_{x\theta}^2 & \sigma_{y\theta}^2 & \sigma_\theta^2 \end{bmatrix} \tag{4.13}$$

where σ_x^2, σ_y^2 and σ_θ^2 represent the variance in x, y and θ, respectively. Other elements represent the covariances between any two of the three variables. σ_x^2 also represents the estimation uncertainty in the lateral direction, while σ_y^2 represents the uncertainty in the longitudinal direction. The actual elements of the covariance matrix depend on the system model. In practice, a simple model assumes that x, y and θ are mutually independent for a tiny displacement. That is to say, all the covariances are set to zero for simplicity.

4.3.3 Data Association

Data association is one of the critical steps for localization. To some extent, it determines the performance of localization. Technically speaking, data association is the task of establishing the correspondences between sensors' measurements and features in a map. Given any new measurements, data association involves determining the correct matches to an existing part of the map. It is universally acknowledged that data association is one of the hardest and most important problems in localization [79]. If correct correspondences cannot be established, then the sensors' measurement cannot provide any useful information for localization. Wrong matches may even distort the estimation.

There are a couple of strategies to deal with data association problem, such as *the maximum likelihood*, *the nearest neighbor*. The maximum likelihood correspondence uses the probabilistic method to determine the most likely value of the correspondence variable, and then takes this value as the best match [21]. What if there is more than one equally likely hypothesis for the correspondence variable? Then it turns out to be not so effective. However, this case can be avoided by designing the system in two techniques. First, make sure that the selected features are sufficiently distinct and far away from each other. Second, ensure that the uncertainty

of the robot's pose remains as small as possible. These two strategies are somehow in conflict with each other. Nevertheless, maximum likelihood techniques are still widely adopted in practice [21]. The straightforward technique is the nearest neighbor correspondence. This classical approach considers each matching between sensor measurements and features in a given map independently. And it does not take the correlated errors of measurement prediction into consideration. So it increases the possibility of wrong matchings [80]. In essence, it is based on the estimated vehicle pose to find the correspondence. Its performance, therefore, is determined by the quality of the pose estimation. A brief analysis of the influence of the heading is shown as follows.

- let θ be the ground truth heading, given in radians;
- let θ_e be the angle uncertainty, and its absolute value is less than 0.09 in radians;
- let r be the range from the vehicle to the pose, and assume it is quite accurate.

$$\begin{aligned} r_{error} &= r\sqrt{(\cos\theta - \cos(\theta+\theta_e))^2 + (\sin\theta - \sin(\theta+\theta_e))^2} \\ &= r\sqrt{2-2cos\theta_e} \\ &= 2r\left|sin\left(\frac{\theta_e}{2}\right)\right| \\ &= r\,|\theta_e| \end{aligned} \tag{4.14}$$

So the related error can be expressed in the equation (4.14). It is proportional to $|\theta_e|$ with a given distance. Reducing the uncertainty in heading will increase the accuracy. In this work, the nearest neighbor approach is also used for the primary map matching step. If the threshold for accepting the matching uncertainty is set to 0.7 m, the real matching error is up to 1.2 m due to the uncertainty of the vehicle's position and heading, as shown in Figure 4.8. Here, the nearest neighbor matching error is defined as the distance from the feature's estimated position to its matched feature's position. And the real matching error is defined as the distance from the feature's real position to its matched feature's position. If a real matching error is bigger than 0.7 m, it is probably a wrong matching. So this method may introduce many wrong matches. Without further operation, these wrong matches will seriously influence the localization accuracy. However, this is just the first step for matching and the wrong matches are sometimes unavoidable. For robust matching and obtaining better performance, a two-point scheme is employed. The two-point approach will

further reject most spurious matches introduced in the nearest neighbor matching step.

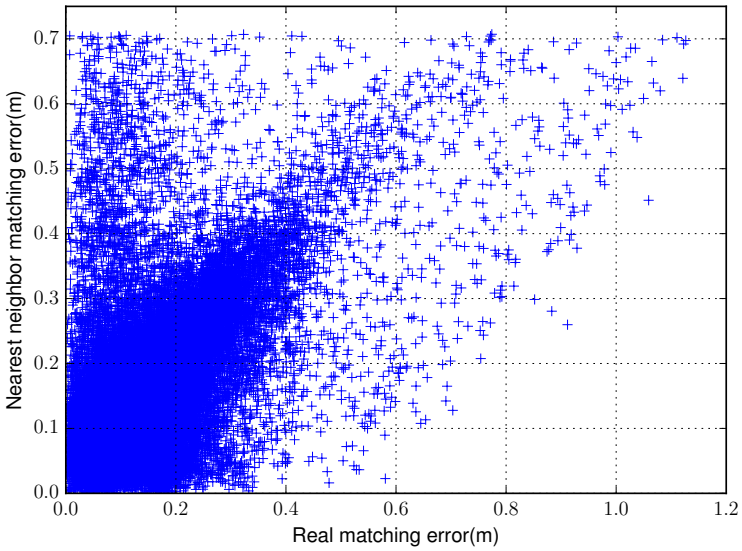


Figure 4.8: The performance of the nearest neighbor matching.

4.3.3.1 Pose Compensation

Since localization needs to estimate both the vehicle's Cartesian position and heading, at least two features are required to determine these parameters in a planar environment. For robust applications, a minimum of three features are needed. The operational method of Velodyne LIDAR also poses a challenge to localization. The main reason is that it cannot detect all the features around itself simultaneously at the same place when its carrier vehicle moves. Therefore, features are detected at different moments and at different poses of the vehicle, as illustrated in Figure 4.9. Consequently, if we use two detected features' coordinates as the centers of two circles and the respective ranges as the radius of circles to plot two circles, these two circles will not intersect at one point which lies on the trajectory of the vehicle. In order to utilize two or more features to locate the vehicle in one position, a little compensation work is needed here by applying dead reckoning in a brief period of time.

In the software side, the localization program will be triggered to update by the trigger event. Here we choose the event of the speed reporting as the trigger event because of its around 100 Hz update rate. Higher update rate means safer driving. During the interval of two consecutive trigger moments Δt, the change in yaw angle, $\Delta\theta$ is calculated by integrating yaw rate over time. And the displacement, Δd

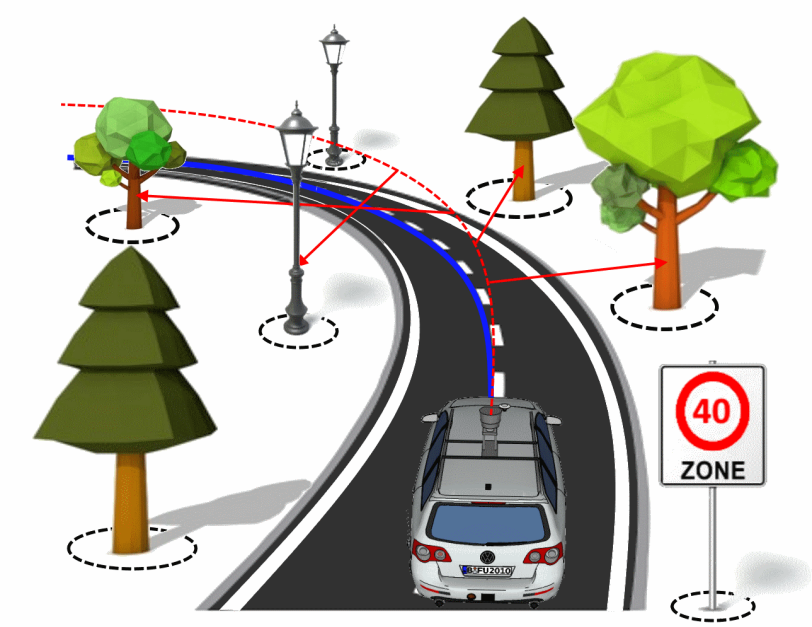


Figure 4.9: Illustration of feature detection by Velodyne LIDAR[9].

is calculated by integrating the average reading of the wheel speed sensors over time. Then to compensate for the positions of those features and the bearings which are angles from the vehicle to poles, the equation (4.15) and (4.16) were used. After compensation, it seems that all detected features were detected at the same time when the triggering event arrives, as illustrate in Figure 4.10.

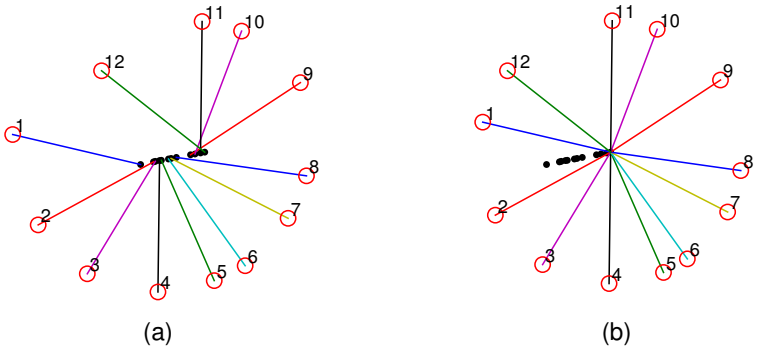


Figure 4.10: Pose compensation. (a) illustrates the LIDAR distortion problem and (b) represents the pose compensation. Here red circles with numbers represent poles, while the black points represent vehicle's positions. The colorful line segments indicate the LIDAR beams.

- let T_i be the instant when the i^{th} pole was detected;

[9]Picture courtesy: Ricardo Carrillo.

- let T_o be the last triggering moment;
- let θ_i be the bearing from the vehicle to the i^{th} pole;
- let r_i be the range from the vehicle to the i^{th} pole;
- let (x_i, y_i) be the coordinates of the i^{th} matched feature in the map.

Now it's time to calculate the percentage of time in the interval Δt, that the i^{th} feature took. Let the coefficient of the percentage be denoted as p_i.

$$p_i = \frac{T_i - T_o}{\Delta t} \tag{4.15}$$

Then the compensated coordinates and bearing would be expressed in the following equation (4.16). And the range values will remain the same.

$$\begin{cases} x_i^{'} = x_i + (1-p_i)\cdot\Delta d[x] \\ y_i^{'} = y_i + (1-p_i)\cdot\Delta d[y] \\ \theta_i^{'} = \theta_i - (1-p_i)\cdot\Delta\theta \end{cases} \tag{4.16}$$

4.3.3.2 Two-Point Matching

After applying pose compensation to both observed features and matched features in the map, finding the correspondence between these two sets of points can be processed. In this part, the two-point matching approach is proposed. Let us assume the vehicle's position is $O(x_o, y_o)$ and its orientation is θ_o. Two matched features' coordinates in the *a priori* map are denoted by $P(x_1, y_1)$ and $Q(x_2, y_2)$, respectively. As introduced before, we need to compensate the features' position such that we can assume these features are detected simultaneously. After compensation, they can be represented as $P^{'}(x_1^{'}, y_1^{'})$ and $Q^{'}(x_2^{'}, y_2^{'})$, respectively. And the measurements to these two features consist of the ranges and bearings. Let $(r_1, \theta_1^{'})$ and $(r_2, \theta_2^{'})$ be the range and bearing pair to the feature $P^{'}$ and $Q^{'}$, respectively. With all these notations, the two-point matching approach can be derived through the following equations.

$$\begin{cases} x_i^{'} = x_o + r_i\cos\left(\theta_o + \theta_i^{'}\right) \\ y_i^{'} = y_o + r_i\sin\left(\theta_o + \theta_i^{'}\right) \end{cases} \quad \text{where } i = 1, 2, \cdots, n \tag{4.17}$$

If subtract the equation (4.17) when $i = 1, 2$, another equation will be got as

follows:

$$\begin{cases} x_1^{'} - x_2^{'} = & r_1 \cos\left(\theta_o + \theta_1^{'}\right) - r_2 \cos\left(\theta_o + \theta_2^{'}\right) \\ y_1^{'} - y_2^{'} = & r_1 \sin\left(\theta_o + \theta_1^{'}\right) - r_2 \sin\left(\theta_o + \theta_2^{'}\right) \end{cases} \tag{4.18}$$

$$\begin{cases} x_1^{'} - x_2^{'} = & \cos\theta_o \left(r_1 \cos\theta_1^{'} - r_2 \cos\theta_2^{'}\right) - \sin\theta_o \left(r_1 \sin\theta_1^{'} - r_2 \sin\theta_2^{'}\right) \\ y_1^{'} - y_2^{'} = & \cos\theta_o \left(r_1 \sin\theta_1^{'} - r_2 \sin\theta_2^{'}\right) + \sin\theta_o \left(r_1 \cos\theta_1^{'} - r_2 \cos\theta_2^{'}\right) \end{cases}$$

let

$$\begin{cases} a = & r_1 \cos\theta_1^{'} - r_2 \cos\theta_2^{'} \\ b = & r_1 \sin\theta_1^{'} - r_2 \sin\theta_2^{'} \end{cases} \tag{4.19}$$

Then

$$\begin{cases} \Delta x = & x_1^{'} - x_2^{'} = & a\cos\theta_o - b\sin\theta_o \\ \Delta y = & y_1^{'} - y_2^{'} = & b\cos\theta_o + a\sin\theta_o \end{cases}$$

$$\begin{cases} \cos\theta_o = & \dfrac{a\Delta x + b\Delta y}{a^2 + b^2} \\ \sin\theta_o = & \dfrac{a\Delta y - b\Delta x}{a^2 + b^2} \end{cases} \tag{4.20}$$

Thus

$$\begin{aligned} \theta_o &= \text{atan2}(\sin\theta_o, \cos\theta_o) \\ &= \text{atan2}((a\Delta y - b\Delta x), (a\Delta x + b\Delta y)) \end{aligned} \tag{4.21}$$

From the equation (4.21), the vehicle's estimated yaw angle can be obtained. This estimated value, however, should be compared with the value estimated in the motion prediction step. The reason for this comparison is two-fold. On the one hand, the estimated yaw angle in the motion predict step is close to the true value though with tiny error; on the other hand, the nearest neighbor matching is prone to introducing many wrong matches and through the two-point matching examination, the wrong matches can be easily identified. Thus, after the two-point matching, the spurious matches can be rejected and better results can be anticipated. After the yaw angle estimation, the positional information can be obtained by the equation (4.17).

4.3.3.3 Trilateration

After applying the pose compensation to the matched features, we can use the trilateration algorithm to estimate the vehicle's position. The trilateration algorithm is the fundamental algorithm used in the GPS solution. The advantage of this approach is that it does not require any angle measurements. Knowing a set of coordinates of reference points and the distance measurements to them, the unknown position can be determined by trilateration calculations. The minimum number of the reference points are three for 2D positioning or four for 3D. The aim of this approach is to find the intersection of related circumferences in the scenario of 2D positioning or spheres in 3D. And their centers are the coordinates of these reference points. The point of the intersection is calculated by solving several non-linear equations simultaneously. The solution to this problem, however, is not feasible because of producing a high-degree non-linear equation [81]. Murphy et. al in [81] thoroughly studied this problem and presented several solutions, including linearized equations, linear least squares method and non-linear least squares. Although it seems that this algorithm is quite promising, this work does not use it as the trilateration algorithm does not provide more accurate results and it is prone to failure under some circumstances.

4.3.3.4 Point Pattern Matching

If the pose estimation presents a big uncertainty, the *Nearest Neighbor* approach may fail to solve the crucial data association problem. Then the estimation cannot obtain measurement update and its uncertainty will increase without bound. Under such circumstance, a more robust matching approach should be employed. Point pattern matching, or point set registration, is an ideal approach to solve this problem. Several techniques can be used to solve the point pattern matching problem. These techniques include the geometric hashing approach [82], the Iterative Closest Point (ICP) approach [83], the triangulation [43] and so on. Compared with the *nearest neighbor* approach, these methods are computational expensive. For a given iterative times, the ICP approach may not be able to give an accurate solution. In this work, the execution time for solving the point pattern matching problem should be less than 0.01 seconds. Thus, these methods are difficult to apply to this work. However, they can be used to find the initial estimate by recovering the correspondence between the map and the measurement.

4.3.4 Measurement Update

The measurement stage uses the measurement data to update the estimation. Normally, the measurement function is represented in the equation (4.22). Further, it can be represented in the equation (4.23). And this equation is widely adopted by numerous papers in robotics [21, 84]. Here Z_j^k denotes the measurements to the k^{th} landmark. The variable items r_j^k and θ_j^k denote the range and bearing to this landmark at the state s_j, respectively. And (x_j^k, y_j^k) are the coordinates of the k^{th} landmark.

$$Z_j = \mathrm{h}(s_j) + \delta_j \tag{4.22}$$

$$\begin{aligned} Z_j^k &= \begin{pmatrix} r_j^k \\ \\ \theta_j^k \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{\left(x_j^k - x_j\right)^2 + \left(y_j^k - y_j\right)^2} \\ \\ atan2\left(\left(y_j^k - y_j\right), \left(x_j^k - x_j\right)\right) - \theta_i \end{pmatrix} \end{aligned} \tag{4.23}$$

However, such equation is set up, provided that the critical data association is known, which is not always true in practice. For instance, in this work the wrong matching is unavoidable. Therefore, we used two-point matching to solve the data association in this thesis. Since the data association step obtained the estimation directly, the measurement function is linear. Thus, the computational complexity for measurement update stage dramatically reduced.

