Precise self-localization of autonomous vehicles using lidar sensors and highly accurate digital maps on highway roads

Farouk Ghallabi

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Precise Self-Localization of autonomous vehicles using lidar sensors and highly accurate digital maps on Highway roads

Localisation précise d'un véhicule autonome en utilisant des télémètres lasers et une carte précise de l'environnement sur autoroutes

Soutenue par

Farouk GHALLABI

Le 30 Juin 2020

Ecole doctorale n° 621

Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité

Informatique temps réel, robotique et automatique

Composition du jury :

M. Philippe Bonnifait Professeur , Université de Technologie de Compiègne.	*Président*
M. Vincent Frémont Professeur, Ecole Centrale de Nantes.	*Rapporteur*
M. José Eugenio Naranjo Professeur, Universidad Polytécnica de Madrid.	*Rapporteur*
M. Paul Honeine Professeur, Université de Rouen.	*Examinateur*
Mrs Marie-Anne Mittet PhD, Groupe Renault.	*Examinateur*
M. Ghayath El-Haj-Shhadeh PhD, Groupe Renault.	*Examinateur*
M. Fawzi, Nashashibi Professeur, INRIA.	*Directeur de thèse*

MINES ParisTech | PSL

Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgements

I would like to express my sincere gratitude to my PhD director Prof. Nashashibi Fawzi from INRIA RITS team and my advisors Dr. ELHaj-shhade Ghayath and Dr. Mittet Marie-Anne from Groupe Renault for the continuous support of my Ph.D study and related research, for their patience, motivation. Their support helped me in all the time of research and writing of this thesis.

I would like to thank my family: my wife, my daughters Khadija and Fatima and my parents for supporting me spiritually throughout writing this thesis and my life in general. I pray God to bless them and to protect them.

Chapter 1

General introduction

Contents

Résumé en français

Dans ce premier chapitre, nous présentons le contexte général du développement des véhicules autonomes ainsi que les différents enjeux et difficultés rencontrés dans ce domaine. Nous commençons par une petite revue historique sur le développement des robots autonomes, puis, nous abordons les différents standard et architectures utilisés dans la conduite autonome.