4.4 Trajectory Smoothing

Due to the presence of the unavoidable uncertainties in maps and the measurement update, the output of the localization EKF oscillates, even taking less correction from the observation, like 2 percent. As a consequence, the trajectory looks like waves especially during making U-turns. And the steering wheel turns much frequently, which makes the driving experience not so pleasant. Under such circumstance, some smoothing techniques are employed. A second smoothing EKF and the smoothing method based on Ackermann constraint are tested in this work.

4.4.1 Smoothing EKF

As its name implied, the smoothing EKF is utilized to smooth the trajectory. This filter uses the odometry to carry out the motion prediction and the output of the localization filter to conduct the measurement update. And it gives higher weight to the odometry. Then it smooths the trajectory. The performance is shown in the evaluation chapter. However, our intuition may tell us that one filter is enough, since two filters have the same structure. So why combining two extended Kalman filters has better performance? Mathematical analysis is shown as follows. First, we use the equation (4.24) and (4.25) to model the system and measurement equations. Since they have the same output, the measurement transition matrix for the smoothing EKF is an identity matrix. Then the relationship between these two filters can be expressed in (4.29). It can be seen that the second filter is unnecessary if and only if K_2 is an identity matrix.

$$X_t^i = F_t X_{t-1}^i + B_t u_t \tag{4.24}$$

where:

- X_t^i: is the state vector at time t.
- $i = 1,2$: is used to differentiate the first and second filter.
- F_t: is the state transition matrix.
- B_t: is the control input matrix
- u_t: is the control vector at time t

$$Z_t = H_t X_t \tag{4.25}$$

where:

- Z_t: is the measurement vector at time t.
- H_t: is the measurement transition matrix.

$$X_t^{1-} = F_t X_{t-1}^1 + B_t u_t \tag{4.26}$$

$$X_t^1 = X_t^{1-} + K_1(Z_t - H_t X_t^{1-}) \tag{4.27}$$

$$X_t^{2-} = F_t X_{t-1}^2 + B_t u_t \tag{4.28}$$

$$X_t^2 = X_t^{2-} + K_2(X_t^1 - X_t^{2-}) \tag{4.29}$$

4.4.2 Ackermann constraint

The concept behind the Ackermann constraint is that the movement of any point is constrained by the Ackermann steering law. And its position cannot swing due to the mechanical and geometrical constraints. Its position can be calculated based on the geometrical constraint. Given the position of the rear wheel center in the present and last instant, an angle can be computed. Ideally, this computed angle should be equal to the averaged yaw angle at these two moments, as shown in Figure 4.11. If the difference between these two angles bigger than a given empirical threshold, the estimated position in the center of the front wheels must be inaccurate to some extent. Then correcting the position of the rear wheel center can achieve a better estimation. Algorithm 1 describes the whole procedure in pseudo-code.

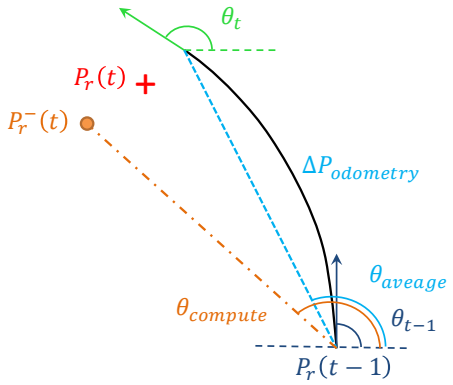


Figure 4.11: Yaw angle relationship between two consecutive instants.

Data: $P_f^-(t)$, vehicle pose estimate from the localization EKF
Result: $P_f(t)$, corrected pose estimate

while do
  $P_r^-(t) = computeRearPose(P_f^-(t))$;
  $\Delta P_r(t) = P_r^-(t) - P_r(t-1)$;
  $\theta_{compute} = atan2(\Delta P_r(t)[1], \Delta P_r(t)[0])$;
  $\theta_{average} = \theta_t - \theta_{t-1}$;
  $\Delta\theta = fabs(\theta_{compute} - \theta_{average})$;
  if $\Delta\theta > \Delta\theta_{threshold}$ **then**
    $P_r(t) = \xi * P_r^-(t) + (1-\xi) * (P_r(t-1) + \Delta P_{odometry})$;
    $P_f(t) = computeFrontPose(P_r(t))$;

Algorithm 1: Position correction based on Ackermann constraint

4.5 Summary

The focus of this chapter was on investigating the feature-based localization scheme under the EKF framework for autonomous vehicles. First, several coordinate systems were defined. Then the EKF localization related parts were extended. Related parts consisted of the map building, motion prediction, data association and measurement update. Especially, in the data association section, a two-point matching approach was proposed which can significantly improve the accuracy of localization. And the two-point matching approach is one of innovative approaches first proposed in this thesis. In order to obtain smooth trajectory, two trajectory smoothing methods were also proposed in the end. The smoothing method based on Ackermann constraint is another original idea in this work.

"To measure is to know. If you can not measure it, you can not improve it"

Lord Kelvin

5 Evaluation and Results

A feature-based localization method was introduced in the previous chapter. And this chapter will present its performance. The results show that it is comparable to the most expensive and currently used system, Applanix POS LV 510. It confirms the idea of using a cheaper localization solution to take the place of the expensive counterpart. Numerous real on-road tests in urban scenarios proved the effectiveness and robustness of the proposed localization scheme as well.

In the first section of this chapter, benchmark will be introduced, which consists of recording datasets and defining the evaluation metrics. Then in the second section, the evaluation results will be presented in terms of accuracy and precision. In the accuracy aspect, it will present the positioning and heading errors compared with the ground truth. In the precision aspect, the trajectory similarity will be tested. The real on-road tests will be presented in the third section. Finally, a concise discussion will be given in the fourth section.

5.1 Benchmark and Datasets

5.1.1 Benchmark

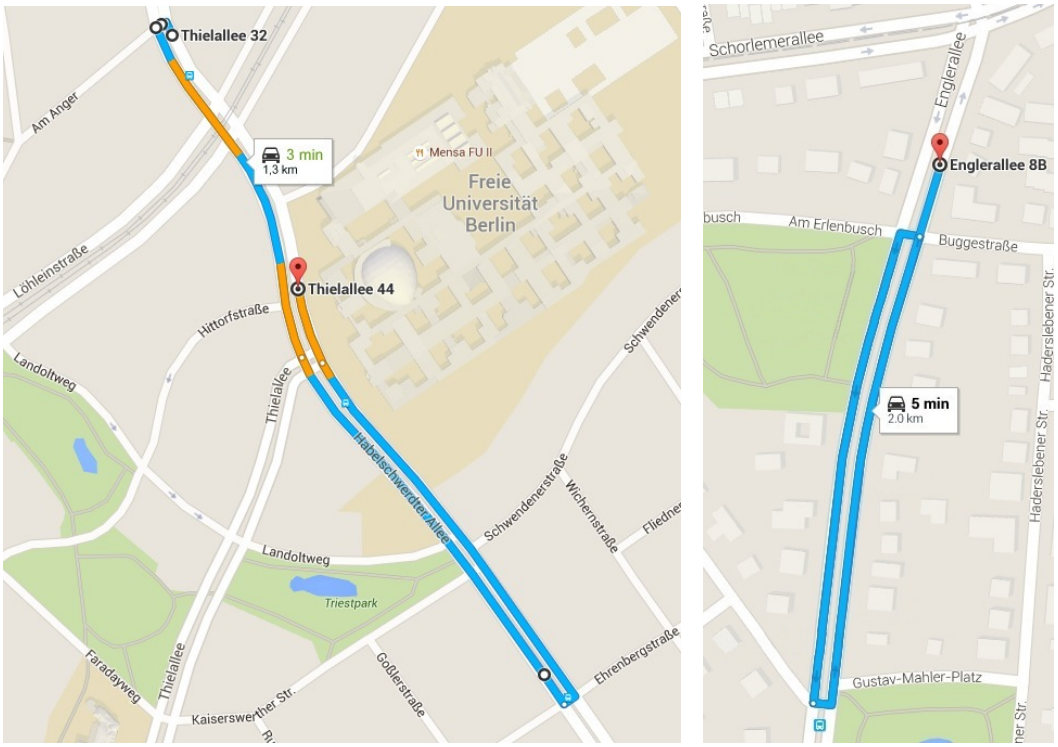
To evaluate the performance of the proposed localization algorithm, a benchmark (also referred to as the ground truth) should be defined or chosen at the beginning. First, it is capable of providing quantitative results to evaluate the effectiveness of an algorithm. Second, several existing algorithms can be compared when they are evaluated under the same benchmark. Basically, there are three ways to obtain the ground truth. One way relies on the highly accurate INS/GNSS systems (with differential GPS or Real-Time Kinematic technique for correction) [85], such as the

widely used Applanix POS LV [40, 44] or NovAtel SPAN-CPT [86, 87]. Unfortunately, this method only works fine in open outdoor environments, but does not work well in urban outdoor environments or indoor scenarios [85]. Another way is implemented through human labeled reference data [46, 88, 89]. In the work by [46], although the vehicle was equipped with the highly accurate DGPS/INS, large estimation errors were still encountered due to the outages and the multipath effects. So reference was still needed to be defined manually. The last method relies on the simulation [85]. It is a convenient and low-cost way, as it allows researchers to examine their algorithms under simplified or well defined conditions over and over again at an early stage. However, simulation cannot fully approximate the real situations, and thus it cannot substitute the real tests.

Besides, benchmarks must have higher degrees of precision than their compared counterparts. Therefore, they are difficult to get owing to constraints from techniques and cost. Even for the high accuracy Applanix, it is not so accurate as it expected to be. The specification reads that its RMS (Root Mean Square) error is forty centimeter even during the GPS outage [32]. The real field tests, however, showed that the lateral error for a single lap can be larger than 1 meters [4]. Actually accessing the quality of the Applanix is proven to be an uneasy task. Nonetheless, the Applanix POS LV and the Velodyne LIDAR are still the most accurate units we can rely on. Therefore, in this work, we got the ground truth by combining an Applanix POS LV 510 and a Velodyne HDL 64E LIDAR scanner.

5.1.2 Datasets

To verify the correctness of the proposed algorithms and evaluate their performance, we collected two datasets around our lab. As introduced before, these two datasets were recorded by MadeInGermany with the Applanix POS LV 510 and the Velodyne HDL 64E LIDAR. The first dataset is named after the street “Thielallee”, where is often chosen by our group to demonstrate the autonomous driving. This test site includes straight sections and U-turns, as shown in Figure 5.1a. The dataset is a 3 minutes long log file, covering 1.3 km trajectory. It recorded several driving states: driving, waiting for traffic lights, or making way for other vehicles during performing a U-turn. The second dataset is also named after the street “Englerallee”, as shown in Figure 5.1b. This test site includes straight sections and two U-turns as well. The dataset recorded a two and half lap trajectory. We also recorded a four lap trajectory in this test site to evaluate the trajectory similarity, which will be introduced in Section 5.2.4.



(a) Thielallee test site[10] **(b)** Englerallee test site[11]

Figure 5.1: Views of the test sites from Google Maps.

Some statistics on the pole-like features in both test sites are listed in Table 5.1. The information includes the test sites' length and number of features and their densities. Density here refers to the number of features per meter. As it can be seen from the densities, there is roughly one feature for every two meters spreading along the streets. So these two areas are rich in pole-like features. They are quite suitable for testing our localization algorithms.

Table 5.1: Map statistics on the test sites

Test site	Length (m)	Pole-like features (Quantity)	Density (1/m)
Thielallee	745	378	0.51
Englerallee	300	224	0.74

The number of features extracted from a full 360 degree scan (or one entire rota-

[10]Image created through Google Maps(Map data ©2016 GeoBasis-DE/BKG (©2009), Google)
[11]Image created through Google Maps(Map data ©2016 GeoBasis-DE/BKG (©2009), Google)

tion of Velodyne) is summarized in Table 5.2. The same information is shown as a boxplot in Figure 5.2. It can be seen that they have the same median value. But their entire distributions are not the same. Actually, both the table and the figure do not show any information about the distribution of a specific whole scan. The value is unevenly distributed around the scan. When the scan directions point towards the road directions, more features will be extracted, or else a few or none features are detected. This is the reason why an entire scan rather than part of scan, like one sixth, is chosen to perform the data association.

Table 5.2: Statistics on the number of features extracted from one entire scan

	Minimum	Maximum	Median	Mean
Thielallee	2	30	13	14
Englerallee	0	28	13	13

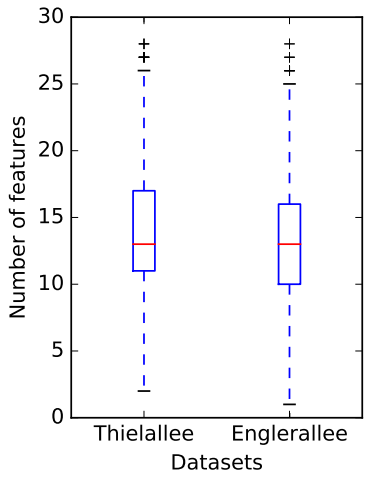


Figure 5.2: The number of features extracted from one entire rotation of velodyne for two datasets.

As introduced before, the features in the map also present some degree of uncertainties, which contribute further errors to the estimation results. The uncertainties are measured by the size of error ellipses. They are presented in two ways. First, statistical results are presented in Table 5.3. The semi-major axes and semi-minor axes of the uncertainty ellipses represent one standard deviation from their means. It can be seen that the longest semi-major axis is more than 60 cm and the longest semi-minor axis is more than 30 cm. Even the mean error is quite large. Second, the positional uncertainties are also depicted in Figure 5.3 for the Thielallee dataset

and Figure 5.5 for the Englerallee dataset based on the LOG maps (see section 4.3.1), Figure 5.4 and 5.6 based on the FIS maps (also see section 4.3.1). Since there is no uncertainty information available for features in the FIS map, all the features are assigned the same uncertainty value, 50 cm for the semi-major axes and 15 cm for semi-minor axes. It can be seen that the features are not evenly distributed, including both sparse and dense areas. The sizes of ellipses are also different from area to area. All of these characteristics lead to changing localization performance.

Table 5.3: Statistics on the uncertainty ellipses of the LOG Maps for two datasets, defined by a semi-major axis a and a semi-minor axis b.

	Thielallee		**Englerallee**	
	a (cm)	b (cm)	a (cm)	b (cm)
Mean	27	13	35	16
Median	27	13	36	17
Maximum	69	31	66	37
Minimum	5	1	4	0

5.1.3 Evaluation Metrics

For evaluating the positioning performance, several metrics were adopted, such as the along track error (ATE), the cross-track error (XTE), the longitudinal error, the lateral error and so on. However, they are not equivalent, as different metrics focus on different properties. The along track error is employed to calculate the error in the intended direction, and it is a projected distance. The value is positive when the error is along the intended direction, and otherwise negative. The cross-track error calculates the distance from the estimated position to a line formed by its nearest way-points in the intended direction. It measures how far the estimated position drifts away from the left or right of the intended course. This metric is widely used in the GPS associated navigation scenarios. The longitudinal and lateral error are reported with regard to the ground truth. Longitudinal refers to forward and backward of the vehicle's moving direction and lateral refers to left and right. Such far, evaluating the along track error and the cross track error needs to use two way-points to determine the intended direction. For the longitudinal and lateral error, they are difficult to calculate as the road is not always straight and spreading

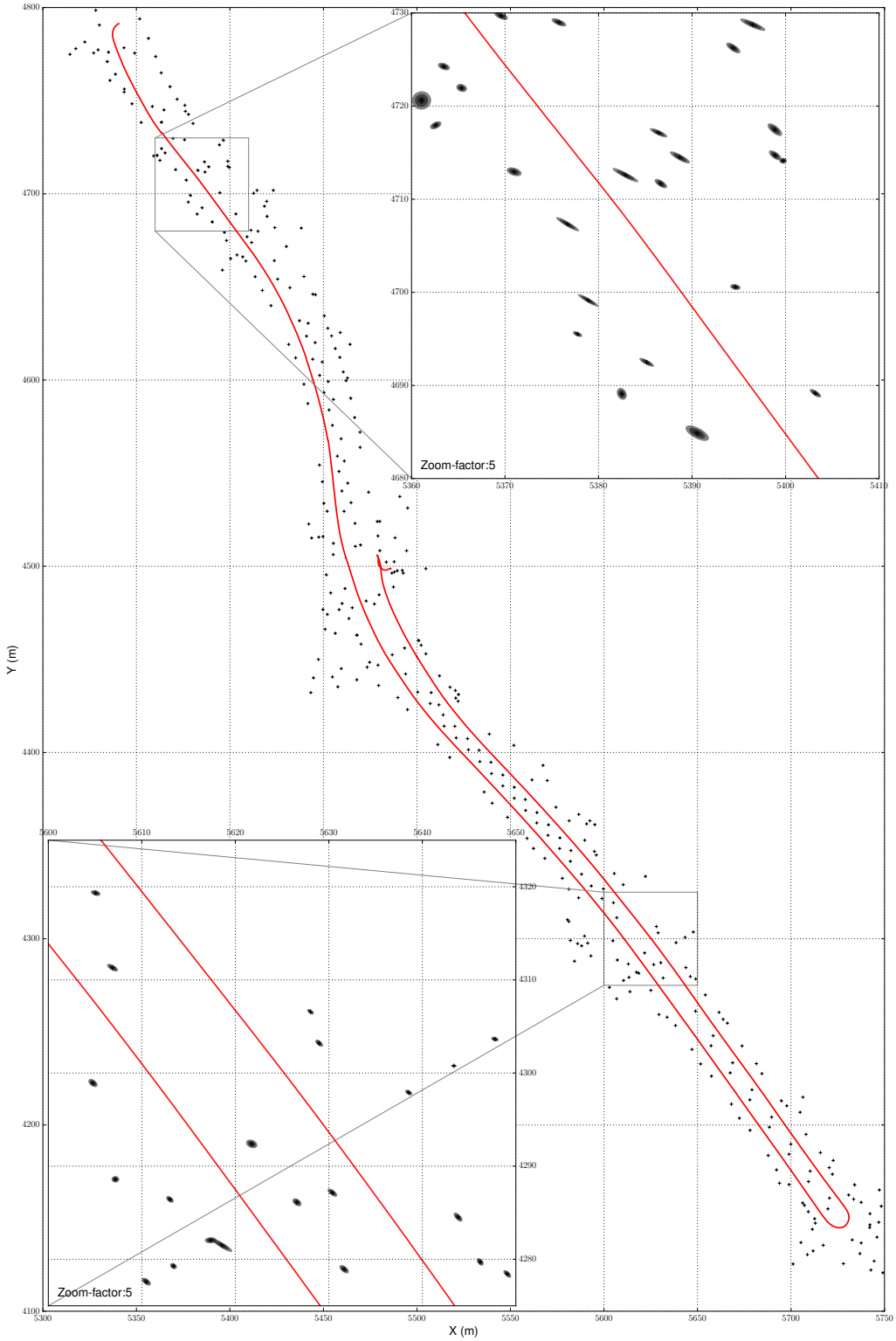


Figure 5.3: Overview of the LOG map for the Thielallee dataset. The red line is the trajectory. The black ellipses present three standard deviation of their means.

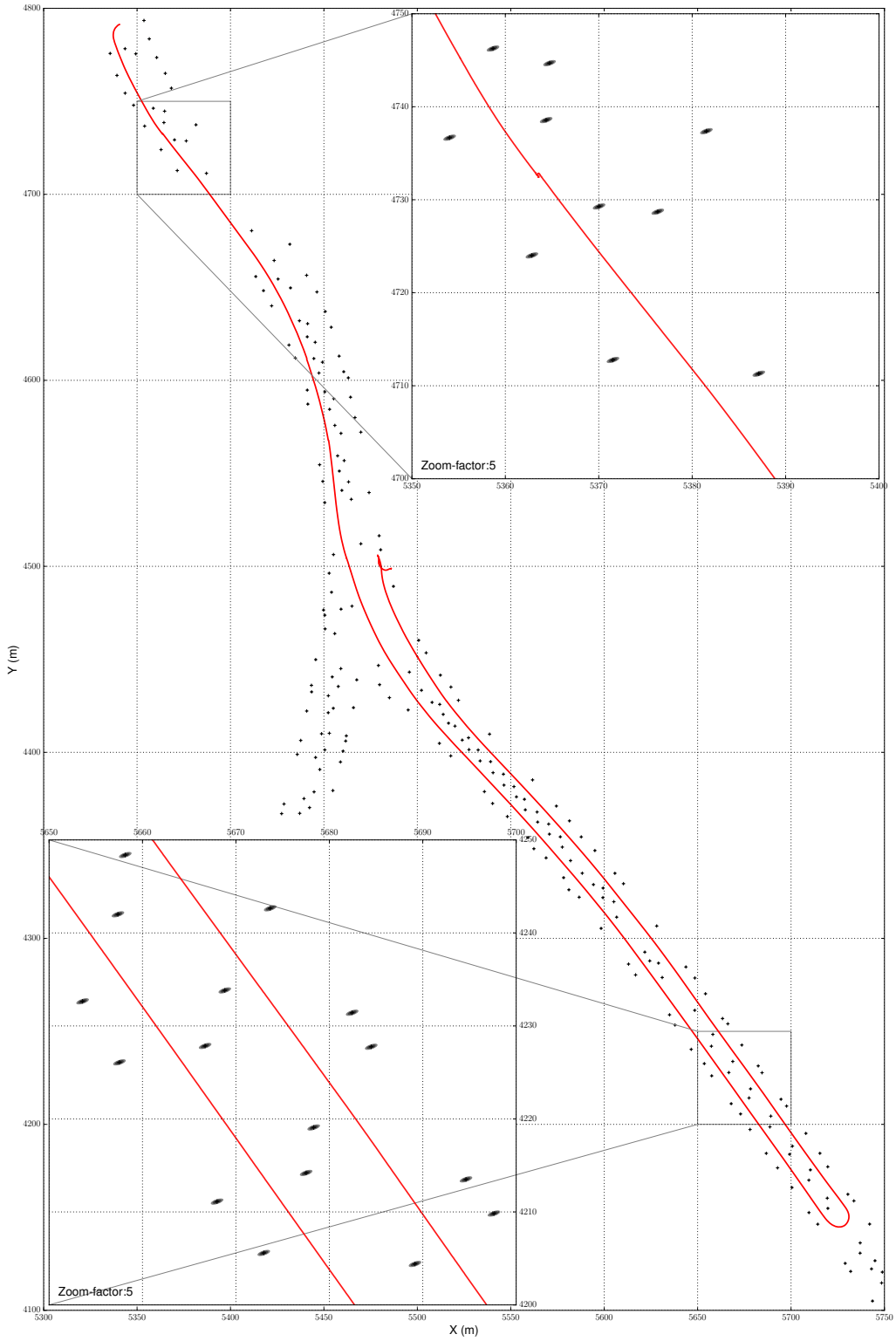


Figure 5.4: Overview of the FIS map for the Thielallee dataset. The red line is the trajectory. The black ellipses present three standard deviation of their means. All the features are assigned the same uncertainty value due to lack of data.

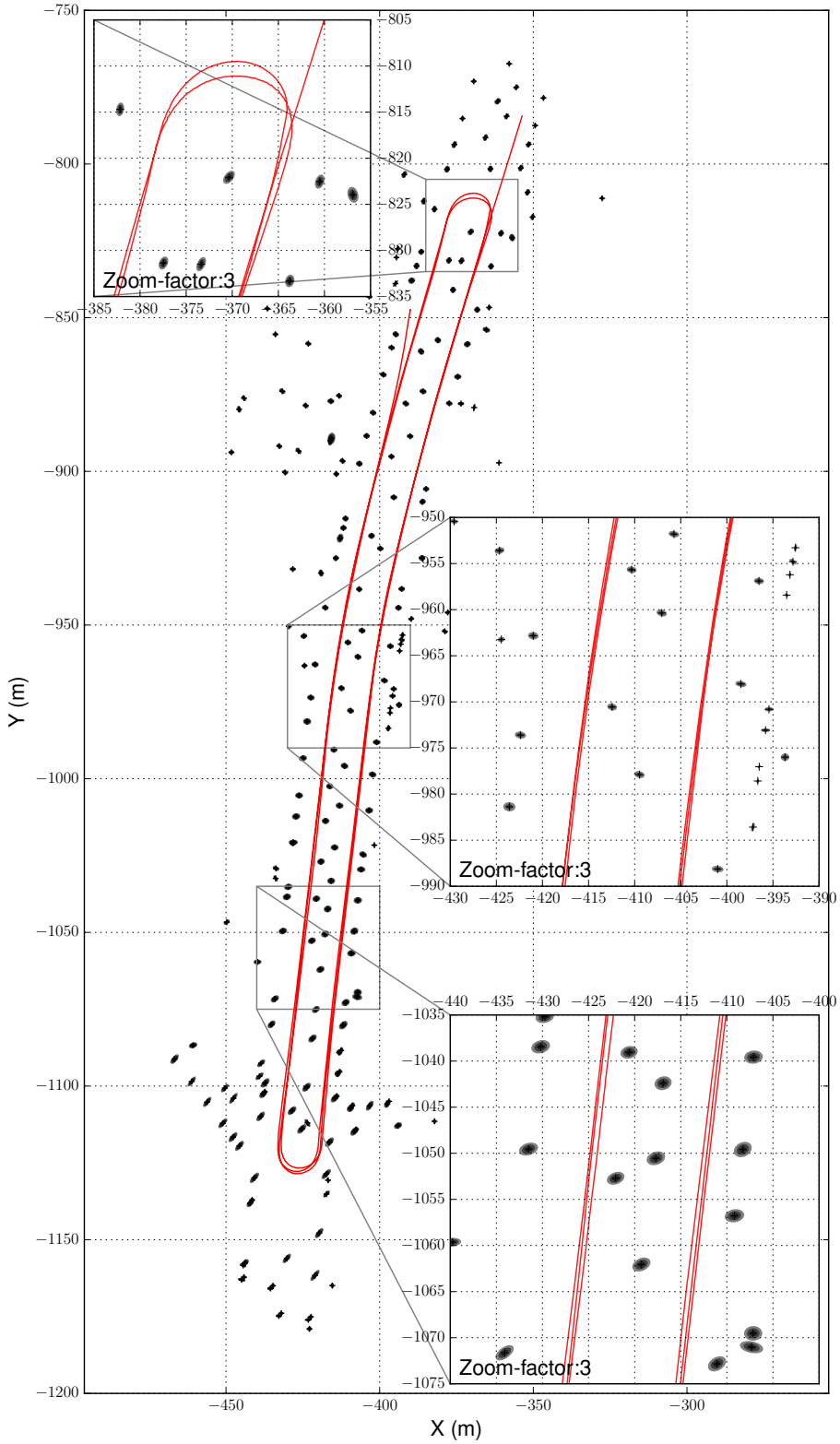


Figure 5.5: Overview of the LOG map for the Englerallee dataset. The red line is the trajectory. The black ellipses present three standard deviation of their means.

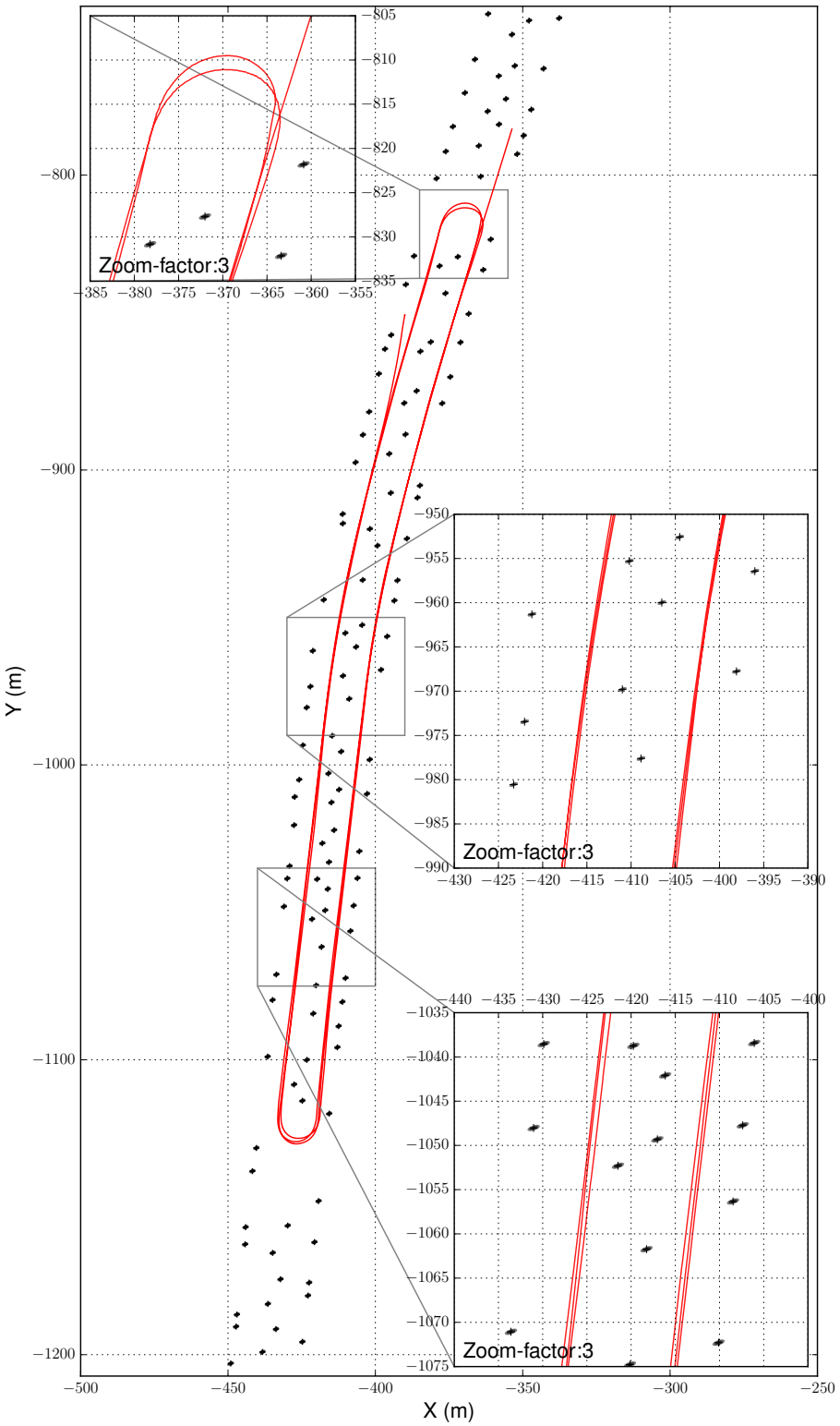


Figure 5.6: Overview of the FIS map for the Englerallee dataset. The red line is the trajectory. The black ellipses present three standard deviation of their means. All the features are assigned the same uncertainty value due to lack of data.

in one direction. And for safety reason, the localization error in the lateral direction is more important than in the longitudinal direction as the lane width is narrower than its length. Therefore, we introduced the dubbed direct-track error (DTE) in this work. We define the direct-track error as the distance from the estimated position to a line formed by the ground-truth position and the yaw angle of the vehicle, as depicted in Figure 5.7. In spirit, the direct-track error defined here is similar to the cross-track error.

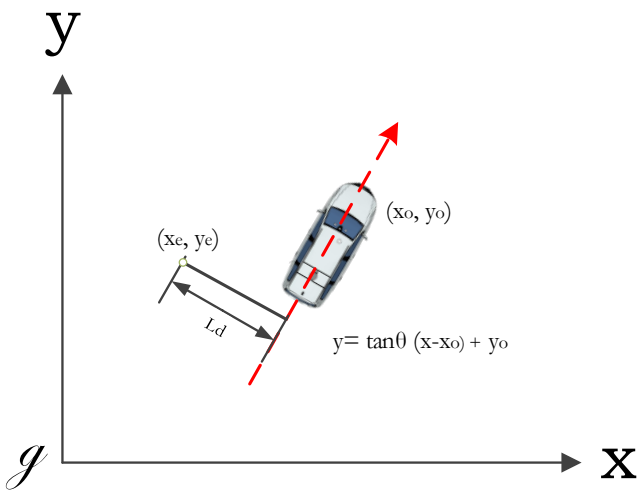


Figure 5.7: Illustration of the direct-track error. Here (x_e, y_e) is the estimated vehicle position. $(x_o, y_o,)$ and θ are vehicle's ground truth position and heading. These two parameters can be expressed by a line passing through $(x_o, y_o,)$ with the slope $tan\theta$. L_d is the direct-track error.