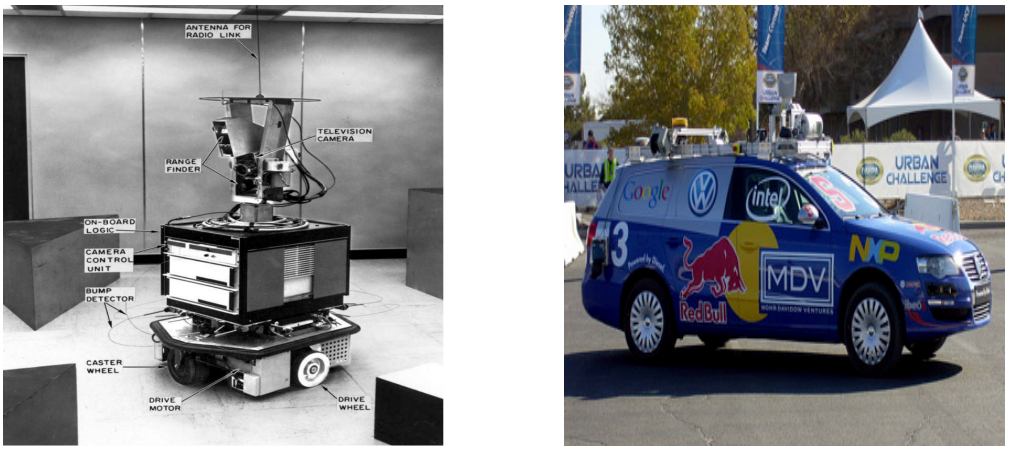
1.1 Context and challenges

The development of self-driving cars has recently gained interest in the scientific research community and the industrial world. The main motivation is to increase the occupant safety and to reduce car crashes as much as possible. Indeed, the National Highway Traffic Safety Administration (NHTSA) reported in 2016 that 94 percent of deadly car crashes on US roads are caused by human error [NHTSA, 2016]. Although a level 5 fully autonomous vehicle does not exist at the moment, driver assistance systems have been effective in reducing fatal crash rates. NHTSA reported in 2018 that the overall fatalities due to car crashes have been reduced by 2.4 percent thanks to the introduction of new sensor-based technologies in recent car models [NHTSA, 2018]. Another motivation is the introduction of driverless robotaxis and shuttles as new business models for private and public transportations. This could have a positive impact on road safety, traffic congestion and parking. Since these services will most probably use electric vehicles, significant improvements could be expected for pollution and energy consumption.

Historically, autonomous mobile robots have been studied for more than 40 years. The first developed mobile robot was presented by the Stanford Research institute (SRI) between 1966 and 1972, named *Shakey* (Figure 1.1a). It was equipped with cameras, range finders and wheel odometers. In 1977, Tsukuba Mechanical Engineering Lab in Japan developed a driverless car capable of running at 30 [km/h] on a dedicated track by following white street marks. Later, in the 1980s, professor Ernst Dickmanns designed a self driving vehicle that achieved high speed driving on highways. Since then, many European projects addressed autonomous driving developments, for instance,

the PROMETEUS (1987-1995), ARGO (1996-2001), Cybercars (2001-2004, 2006-2008), Citymobil projects (2006-2011) and (2012-2016).

A first major step toward autonomous vehicles has been demonstrated in the DARPA Grand Challenge in 2005. The vehicles had to cross 200 Km of desert roads in fully autonomous mode at an average speed of 30 [km/h]. *Stanley*, the winner of the challenge, was created by Stanford University's Stanford Racing Team in cooperation with the Volkswagen Electronics Research Laboratory (ERL) (figure 1.1b). A second major step is the DARPA Grand Challenge in 2007 which addressed urban environments (DARPA URBAN CHALLENGE). This event required teams to build an autonomous vehicle capable of driving in traffic, performing complex maneuvers such as merging, passing, parking and negotiating intersections. The entry Tartan Racing of the Carnegie Mellon team crossed the finish line first.



(a) Shakey robot

(b) Stanely vehicle

Figure 1.1: Examples of autonomous mobile robots

In general, fully autonomous robots require three different modules to operate: perception, path planning and control (Figure 1.2). Perception continuously provides information about the surrounding environment as well as the location of the robot (i.e. robot localization). Environment perception includes: static and moving obstacle detection, segmentation, classification and tracking. Different sensor technologies can be used (e.g. Radar, camera, LiDARs and ultrasounds) and fused together (i.e. sensor data fusion). Localization estimates the position of the vehicle in the surrounding environment with respect to a reference coordinates system. Path planning consists in finding the appropriate path the robot has to follow in order to navigate from point A to point B while avoiding collisions with obstacles. The control module calculates the set of commands to be applied to follow the path generated by the planning. The cascade input-output relation presented in Figure 1.2 shows the importance of the perception module. Indeed, if something goes wrong in perception, path planning and control can be highly affected.

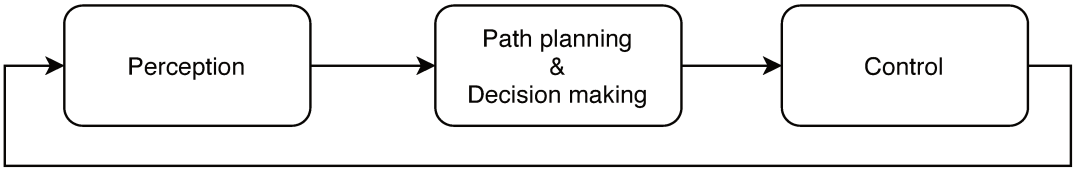


Figure 1.2: Fundamental components of automated driving

1.2 Current trends for automated driving

1.2.1 SAE Levels of automated driving

A lot of efforts have been put into defining standards to facilitate the development and industrialization of autonomous driving. The society of Automotive Engineers (SAE) defines six levels of driving automation ranging from 0 (fully manual) to 5 (fully autonomous). They vary depending on the degree of human involvement in the act of driving and are specified to assist the development and the commercialization of automated driving functions while meeting safety standards. Figure 1.3 summarizes the six levels and the corresponding features. From SAE level 0 to level 2, the human driver is still involved in the driving task. Starting from level 3, the driving task is delegated to the system. In level 3 the system can request a take over by the driver when certain conditions are not met, whereas in level 4 and 5 this is automatically handled by the system. Level 5 is a fully autonomous system that can drive under all conditions.