Another important metric is the deviation from the ground-truth heading, or the heading estimation error, as the heading plays more important role in the pose estimation. First, an error existed in the heading estimation can contribute error to the pose estimation. Second, a big heading estimation error can lead to failure during the data association stage or loop closure stage. Thus, we employ both the positioning errors and the heading estimation error to evaluate the localization performance of the algorithm.

5.2 Experimental Results

The experimental results are presented in this section and the quantitative results are presented in the beginning. With such results, we can get a better understanding of the localization performance. The performance is evaluated in terms of accuracy and precision. To evaluate their accuracies, two kinds of errors, the positioning estimation errors and the heading estimation errors, are calculated. The estimation errors are calculated at each time step as the autonomous vehicle is a high mobility robot and for safety reasons, it needs to know its position in real-time. The interval between two consecutive time steps is equal to that of wheel speed trigger events, around 0.01 seconds. Therefore, many charts shown below put the time step on their horizontal axes or x-axes, that is, the estimation errors over time step are presented. The positioning estimation errors include the lateral error, the longitudinal error and previously dubbed direct track error. However, such errors are presented only for the purposes of reference as the ground truth is also not precise. To evaluate their precision, the method for measuring trajectory similarity is employed.

Table 5.4: Comparison of RMS errors for two datasets, measured in centimeters

		ATE	XTE	Lateral	Longitudinal	DTE
Thielallee	LOG Map	15.95	19.20	19.54	15.52	20.94
	FIS Map	101.59	87.45	106.23	81.65	104.43
Englerallee	LOG Map	55.79	47.22	45.47	57.22	51.73
	FIS Map	98.19	103.80	117.70	81.02	112.98

Table 5.5: Comparison of heading errors for two datasets

		Mean	Std.dev.	RMS
Thielallee	LOG Map	0.04°	0.01°	0.011°
	FIS Map	0.14°	0.29°	0.33°
Englerallee	LOG Map	−0.22°	0.34°	0.41°
	FIS Map	−0.27°	0.26°	0.38°

Before starting new subsections, some results are first summarized in Table 5.4 and 5.5, which were obtained by playing the log files over 50 times. The RMS errors for positioning estimation are presented in Table 5.4. The performance for the

Thielallee dataset is much better than that of the Englerallee dataset as the Englerallee dataset includes two and half rounds. Although during data recording stage, the human driver was trying to keep two rounds with the same trajectories, the Applanix POS LV still showed big discrepancies between two rounds. These discrepancies can be a few meters, as obtained from the empirical data. Thus, for multiple rounds trajectory, another metrics should be adopted rather than directly use the Applanix results as reference. Nonetheless, it still maintains the error levels at a comparable order of magnitude as the former dataset. For the heading estimation error, the evaluation for the Thielallee dataset also has better results.

As introduced in the previous section, two datasets are used to evaluate the performance of the proposed algorithm. So the results for each dataset are shown separately. The following two subsections will show the results in figures which will take up many pages. Due to page restrictions, for the same dataset, unless explicitly point out, only the results based on the LOG map will be presented.

5.2.1 Thielallee Dataset

To begin with, Figure 5.8 and 5.9 show the comparisons of the estimated trajectories with the Applanix reported trajectory. The blue line indicates the estimated trajectory while the green line shows the estimated trajectory before smoothing. And the red line represents the Applanix reported ground truth trajectory. It can be seen that the estimated trajectories match well with the Applanix trajectory most of the time, except the U-turn section. The reason is that the heading estimation error is bigger during performing U-turns. The bigger heading estimation errors lead to big positioning errors. Then, to show how the positioning errors change, trajectories are colored by the positioning errors in Figure 5.10. Finally, Figure 5.11 shows the lateral and longitudinal errors in histogram form. It can be seen that these errors are mainly concentrated within ± 40 cm of their ground truth. In some poor feature areas, the performance is a little bit worse than this value but only exists in the tails of the histograms.

The number of poles utilized to estimation may affect the performance of our tests. As can be expected, the estimation accuracy depends to some extent on the number of pole-like features. The direct track error versus the number of poles is shown as a boxplot in Figure 5.12. The estimation errors are plotted as a function of the number of poles for the Thielallee dataset. The matched poles varied between 3 and 27. It can be seen that as the number of poles increases, the error has a trend toward decreasing. However, this trend is not so dramatic because the accuracy is

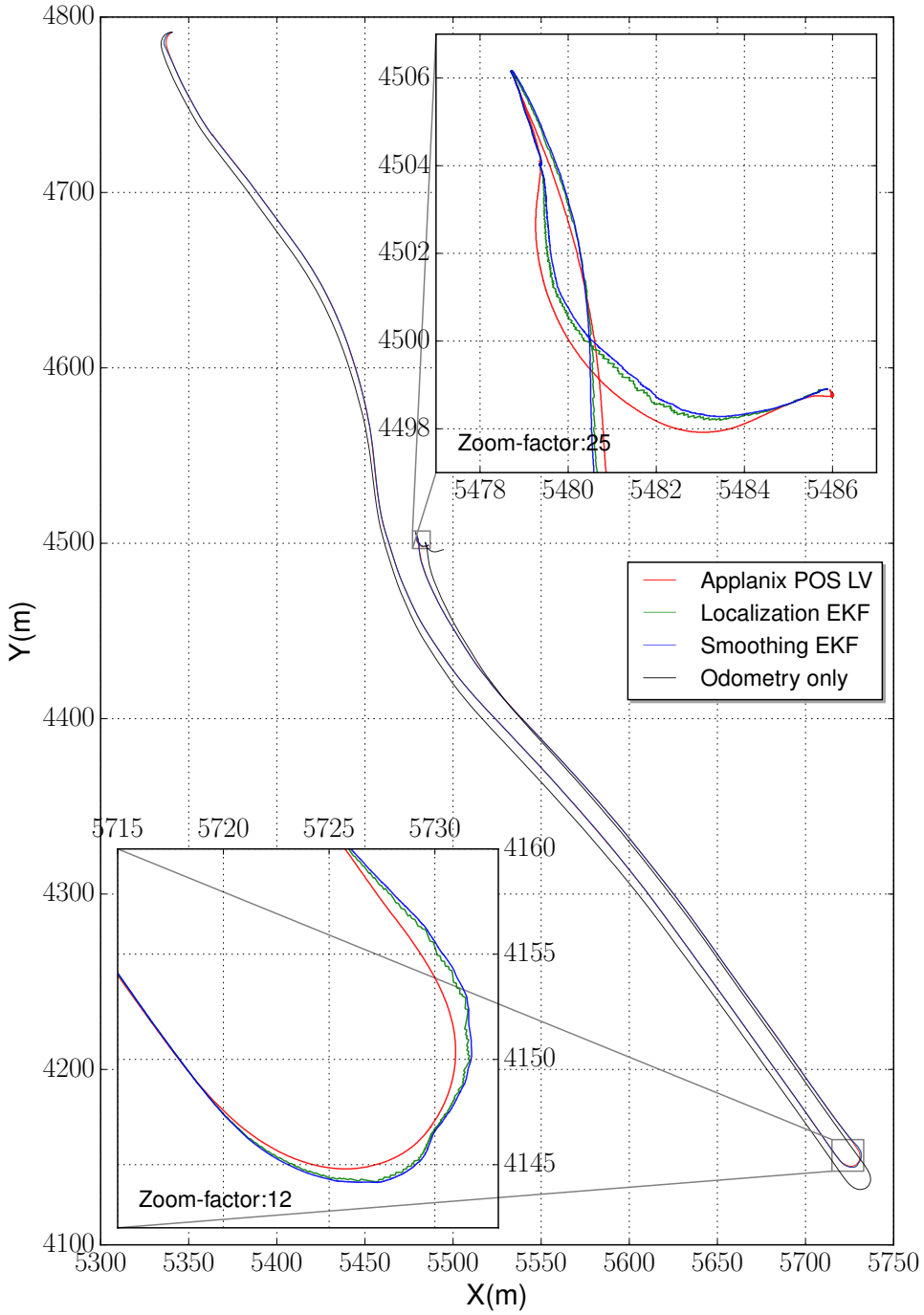


Figure 5.8: Comparison of trajectories for the Thielallee dataset by using the LOG map. The red line represents the Applanix reported ground truth trajectory. The blue line indicates the estimated one and the green line shows the estimated trajectory before smoothing. And the trajectory estimated by dead reckoning is colored black. Two important sections are zoomed in with different zoom factors to highlight details.

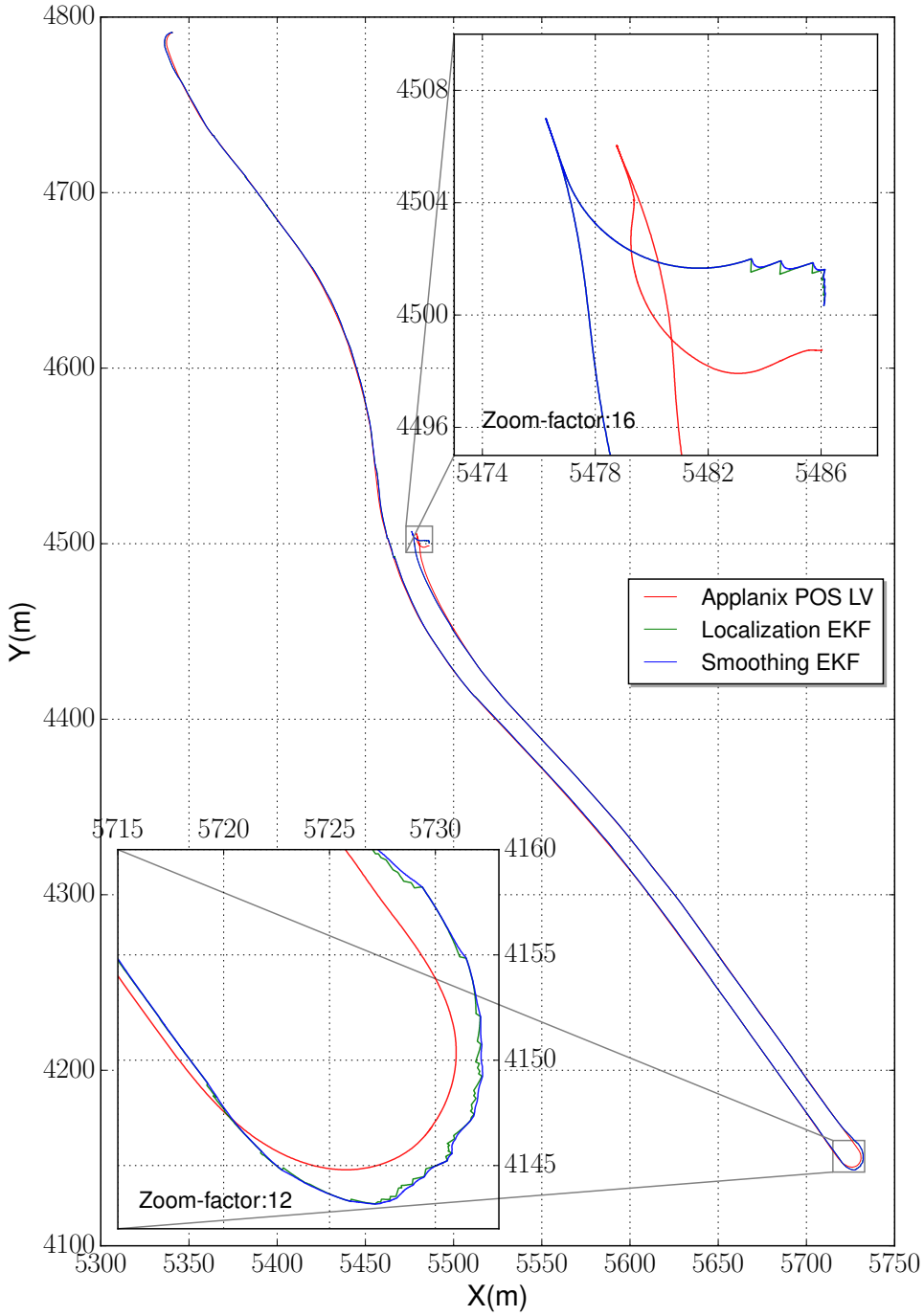


Figure 5.9: Comparison of trajectories for the Thielallee dataset by using the FIS map. The red line represents the Applanix reported ground truth trajectory. The blue line indicates the estimated one and the green line shows the estimated trajectory before smoothing. And the trajectory estimated by dead reckoning is colored black. Two important sections are zoomed in with different zoom factors to highlight details.

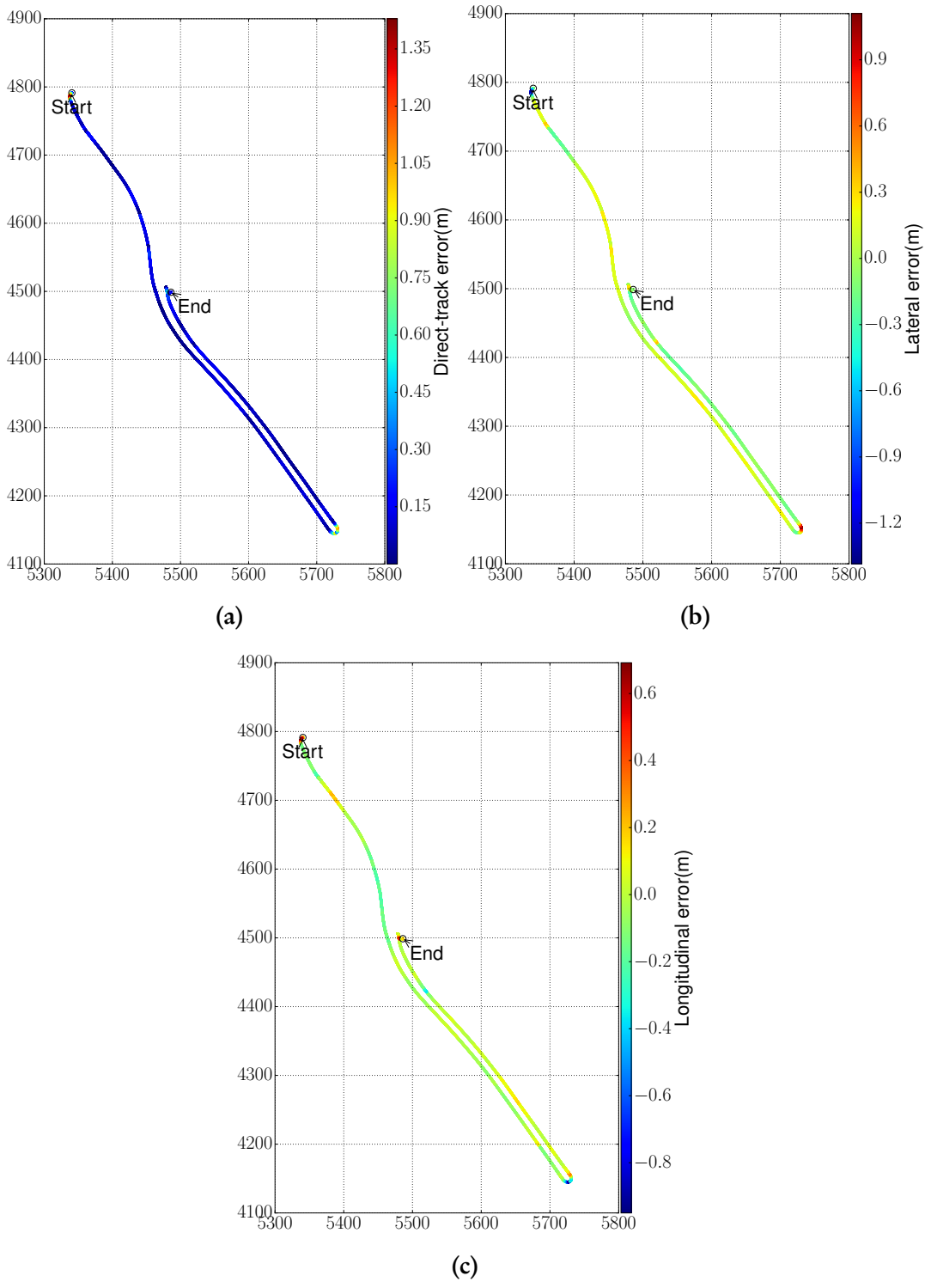


Figure 5.10: Trajectories colored by the positioning errors for the Thielallee dataset. In (a), for the direct-track error , the darker the red, the bigger the error. In (b), for the lateral error, the darker the blue, the bigger the error. In (c), for the longitudinal error, the darker the red, the bigger the error.

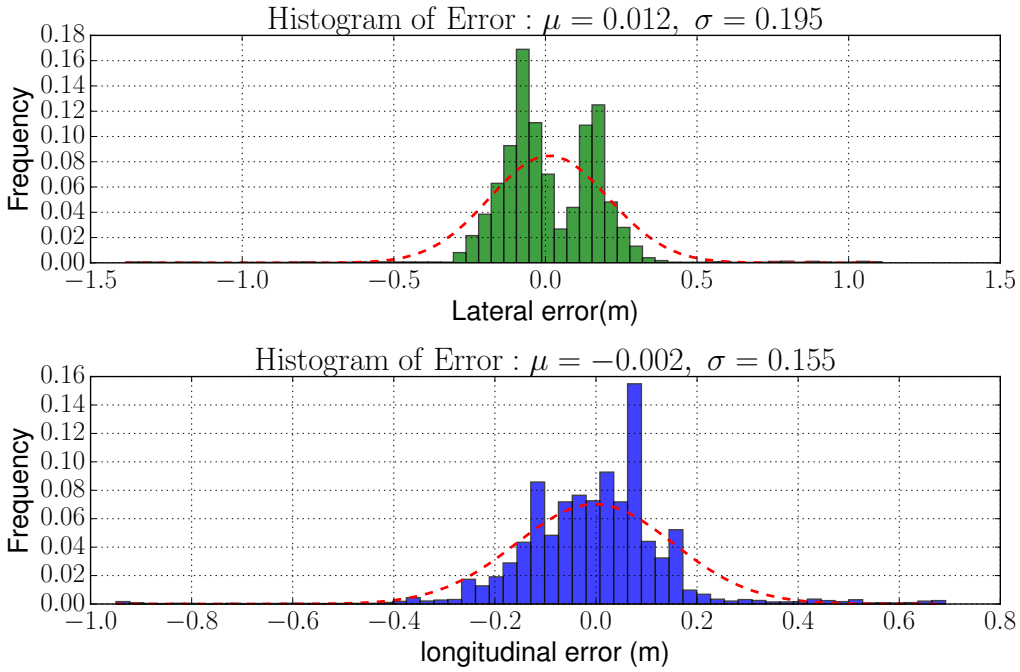


Figure 5.11: Histograms of lateral and longitudinal error for the Thielallee dataset. It can be seen that these two errors are mainly concentrated within ±40 cm of their ground truth.

also affected by other reasons, such as the feature uncertainties in the prior map.

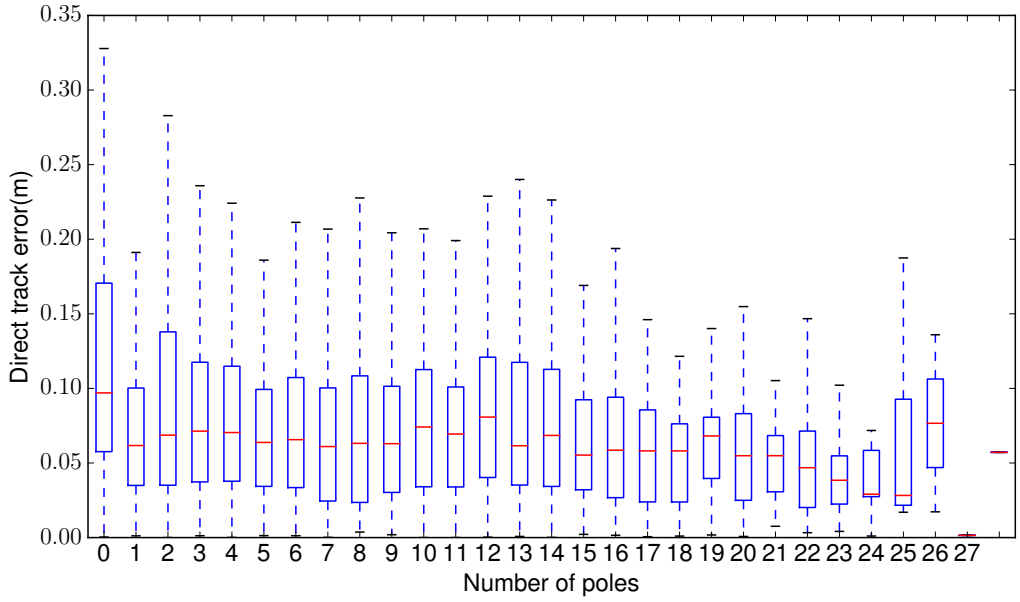


Figure 5.12: Boxplot of the direct track error versus the number of poles for the Thielallee dataset.

After evaluating the positioning performance, the heading estimation perfor-

mance can be seen from Figure 5.13 to 5.15. In Figure 5.13, the estimated yaw angle matches well with the ground truth counterpart. And their difference can be seen from Figure 5.14. Most of the time, the error is within ±0.2 degrees, and the worse parts happen when the vehicle makes sharp turns, like U-turns. Due to the influence of the lag problem from the gyroscope, the estimation is smaller than the Applanix value at the beginning of the sharp turns. The same results are also shown in histogram form, as demonstrated in Figure 5.15. The histogram shows that most values lie within 2 sigmas, ±0.2 degrees. Therefore, the heading estimation is quite accurate.

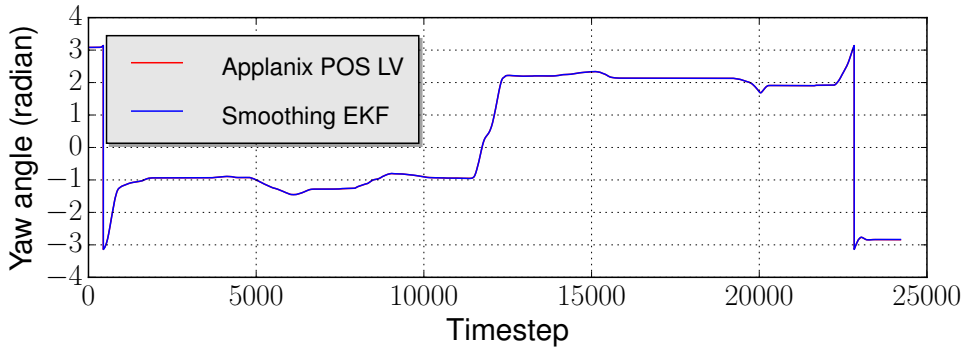


Figure 5.13: Comparison of estimated yaw angle against the ground truth. The estimated value matches well with the ground truth. There is no distinct discrepancy from an overall perspective.

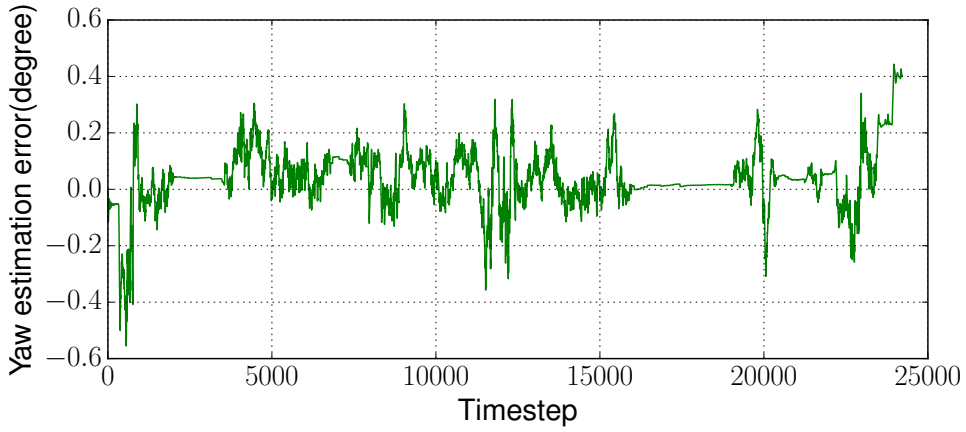


Figure 5.14: The heading estimation error over time for the Thielallee dataset. Though under the worse case, the deviation is bigger than 0.4 degrees, such case is very rare. Most frequently, the estimated errors are within ± 0.2 degrees.

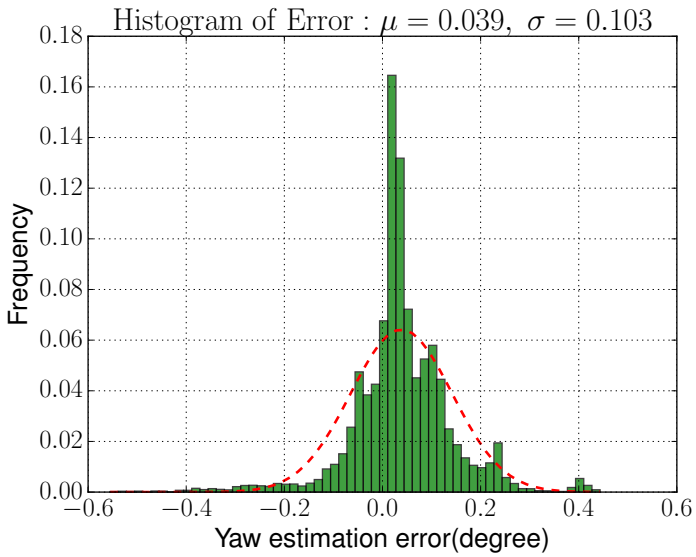


Figure 5.15: Histogram of the heading estimation error for the Thielallee dataset. The red dash line here is employed to fit the histogram. It indicates that the error follows the Gaussian distribution. Knowing the distribution and its related parameters, it can be clearly seen that over 96% of values are concentrated within ±0.3 of the ground truth.

5.2.2 Englerallee Dataset

For the Englerallee dataset, the evaluation process is the same as that of the Thielallee dataset. The comparison of the estimated trajectories with the Applanix reported ground truth can be seen in Figure 5.16 and 5.17. And the trajectories are colored by the positioning errors in Figure 5.18. It can be seen that the worse performance also happened when the vehicle performed U-turns. Figure 5.19 shows the lateral and longitudinal errors in histogram form. Comparison of Figure 5.22 and Figure 5.21 shows that the heading estimation follows a similar pattern. It performs badly when the vehicle makes U-turns. Apart from this, it achieves as good performance as the former dataset, i.e., it also stays within ±0.5 degrees. By comparison, the performance is not as good as that of the Thielallee dataset, because the Englerallee dataset has two laps and the GPS values for the same position are different. Actually, the discrepancy for two laps is quite large. This, on the one hand, increased the uncertainties of features' positions upon creating the feature maps. On the other hand, here directly taking the uncorrected value as the ground truth is not a suitable reference. To achieve better results, maybe in the future the corrected value should be used as the ground truth, such as using post-processing results. Besides, for a multi-lap trajectory, assessing its repeatability is a good choice [90].

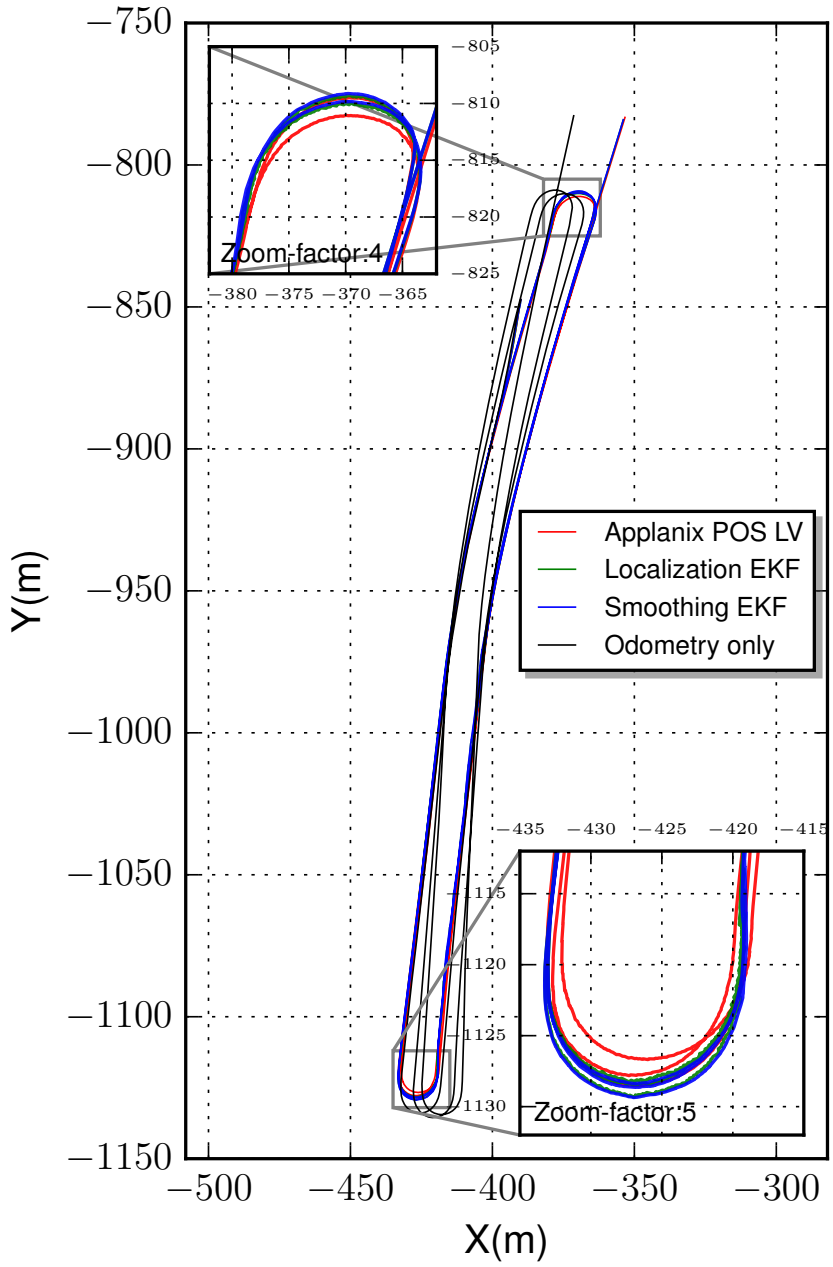


Figure 5.16: Comparison of trajectories for the Englerallee dataset by using the LOG map. The red line represents the Applanix reported ground truth trajectory. The blue line indicates the estimated one and the green line shows the estimated trajectory before smoothing. And the trajectory estimated by dead reckoning is colored black. Two important sections are zoomed in with different zoom factors to highlight details.

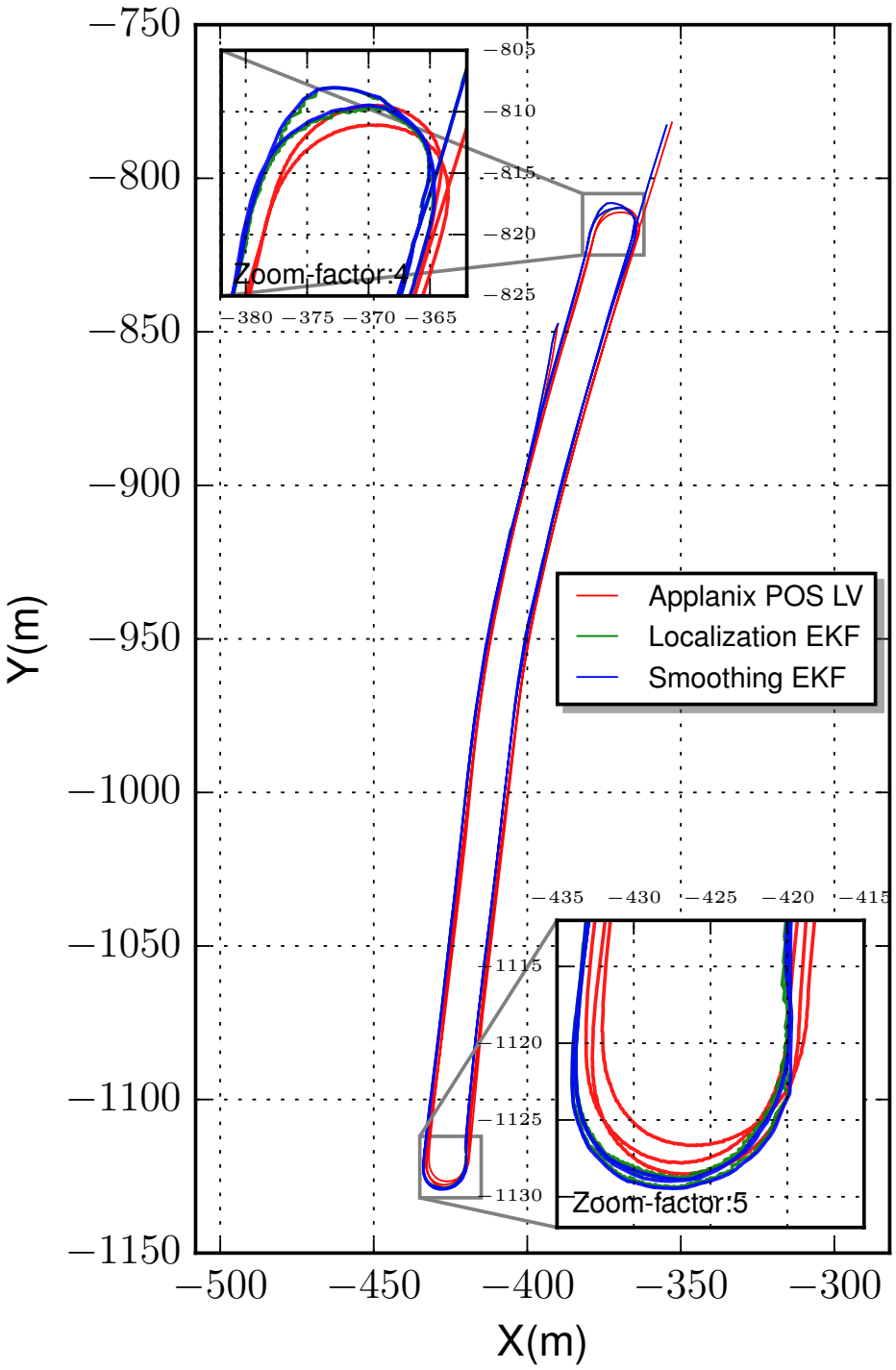


Figure 5.17: Comparison of trajectories for the Englerallee dataset by using the FIS map. The red line represents the Applanix reported ground truth trajectory. The blue line indicates the estimated one and the green line shows the estimated trajectory before smoothing. And the trajectory estimated by dead reckoning is colored black. Two important sections are zoomed in with different zoom factors to highlight details.