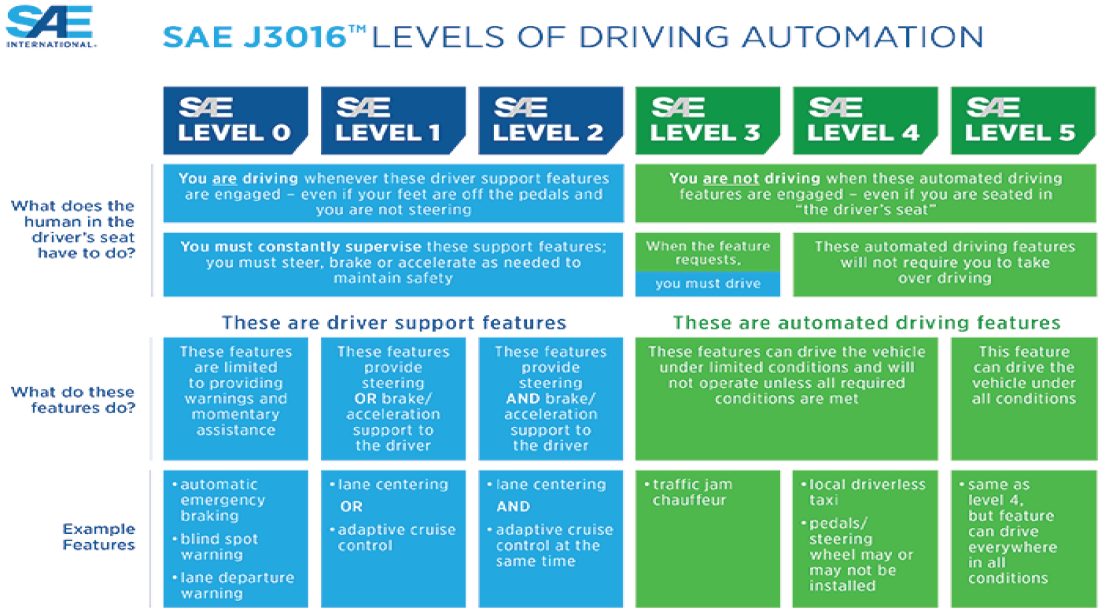


Figure 1.3: SAE levels of automated driving [SAE, 2020]

1.2.2 Automated driving requires multiple sensors

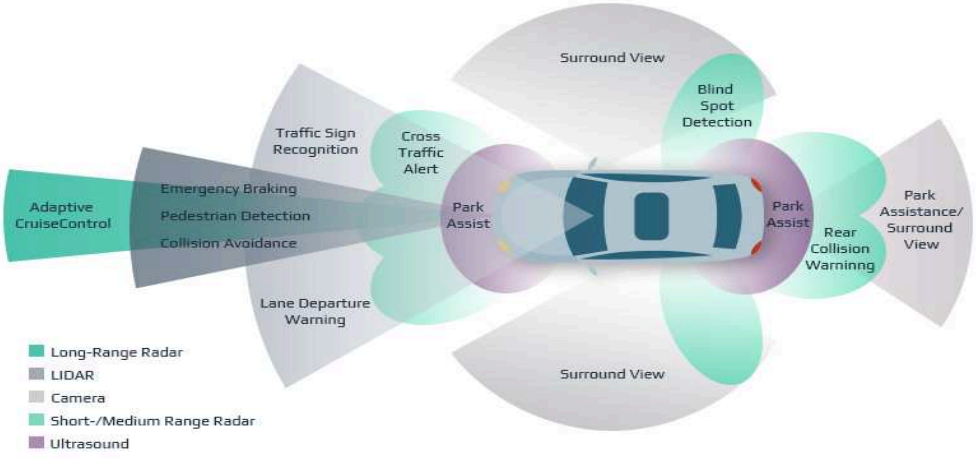


Figure 1.4: Examples of most common sensors used for ADAS and automated driving.