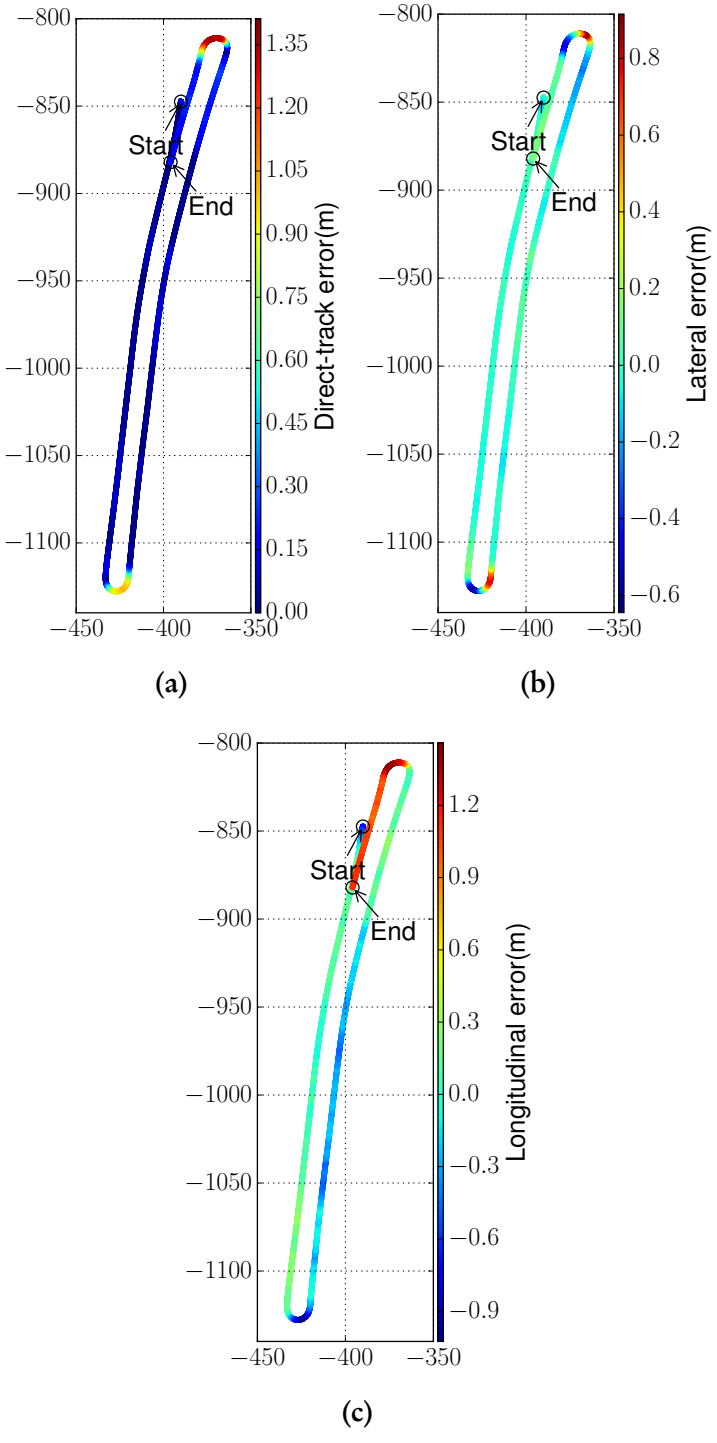


Figure 5.18: Trajectories colored by the positioning errors for the Englerallee dataset. In (a), for the direct-track error, the darker the red, the bigger the error. In (b), for the lateral error, the darker the blue, the bigger the error. In (c), for the longitudinal error, the darker the red or the blue, the bigger the error.

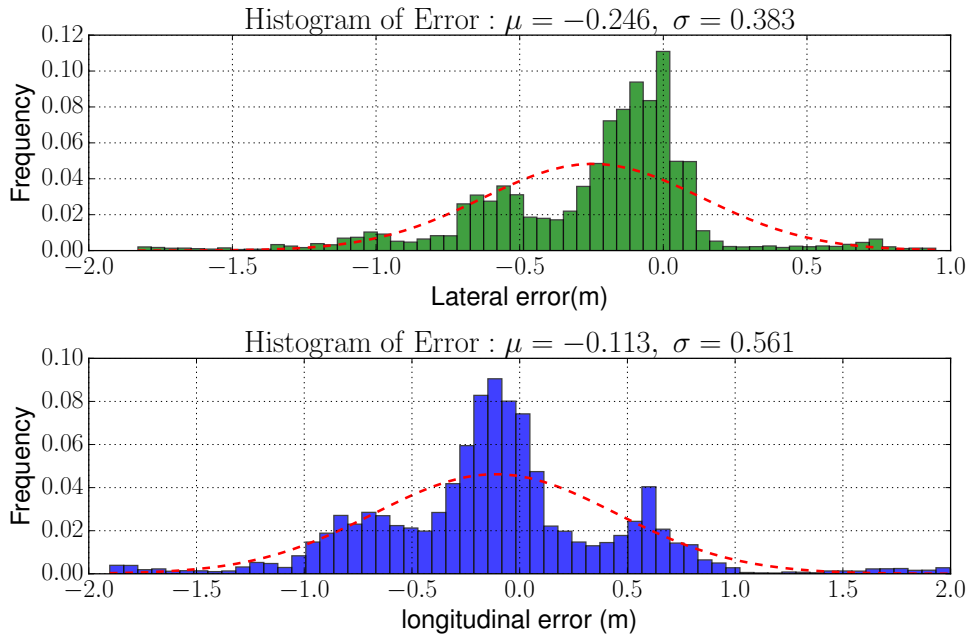


Figure 5.19: Histograms of lateral and longitudinal error for the Englerallee dataset.

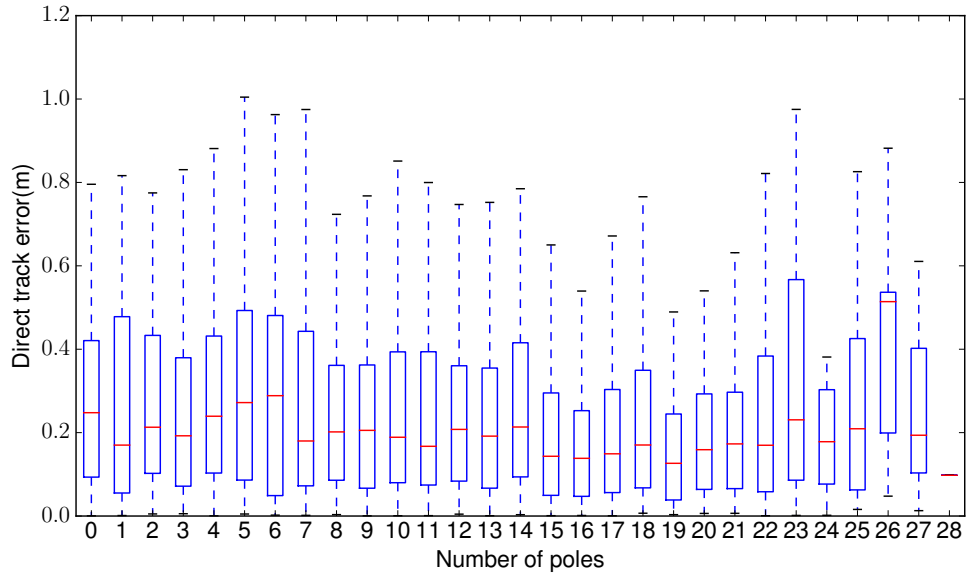


Figure 5.20: Boxplot of the direct track error versus the number of poles for the Englerallee dataset.

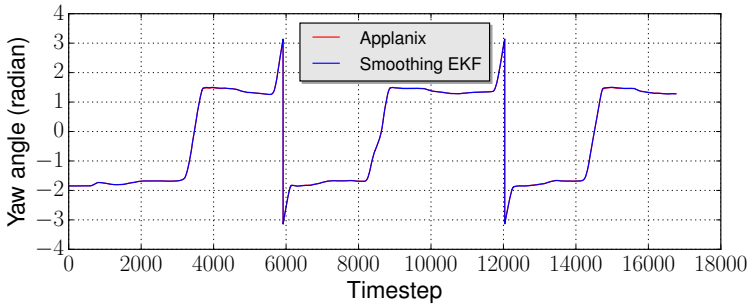


Figure 5.21: Comparison of estimated yaw angle against the ground truth. The estimated value matches well with the ground truth. There is no distinct disparity from an overall perspective. And the sharp changes of angle indicate U-turns

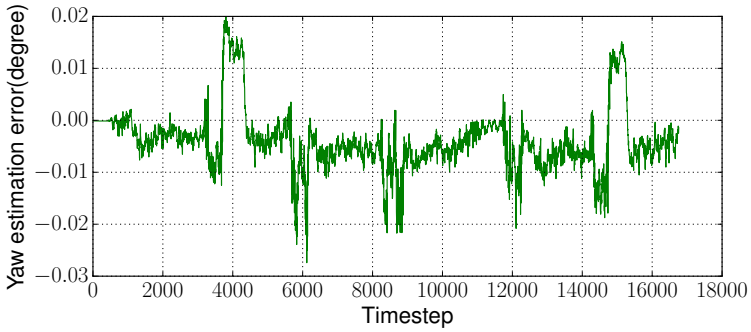


Figure 5.22: The heading estimation error over time for the Englerallee dataset. If compare this figure with Figure 5.21, it can be clearly seen that the performance is quite bad when the vehicle performs U-turns. Except that the errors almost stay with ±0.5 degrees.

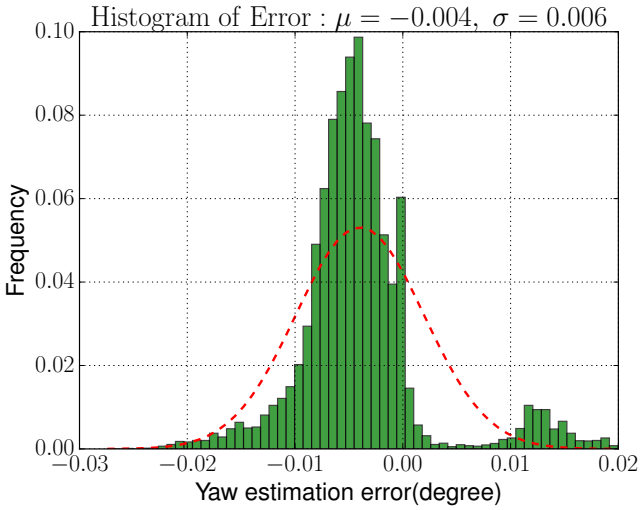


Figure 5.23: Histogram of the heading estimation error for the Englerallee dataset. The estimation error follows Gaussian distribution. About 99.7% of all errors lie with ±1 degrees.

5.2.3 Trajectory Smoothing

As introduced in the previous chapter, it is difficult to get very smooth trajectories by using only one EKF. Two schemes are employed. Adding another EKF, here dubbed as the double EKF scheme, is the main method. The other one is based on the Ackermann constraint. The performance is shown in Figure 5.24 and 5.25. As can be seen, the trajectories have been greatly smoothed via these two methods. And the Ackermann constraint scheme gets better performance at the end of U-turns, as the trajectories are even straighter.

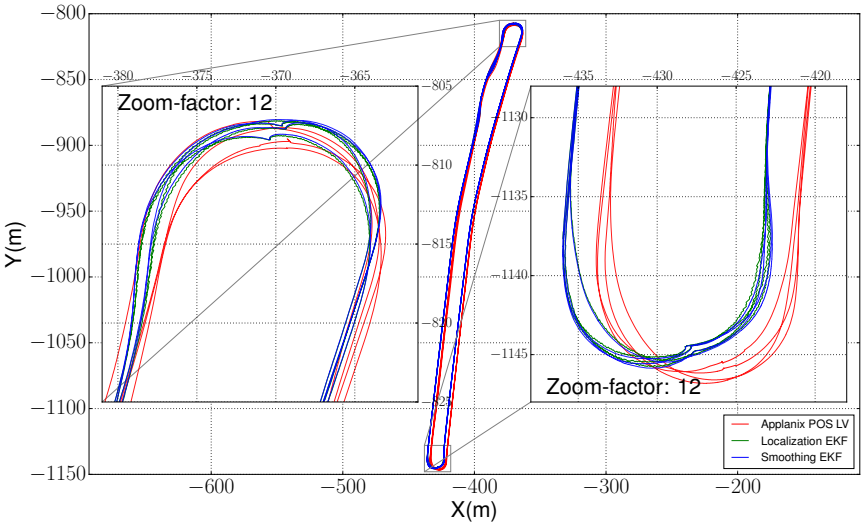


Figure 5.24: Trajectory smoothing based on the double EKF scheme.

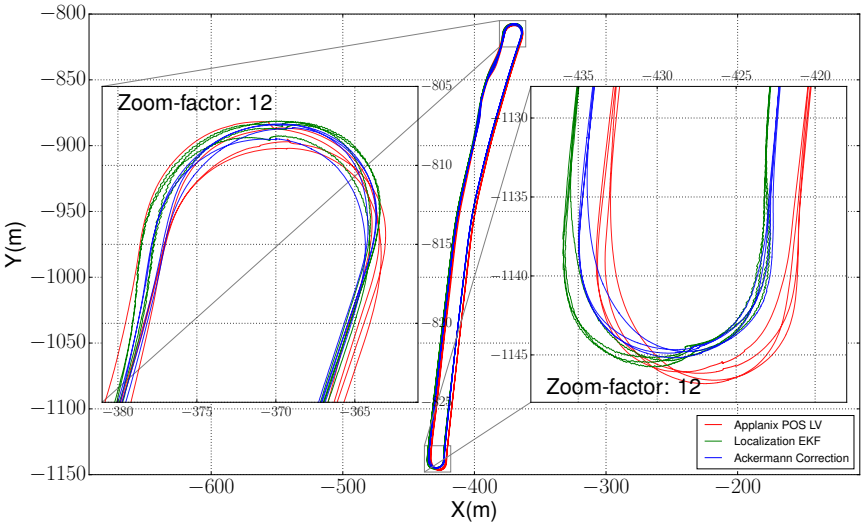


Figure 5.25: Trajectory smoothing based on the Ackermann constraint scheme.

5.2.4 Trajectory Similarity Measure

The accuracy of Applanix results is limited and the reported value has dramatic offset, sometimes up to several meters. Under such circumstance, directly comparing the estimation with the Applanix results does not make any sense. Therefore, extra metrics should be employed. Here the trajectory similarity measure is proposed, which is similar to reproducible measure [90] or precision measurement. The essence of the trajectory similarity measure is that the localization algorithm ideally should gain the same estimation when the vehicle travels through the same position. Trajectory similarity measures have been intensively studied in the community [91, 92]. Several popular measures were proposed, including Euclidean distance measure, Dynamic Time Warping (DTW) based, Edit distance based and longest common subsequence based measures [92]. In this work, the Dynamic Time Warping based measure is used.

To test the trajectory similarity, an extra dataset was recorded in the test area of Englerallee. The test driver was trying to drive on the same route on all four laps. For each lap, the starting and stopping position were almost in the same position, within 10 cm uncertainties. The Applanix POS LV reported trajectory and the estimated trajectory are shown in the Figure 5.26. As it can be seen that the estimated trajectory has higher similarity than the Applanix reported one. Especially at the starting and ending position, Applanix had over 1 meters drift for one lap. The results of the trajectory similarity measure based on the Dynamic Time Warping method are given in Table 5.6. Smaller numbers mean higher similarity, as the DTW method calculates the minimum overall distance between two laps. So it also has unit, here in meters. The first lap is chosen as the reference trajectory. The three remaining laps are compared with that of the reference. According to the results, we can safely conclude that the estimated trajectory has higher similarity than the one reported in Applanix.

Table 5.6: Comparison of the trajectory similarity between the estimated and the Applanix reported trajectory. T1 to T4 means the first to the fourth lap. T1/T2(also T3 and T4) means the comparison between the first and the other three laps.

	T1/T2	T1/T3	T1/T4
Estimated	689.39 m	637.47 m	882.88 m
Applanix	866.54 m	956.56 m	1462.18 m

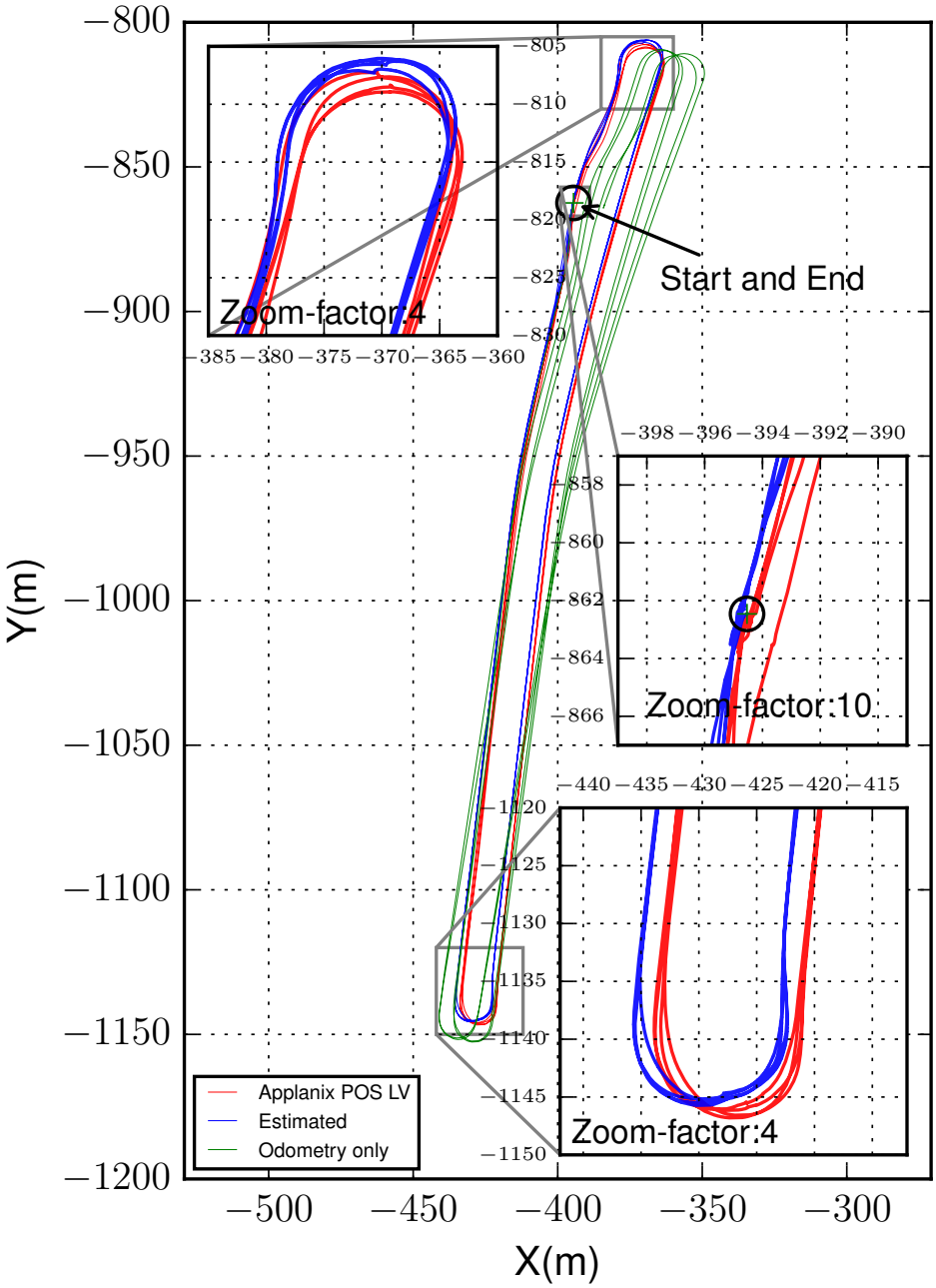


Figure 5.26: Trajectory similarity measure in the test area of Englerallee. Four laps were driven in the test area, starting and stopping in the same position for all laps. It can be seen that the estimated trajectory has higher similarity than the Applanix reported one. Especially at the starting and ending position, Applanix had over 1 meters drift for one lap.

5.2.5 Execution Times

For autonomous vehicles, the higher the data output rate the localization unit provides, the safer the vehicle drives and the smoother the controller works. In this work, the data output rate is around 100 Hz, which is the same as that of the speed sensors. For a real-time application, this means the algorithm should be executed within 10 ms. The most time consuming part of this work is the localization and visualization section. The localization section consists of the feature matching and two-point localization. The execution time of this section is positively correlated with the number of detected features. Figure 5.27 shows the relationship that as the number of poles increases, the related execution time increase as well. However, the longest execution time is still shorter than 0.3 ms, which is far less than 10 ms. As for visualization section, the average execution time is about 0.19 ms. Even the longest time is shorter than 6 ms. Thus, the proposed algorithm can be executed in real-time. Please note that all these results were obtained on an Intel i7 4910MQ mobile workstation.

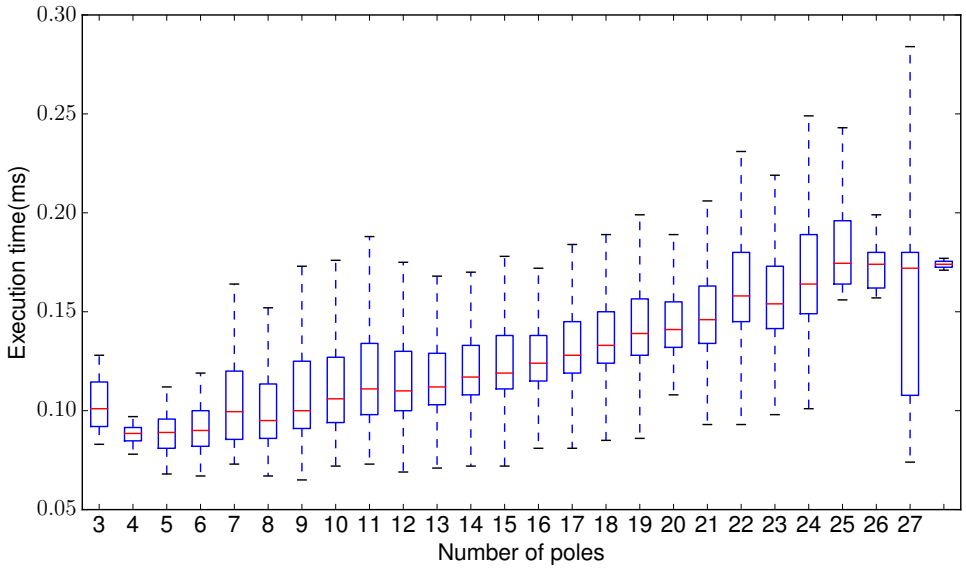


Figure 5.27: Boxplot of the execution time versus the number of poles.

5.3 Real On-Road Tests

To evaluate the proposed method in real urban scenarios, many field tests have been conducted in the test areas of Thielallee and Englerallee (see Figure 5.28). Using previously introduced localization method, the vehicle was driven autonomously on

the roads with minimal human intervention. And it drove along with other traffic participants. The interventions occurred when we had to pause the vehicle in case of heavy traffic, while making U-turns. But the necessary interventions didn't have much influence on the localization performance.

Figure 5.28: Real road autonomous test in the test area of Englerallee.

The real on-road tests did validate the effectiveness of the proposed localization method. And they achieved the same order of accuracy and precision as the evaluation based on datasets. For instance, the localization method can get similar trajectories for all laps. That means the proposed localization method has high repeatability. In comparison to the Applanix POS LV, our localization method is more accurate and precise. Even when Real Time Kinematic (RTK) correction is available, the Applanix sometimes still presents up to several meters drift. In contrast, the proposed method presents much lower uncertainty in position.

Besides, maps created based on the online data provided by the FIS-Broker (see section 4.3.1) were tested in the previously mentioned test areas as well. Although such maps are less accurate than those created through the log files of the test drive, they are accurate enough for supporting autonomous driving. The main problem for these maps is that the uncertainty of each feature is different. The data in some areas is much more accurate than other areas. As a result, the localization provides much smoother results in areas with much accurate data. However, this way has its superior advantages, which arrows to create a city scale map in a cheaper, easier and faster way.

5.4 Discussion

The results presented in the previous section have successfully evaluated the effectiveness of the proposed approach to localize the test autonomous vehicle by employing pole-like features. The proposed algorithm can deliver an RMS error in centimeter range. And the RMS accuracy of heading estimation errors is within ± 0.4 degree.

However, this method is not without problems. First, the Applanix cannot provide consistent value when driving around the same place several times. The ground truth is, therefore, not consistent for direct comparison. That is why the performance on the Englerallee dataset is worse. Second, the Velodyne feature extraction algorithm is not robust enough to get rid of surplus features, like pedestrians and planes of moving vehicles. These wrong features can add a burden to the calculation and deteriorate the performance.

To improve the performance, the existing problems should be solved. First, the corrected ground truth rather than raw data from Applanix should be utilized for both mapping and evaluation. Second, a more robust feature extraction algorithm is needed in the future. Or adding a camera for assisting the feature detection and extraction is another way to go.

5.5 Summary

In this chapter, the performance of the proposed localization algorithm was presented. It confirmed the effectiveness of the approach. To do that, the benchmark and datasets for accessing the performance were introduced in the beginning. Then, evaluation results based on datasets and real on-road tests were provided. Finally, a brief discussion was made.

"Always look on the bright side of life."

Monty Python

6 Conclusion and Future Work

Autonomous driving technology is one of the most promising technologies, attracting intense research interest from both academia and industry. Fully autonomous driving is within the realm of possibility. In the near future, autonomous driving will allow people to have much safer and more convenient driving experience. Following this trend, the focus of this thesis is to solve the fundamental problem of autonomous driving, localization. A feature-based localization method is proposed with the aim of replacing the currently used expensive INS/GPS system, which is four times more expensive than the vehicle. This method is based on the pre-built feature maps and online detection and matching features for localization. The performance of the proposed localization algorithm is evaluated through two datasets, and real on-road tests as well. The results indicate that it is comparable to the expensive system.

This chapter intends to finish the thesis by summarizing the previous chapters, providing a retrospective view on localization, and giving an outlook on future work.

6.1 Conclusion

As an outdoor wheeled robot, an autonomous vehicle should always have an accurate knowledge of its pose within an environment. This is because any high-level tasks that an autonomous vehicle performs need such knowledge. Thus, the main target of this work has concentrated on the accurate localization for autonomous vehicles by using fewer and cheaper or just on-board sensors to develop a robust alternative to the most expensive currently used Applanix POS LV 510. In the first two chapters of the thesis, the motivation and related work have been introduced,

intending to help readers to have a better grasp of related areas. Especially, chapter 2 presented an overview of state-of-the-art LIDAR-based localization approaches for autonomous vehicles, reviewing the different sensor setups and experimental results. These approaches highlighted the common practice in the community that localizing autonomous vehicles heavily relies on the pre-built lane-level maps at high levels of precision. Such maps are obtained from highly precise survey vehicles and through off-line processes, such as GraphSLAM.

Sensors play important roles in enabling the capabilities of autonomous driving, as they serve as eyes and ears to them. Therefore, sensor set-up was introduced in chapter 3. The overall sensor set-up for our test vehicle was introduced first. Then sensors used in this work were introduced thoroughly to some extent, especially their application challenges encountered in this work.

Chapter 4 and chapter 5 are the main parts in this thesis. Chapter 4 proposed a feature map based localization scheme under the EKF framework. It was involved in defining several coordinates, constructing the feature map, modeling the motion model and measurement update. The performance of the proposed approach was presented in chapter 5. The results indicate that it is comparable to the most expensive and fundamental system equipped on the vehicle. Therefore, this thesis contributes to strengthening the idea that it is possible to substitute the expensive localization systems by cheaper ones.

6.2 Contributions

The main innovations and contributions of this thesis can be summarized in three aspects.

(**A**) An innovative two-point localization scheme is proposed. It greatly mitigates the influence of the wrong feature matching during the data association stage. Thus, it gives much more precise estimation.

(**B**) A new trajectory smoothing method that is based on the Ackermann constraint is proposed. The proposed method can ensure much smoother trajectories especially during U-turns.

(**C**) The idea of using online data to create feature maps is also evaluated in this thesis and tested on the real roads. It is less accurate than the method of using the log files to create feature map. But it can create city scale feature maps in a more efficient and convenient way.

6.3 Future Work

Although the proposed method is proven to be effective, there is still room for improvements and refinements. For instance, to increase the accuracy of the localization, one should look into improving the results of the reference platform. So a couple of expected improvements are summarized as follows:

1. **Better Reference System**

Though Applanix POS LV 510 can produce more accurate location estimation than other low cost solutions, it cannot provide consistent estimation. This leads to two unexpected results. First, using the readings of Applanix as ground truth for the purposes of comparison may not reflect the real performance of the proposed approach. Second, building maps based on Applanix presents uncertainties, which affects the performance of localization. Therefore, a correction work should be done with the reference.

2. **Robust Feature Detection Solution**

Currently features are only extracted from 3D Velodyne point clouds. However, the Velodyne feature extraction algorithm is not robust enough to get rid of surplus features, like pedestrians and planes of moving vehicles. These surplus features can add burden to the calculation and worsen the performance. Therefore, a more robust feature extraction algorithm is needed in the future. Maybe it is wise to resort to computer vision for removing surplus features.

3. **Smart Map Management**

In the present work, each map consists of several hundred of features. In such scale, map storage and management are straightforward. As the map size increases, a naive map management algorithm turns out to be inefficient and an efficient method is needed. For example, applying the binary tree is a good choice for map storage and querying. Designing a dynamic map management algorithm is another way to go. Such maps can maintain parts of maps in the memory according to the vehicle's location. In this approach, the vehicle can fulfill a city scale driving.

Bibliography

[1] Ronald K Jurgen. Autonomous vehicles for safer driving. *Training*, pages 11–20, 2007. (Cited on page 2.)

[2] World Health Organization. Road traffic injuries, May 2016. http://www.who.int/violence_injury_prevention/road_traffic/en/. (Cited on page 2.)

[3] The Guardian. How autonomous cars are driving the city of the future, April 2016. https://www.theguardian.com/media-network/2016/apr/20/autonomous-driverless-cars-how-cities-might-look-vehicles. (Cited on page 2.)

[4] Miao Wang. *A Cognitive Navigation Approach for Autonomous Vehicles*. Mensch & Buch, 2012. (Cited on pages 2, 5, 7, 13, 14, 29, 30, 31 and 68.)

[5] Ruth Eckdish Knack. Pay as you parkpay as you parkpay as you park, May 2005. http://shoup.bol.ucla.edu/PayAsYouPark.htm. (Cited on page 3.)

[6] Germany Trade & Invest. Germany – the world's automotive hub of innovation, May 2016. http://www.gtai.de/GTAI/Navigation/EN/Invest/Industries/Mobility/automotive,t=industry--market-numbers,did=247736.html. (Cited on page 3.)

[7] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7,3:376–382, 1991. (Cited on pages 4 and 15.)

[8] Armin Hornung, Stefan Oßwald, Daniel Maier, and Maren Bennewitz. Monte carlo localization for humanoid robot navigation in complex indoor environments. *International Journal of Humanoid Robotics*, 11(02):1441002, 2014. (Cited on page 4.)

[9] Jesse Levinson and Sebastian Thrun. Map-based precision vehicle localization in urban environments. *Robotics: Science and Systems*, 2007. (Cited on pages 4 and 27.)

[10] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. (Cited on page 4.)