To ensure the driver's safety, the vehicle has to "see" and sense everything in the surrounding environment. Many sensor technologies have been used today in production cars such as radars, cameras, ultrasounds and LiDARs (Figure 1.4) to enable different features such as Adaptive Cruise Control (ACC), Parking Assist, Blind Spot Detection, Lane keeping assist (LKA), etc. Because each sensor has its pros and cons, multiple sensors are used to overcome limitations of each individual sensor. In the following paragraphs, we introduce concepts of Radar, Camera and LiDAR sensor technologies.

A Radar

Radar technology is widely used in automotive applications, as it is capable of providing distance, velocity and angular information of the targets of interest (car, truck, motorcycle, pedestrian etc.) even in inclement weather. The most popular radar-based driver assistance system is ACC (Adaptive Cruise Control). In addition to the vehicle speed, ACC maintains a safe separation distance from the car in front. To do that, radars use special waveforms like FMCW (frequency-modulated continuous wave). The Frequency Synthesizer generates the FMCW wave or chirp at the desired frequency. The wave is then amplified using a power amplifier. The transmit (TX) antenna converts the electrical energy of the signals to electromagnetic signals and transmits them through free space. The receive (RX) antenna receives the reflected signal (Figure 1.5) from the target and feeds it to the low-noise amplifier. Then, mixer down-converts the signal to base hand. Finally, the signal is converted from analog to digital so the signal processing can be performed to extract target attributes.

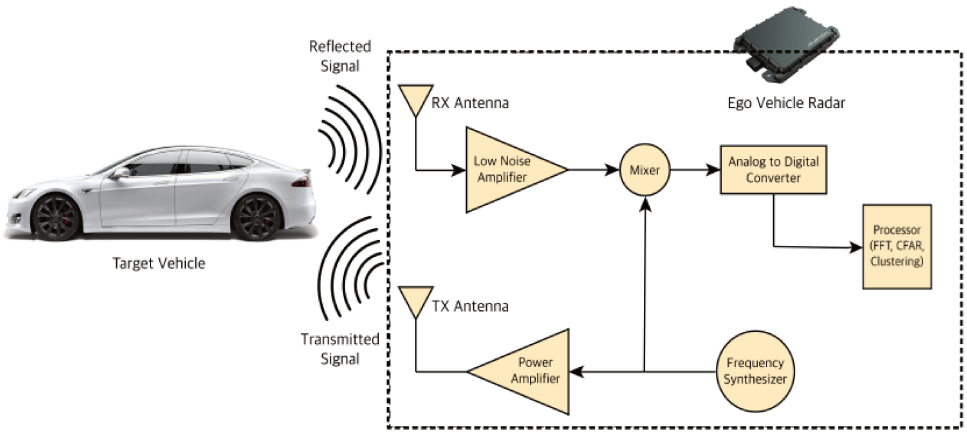


Figure 1.5: Radar pipeline.

B Camera

Camera is among the most used sensing technologies for perception. It offers interesting features such as shapes, textures, colors, etc. The major limitation is being a passive sensor that depends on external lighting conditions. Dealing with camera images is essentially doing geometry. Indeed, images are mapping functions of the 3D world into a 2D plane where each pixel value depicts the reflected light from an obstacle. Therefore, a major difficulty encountered in image processing is the ability to recover the 3D model from a set of 2D images. A typical mathematical model to describe the geometric projection of the 3D world into a 2D plane is the *pinhole camera model* where the camera is described as a point (figure 1.6a). Two different technologies are used in the development of vision perception systems: monocular and stereo vision.