[11] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008. (Cited on page 4.)

[12] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: the ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):598–614, 2008. (Cited on page 4.)

[13] Tomislav Kos, Ivan Markezic, and Josip Pokrajcic. Effects of multipath reception on gps positioning performance. In *Elmar, 2010 Proceedings*, pages 399–402. IEEE, 2010. (Cited on page 4.)

[14] Wikipedia. History of autonomous car, September 2015. https://en.wikipedia.org/wiki/History_of_autonomous_car. (Cited on page 5.)

[15] DARPA. Urban challenge, September 2015. http://archive.darpa.mil/grandchallenge/. (Cited on page 6.)

[16] Google. On the road, September 2015. https://www.google.com/selfdrivingcar/where/. (Cited on page 7.)

[17] Freie Universität Berlin. Autonomous car of freie universität berlin travels across mexico car reaches mexico city after 2400 kilometers, October 2015. http://www.fu-berlin.de/en/presse/informationen/fup/2015/fup_15_319-autonom-in-mexiko/index.html/. (Cited on page 8.)

[18] Bennet Fischer, TinoschGanjineh, DanielGöhring, Sebastian Hempel, Tobias Langner, David Latotzky, Arturo Reuschenbach, Michael Schnürmacher, Ernesto Tapia, Fritz Ulbrich, Vitali Schauermann, Miao Wang, Fabian Wiesel, Till Zoppke, and Raul Rojas. Technical report autonomos 2010. Technical report, AutoNOMOS Labs, 2011. (Cited on pages 9, 28, 30 and 32.)

[19] Velodyne LiDAR Inc. *Velodyne HDL 64E User's Manual*, 2011. AvalibleOnline:http://www.velodynelidar.com/lidar/products/manual/63-HDL64ES2Manual_RevD_2011_web.pdf. (Cited on page 9.)

[20] ZJ 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 *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1554–1559. IEEE, 2013. (Cited on page 12.)

[21] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. (Cited on pages 12, 13, 14, 20, 35, 56, 57 and 63.)

[22] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and JuanManuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015. (Cited on pages 12 and 13.)

[23] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. (Cited on page 12.)

[24] Juan Nieto, Jose Guivant, and Eduardo Nebot. A novel hybrid map representation for denseslam in unstructured large environments. Technical report, Australian Center for Field Robotics, Department of Mechanical and Mechatronic Engineering, The University of Sydney, 2003. (Cited on pages 12 and 13.)

[25] DARPA Urban Challenge. Route network definition file (RNDF) and mission data file (MDF) formats. Technical report, Defense Advanced Research Projects Agency, March 2007. (Cited on page 13.)

[26] P. Misra and P. Enge. *Global positioning system: signals, measurements, and performance*. Ganga-Jamuna Press, 2011. (Cited on page 14.)

[27] A.D. Helfrick. *Electronics In The Evolution Of Flight*. Centennial of flight series. Texas A & M University Press, 2004. (Cited on page 14.)

[28] Clark F Olson. Selecting landmarks for localization in natural terrain. *Autonomous Robots*, 12(2):201–210, 2002. (Cited on page 15.)

[29] Ullrich Scheunert, Heiko Cramer, and Gerd Wanielik. Precise vehicle localization using multiple sensors and natural landmarks. In *Proceedings of the Seventh International Conference on Information Fusion*, pages 649–656, 2004. (Cited on pages 15 and 16.)

[30] Davide Ronzoni, Roberto Olmi, Cristian Secchi, and Cesare Fantuzzi. Agv global localization using indistinguishable artificial landmarks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 287–292. IEEE, 2011. (Cited on page 15.)

[31] S Hofmann, C Kuntzsch, MJ Schulze, D Eggert, and M Sester. Accuracy analysis of a low-cost platform for positioning and navigation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:1–6, 2012. (Cited on page 15.)

[32] Applanix. *POS LS Specification*, 2012. http://www.applanix.com/media/downloads/products/specs/poslv_specifications.pdf. (Cited on pages 16, 29 and 68.)

[33] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008. (Cited on pages 16 and 29.)

[34] Thorsten Weiss, Nico Kaempchen, and Klaus Dietmayer. Precise ego-localization in urban areas using laserscanner and high accuracy feature maps. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 284–289. IEEE, 2005. (Cited on page 16.)

[35] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*, volume 4, page 1. Citeseer, 2007. (Cited on page 17.)

[36] J.S. Levinson, S. Thrun, D. Koller, M. Levoy, and Stanford University. Dept. of Computer Science. *Automatic Laser Calibration, Mapping, and Localization for Autonomous Vehicles*. Stanford University, 2011. (Cited on page 17.)

[37] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4372–4378. IEEE, 2010. (Cited on page 17.)

[38] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 176–183. IEEE, 2014. (Cited on page 17.)

[39] Ryan W Wolcott and Ryan M Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2814–2821. IEEE, 2015. (Cited on page 17.)

[40] Claus Brenner. Global localization of vehicles using local pole patterns. In *Pattern Recognition*, pages 61–70. Springer, 2009. (Cited on pages 17 and 68.)

[41] Claus Brenner. Extraction of features from mobile laser scanning data for future driver assistance systems. In *Advances in GIScience*, pages 25–42. Springer, 2009. (Cited on page 17.)

[42] Claus Brenner. Vehicle localization using landmarks obtained by a lidar mobile mapping system. *Int. Arch. Photogramm. Remote Sens*, 38:139–144, 2010. (Cited on page 17.)

[43] Jan-Henrik Haunert and Claus Brenner. Vehicle localization by matching triangulated point patterns. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–351. ACM, 2009. (Cited on pages 17 and 62.)

[44] Alexander Schlichting and Claus Brenner. Localization using automotive laser scanners and local pattern matching. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 414–419. IEEE, 2014. (Cited on pages 17, 18 and 68.)

[45] Ning Liu, Yilong Yin, and Hongwei Zhang. A fingerprint matching algorithm based on delaunay triangulation net. In *Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on*, pages 591–595. IEEE, 2005. (Cited on page 18.)

[46] Ian Baldwin and Paul Newman. Road vehicle localization with 2d push-broom lidar and 3d priors. In *Robotics and automation (ICRA), 2012 IEEE international conference on*, pages 2611–2617. IEEE, 2012. (Cited on pages 18 and 68.)

[47] Ian Baldwin and Paul Newman. Laser-only road-vehicle localization with dual 2d push-broom lidars and 3d priors. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2490–2497. IEEE, 2012. (Cited on page 18.)

[48] Winston Churchill and Paul Newman. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4525–4532. IEEE, 2012. (Cited on page 18.)

[49] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013. (Cited on page 18.)

[50] Will Maddern, Geoffrey Pascoe, and Paul Newman. Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1684–1691, Seattle, WA, USA, May 2015. IEEE. (Cited on page 18.)

[51] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014. (Cited on page 20.)

[52] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. (Cited on page 20.)

[53] Ramsey Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation. *IEEE Signal processing magazine*, 29(5):128–132, 2012. (Cited on page 20.)

[54] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997. (Cited on pages 22 and 23.)

[55] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric Wan. The unscented particle filter. In *NIPS*, pages 584–590, 2000. (Cited on page 22.)

[56] Eric Wan, Ronell Van Der Merwe, et al. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000. (Cited on page 23.)

[57] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, Oregon Health & Science University, 2004. (Cited on pages 23 and 24.)

[58] RR Labbe. Kalman and bayesian filters in python, 2015. (Cited on page 24.)

[59] Seongkeun Park, Jae Pil Hwang, Euntai Kim, and Hyung-Jin Kang. A new evolutionary particle filter for the prevention of sample impoverishment. *IEEE Transactions on Evolutionary Computation*, 13(4):801–809, 2009. (Cited on page 25.)

[60] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002. (Cited on page 25.)

[61] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009. (Cited on page 25.)

[62] Kay Ch Fuerstenberg, Klaus CJ Dietmayer, and Volker Willhoeft. Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 31–35. IEEE, 2002. (Cited on page 30.)

[63] Herman Bruyninckx. Open robot control software: the orocos project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2523–2528. IEEE, 2001. (Cited on page 31.)

[64] N. Muhammad. *Contributions to the Use of 3D Lidars for Autonomous Navigation: Calibration and Qualitative Localization*. INSA, 2012. (Cited on pages 33 and 34.)

[65] Annalisa Milella and Giulio Reina. Rough terrain mobile robot localization using stereovision. In *ASME 2007 International Mechanical Engineering Congress and Exposition*, pages 1175–1181. American Society of Mechanical Engineers, 2007. (Cited on page 35.)

[66] PA Simionescu and D Beale. Optimum synthesis of the four-bar function generator in its symmetric embodiment: the ackermann steering linkage. *Mechanism and Machine Theory*, 37(12):1487–1504, 2002. (Cited on page 35.)

[67] Philipp Woock, Frank Pagel, Michael Grinberg, and Dieter Willersinn. Odometry-based structure from motion. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 1112–1117. IEEE, 2007. (Cited on page 35.)

[68] Niko Sünderhauf, Kurt Konolidge, Thomas Lemaire, and Simon Lacroix. Comparison of stereovision odometry approaches. In *Workshop Planetary Rovers, IEEE International Conference on Robotics and Automation (ICRA05)*. Citeseer, 2005. (Cited on page 35.)

[69] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y Ng, and Sebastian Thrun. Discriminative training of kalman filters. In *Robotics: Science and Systems*, pages 289–296, 2005. (Cited on page 35.)

[70] Robert Bosch GmbH. How does esp work, January 2015. http://products.bosch-mobility-solutions.com/en/de/specials/specials_safety/bosch_esp_3/esp__facts_4/esp_technik_2/esp_questions_and_answers_16.html. (Cited on page 35.)

[71] Clément Fouque, Philippe Bonnifait, and David Bétaille. Enhancement of global vehicle localization using navigable road maps and dead-reckoning. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 1286–1291. IEEE, 2008. (Cited on page 36.)

[72] ST Microelectronics. Everything about stmicroelectronics'3-axis digital mems gyroscopes, ta0343, technical article. *ST Microelectronics. July*, 2011. (Cited on page 36.)

[73] Samuel Picton Drake. *Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)*. Technical note. DSTO Electronics and Surveillance Research Laboratory, April 2002. (Cited on page 44.)

[74] Sebastian Hempel. Geometrierekonstruktion von lidar-punktwolken zur echtzeitverarbeitung für autonome fahrzeuge. Master's thesis, Free University of Berlin, 2009. (Cited on page 47.)

[75] Senate Department for Urban Development and the Environment. City trees- overview of the stock data, December 2016. http://www.stadtentwicklung.berlin.de/umwelt/stadtgruen/stadtbaeume/en/daten_fakten/uebersichten/index.shtml. (Cited on page 51.)

[76] Senate Department for Urban Development and the Environment. City trees-overview of the stock data, December 2016. http://www.stadtentwicklung.berlin.de/bauen/beleuchtung/de/zustaendigkeiten.shtml. (Cited on page 51.)

[77] Environmental Systems Research Institute. About coordinate systems and map projections, December 2016. http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=About_coordinate_systems_and_map_projections. (Cited on page 51.)

[78] Samuel Thomas Pfister. *Algorithms for mobile robot localization and mapping, incorporating detailed noise modeling and multi-scale feature extraction*. PhD thesis, California Institute of Technology, 2006. (Cited on page 52.)

[79] Ayoung Kim. Active visual slam with exploration for autonomous underwater navigation. Technical report, DTIC Document, 2012. (Cited on page 56.)

[80] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890–897, 2001. (Cited on page 57.)

[81] W Murphy and Willy Hereman. Determination of a position in three dimensions using trilateration and approximate distances. *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95*, 7:19, 1995. (Cited on page 62.)

[82] Dustin Lang et al. Blind astrometric calibration of arbitrary astronomical images. *Astrometry. net*, pages 1–55, 2009. (Cited on page 62.)

[83] Andriy Myronenko, Xubo Song, and Miguel A Carreira-Perpinán. Non-rigid point set registration: Coherent point drift. In *Advances in Neural Information Processing Systems*, pages 1009–1016, 2006. (Cited on page 62.)

[84] Evgeni Kiriy and Martin Buehler. Three-state extended kalman filter for mobile robot localization. *McGill University., Montreal, Canada, Tech. Rep. TR-CIM*, 5, 2002. (Cited on page 63.)

[85] Oliver Wulf, Andreas Nüchter, Joachim Hertzberg, and Bernardo Wagner. Ground truth evaluation of large urban 6d slam. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 650–657. IEEE, 2007. (Cited on pages 67 and 68.)

[86] Zhenning Tao, Philippe Bonnifait, Vincent Fremont, and Javier Ibanez-Guzman. Mapping and localization using gps, lane markings and proprioceptive sensors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 406–412. IEEE, 2013. (Cited on page 68.)

[87] Will Maddern, Alexander D Stewart, and Paul Newman. Laps-ii: 6-dof day and night visual localisation with prior 3d structure for autonomous road vehicles. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 330–337. IEEE, 2014. (Cited on page 68.)

[88] Albert S Huang. *Lane estimation for autonomous vehicles using vision and lidar*. PhD thesis, Massachusetts Institute of Technology, 2010. (Cited on page 68.)

[89] André Ibisch, Stefan Stumper, Harald Altinger, Marcel Neuhausen, Marc Tschentscher, Marc Schlipsing, Jan Salinen, and Aaron Knoll. Towards autonomous driving in a parking garage: Vehicle localization and tracking using environment-embedded lidar sensors. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 829–834. IEEE, 2013. (Cited on page 68.)

[90] Robert Spangenberg. *Landmark-based Localization for Autonomous Vehicles*. PhD thesis, Freie Universität Berlin, 2016. (Cited on pages 84 and 91.)

[91] Pierre Roduit. *Trajectory analysis using point distribution models: algorithms, performance evaluation, and experimental validation using mobile robots*. PhD thesis, Citeseer, 2009. (Cited on page 91.)

[92] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*, pages 13–22. Australian Computer Society, Inc., 2013. (Cited on page 91.)

A

Zusammenfassung

Aufgrund vielversprechender Vorteile werden autonome Fahrzeuge als die Zukunft des Straßenverkehrs angesehen. Es ist zu erwarten, dass sie in absehbarer Zeit zum Alltag gehören werden. Da autonome Fahrzeuge viel sicherer fahren als menschliche Fahrer und deutlich weniger CO2-Emissionen erzeugen, werden sie zu einem neuen Trend im akademischen und industriellen Bereich. Obwohl die damit verbundenen Technologien seit Jahrzehnten erforscht und entwickelt wurden, müssen noch einige Hindernisse überwunden werden. Eines dieser großen Hindernisse sind die Kosten der erforderlichen Sensoren. Daher wäre die Verwendung von weniger teuren Geräten eine langfristige und effektive Lösung.

Der Schwerpunkt dieser Dissertation liegt auf dem Entwurf und der Implementierung eines featurebasierten Lokalisierungsalgorithmus, der den teuersten Teil des derzeitigen autonomen Testfahrzeugs, ein inertiales DGPS-Navigationssystem (Applanix POS LV 510) robust ersetzen kann. Eine entscheidende Implementierung verwendet, unter Zuhilfenahme von Velodyne LIDAR, Gyroskop, Raddrehzahlsensoren, den oft genutzten erweiterten Kalmanfilter (EKF), um das Fahrzeug zu lokalisieren. Das Velodyne LIDAR wird verwendet, um pfahlartige Strukturen aus der zuvor abgebildeten Umgebung zu extrahieren. Gyroskop und Raddrehzahlsensoren werden verwendet, um eine relative Lokalisierung durchzuführen, die hauptsächlich während der Bewegungsprädiktion des EKF verwendet wird. Ein Vorteil dieser Methode ist, dass sie keine GPS-Informationen nach der Initialisierung benötigt, wodurch sie genauer und robuster ist als eine GPS-Lösung. Die Leistungen des vorgeschlagenen Lokalisierungsverfahrens werden durch zwei Datensätze evaluiert. Die Ergebnisse zeigen, dass es mit dem INS / DGPS-System vergleichbar ist. Die echten On-Road-Tests in städtischen Szenarien verifizierten auch die Wirksamkeit und Robustheit des vorgeschlagenen Lokalisierungssystems.

B

About the Author

Since September 2012, Xiuyan Guo has been studying and working in the AutoNOMOS group as a PhD candidate, financially supported by the China Scholarship Council. His work concentrates on SLAM and localization problems for autonomous vehicles. Before switching to computer science at the Free University of Berlin, he studied electrical engineering in China, gaining rich experience in hardware design and micro-controller and FPGA related programming. He received the B.E. degree in electronic and information engineering, and the M.E. degree in signal and information processing from Northwestern Polytechnical University, Xi'an, China, in 2008 and 2011, respectively.