- **Monocular vision** is the standard technology which uses only one calibrated camera. The recovery of the 3D world is not straightforward due to the challenge of scale estimation.
- **Stereo vision** uses two calibrated monocular cameras pointed in the same direction and separated by some distance known as baseline. With stereo vision, it is possible to estimate the depth of image features and therefore reconstruct the 3D model of the environment.

Using cameras for perception has been studied for many years and has shown outstanding results. The pixel size density of images allows to recover a variety of useful information for autonomous driving. In addition, image data is now used extensively with AI (Artifical Intelligence) algorithms to perform: detection, segmentation, classification, etc.

C LiDAR

LiDAR (Light Detection and Ranging) is an active sensor that measures the distance to an object by using the round-trip time of flight of laser light pulse. The time of flight is the time that laser light spends to travel a distance from a light emitter to an object then returns back to a receiver (i.e. phase shift in Figure 1.6b). Given the speed of light c and the time of flight tof, the distance of the emitter to the object is deduced as follows:

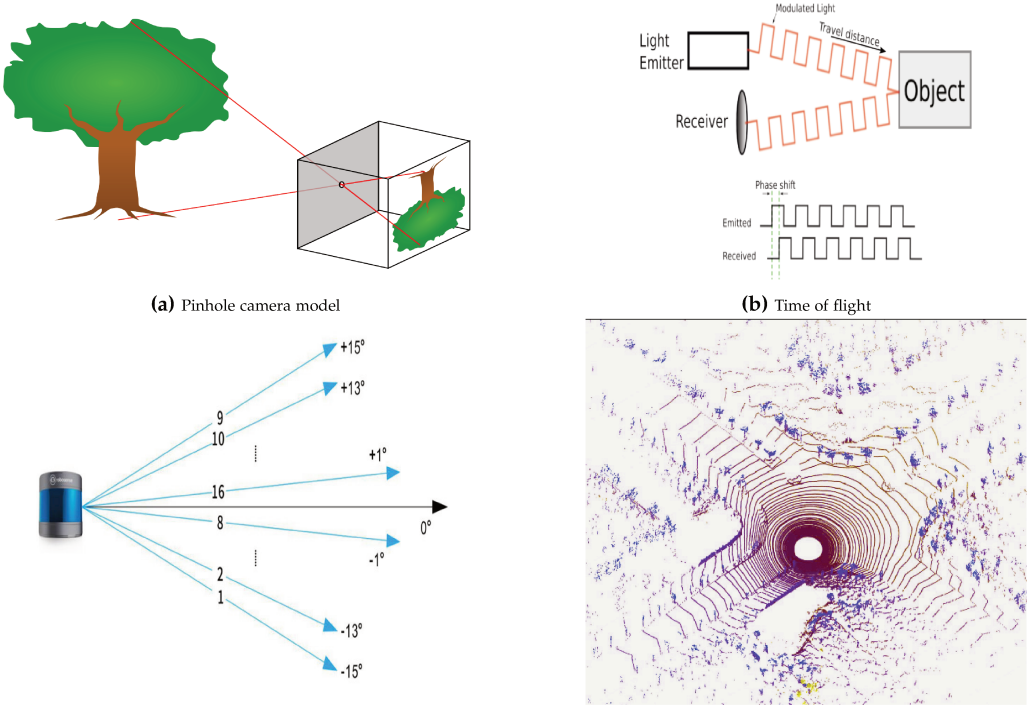
$$d = \frac{c \times tof}{2}$$

In addition range measures, LiDARs also provide a reflectivity (or intensity) data which is a measure, collected for every point, of the return strength of the laser pulse that generated the point. This information can be very relevant to characterize specific objects for which the intensity is very high (traffic signs, vehicle license plates, road paintings, etc). In robotics and automotive applications, LiDARs are designed to be eye safe. A rotating mirror is generally used to change the direction of the laser pulse allowing to cover a field of view up to 360 degrees horizontally. LiDAR providers use a multi-beam LiDAR which is an array of multiple beams (or layers) in order to generate a 3D point cloud (see Figure 1.6d). The main advantages of LiDAR sensors are the accuracy of the measured range distance (2-5 centimeters), detection range (up to 100-150 meters) and the resolution. However, laser beams are affected by rain and fog due to laser light scattering caused by water droplets. As a result, the generated point clouds may include empty regions without significant laser light returns.

Another typical problem for LiDAR sensors is the vertical beam resolution that is the vertical angle between two consecutive layers. For some sensors, such as the Velodyne VLP16, the vertical resolution may fall down to 2 degrees (figure 1.6c). The impact of such a resolution is seen in the density of the point cloud. For example, a resolution of 2 degrees is translated into a vertical distance of 1.7 m at 50 meters range.

D Comparison

Radar, LiDAR and Camera technologies are compared in Table 1.1. Radars are suitable for long-range detection (up to 300 m) and can directly measure range and relative radial velocity of the targets. They are robust against adverse weather and can be integrated in car bumper or fascia. The price of radar sensors has significantly decreased in recent years thanks to technology breakthrough (SiGe, RFCMOS) and integration capabilities. They are widely present today in production cars, to enable system features like ACC, TJA and AEB. However, Radars have limited capabilities of spatial (angular) resolution (i.e. separation of targets at the range and speed) and target classification. Cameras are also used in ADAS applications today, especially for features like Lane-Departure Warning (LDW) and Lane-Keeping Assist (LKA), and AEB as well (with or



(a) Pinhole camera model

(b) Time of flight

(c) Velodyne VLP16 vertical beam resolution

(d) Generated point cloud (HDL-64).

Figure 1.6: Illustration of camera pinhole model, LiDAR time of flight, LiDAR resolution and point cloud data.

without fusion with Radars). They offer many advantages such as resolution and richness of information (color, shapes, texture). In addition, the evolution of processing units has enabled the implementation of sophisticated AI-based algorithms for target detection and classification. However, being a passive sensor, Camera is sensitive to lighting conditions and poorly estimates depth data (monocular vision). Finally, LiDARs provide accurate 3D representation of the environment, with an accuracy of 2 of 5 cm. Compared to Radars, the resolution could be 30 times better, but the data is still sparse compared to Camera. The vertical resolution of the LiDAR depends on the number of layers (sources) and/or the scanning mechanism. The main challenge of LiDAR manufacturers today is to find the best compromise between those design parameters, in order to improve the cost of the sensor that is generally high. LiDAR is also sensitive to rain and fog, because laser pulses are scattered by water droplets.

Table 1.1: Characteristics of Radars, LiDARs and camera for automated driving. ++: ideally suited, +: good performance, m: medium, o: possible with additional effort, -: not suited

	Radars	LiDARs	Cameras
Direct range measurement	++	++	o - only stereo
Detection range	++ (300 meters)	+ (100-150 meters)	m (40-120 meters)
Accuracy	+	++	m
Resolution	m	+ [1]	++
Weather conditions[2]	++	+	+
Light sensitivity	++	++	-
Price for commercial car products	Low	High	Medium

[1] The resolution of 3D LiDARs depends on the number of layers. Compared to Radars, LIDARs have much better resolution, whereas compared to cameras, they still have low resolution.

[2] LiDARs and cameras are sensitive to rain and fog.

To conclude, there is not a single sensor that is suitable in all conditions. Instead, each sensor has its advantages and disadvantages. In some applications, some criteria may be privileged over others. For example, LiDARs and cameras have been extensively used for (but not limited to): localization, object detection & classification and mapping, whereas radars are suitable to detect metal objects even at longer ranges (300 m). Choosing a suitable sensor for a given application is not straightforward and should be carefully studied. In the following paragraph, we discuss the localization task which is the main focus of study of this thesis.

1.2.3 Localization for autonomous driving

Localization is a crucial perception task that focuses on estimating the position of the vehicle in the surrounding environment with respect to a reference coordinate system. Very often, we want to guarantee two essential properties of localization systems. *Localization accuracy* that consists in reducing metric errors of the estimation process and *localization integrity* that is defined as the measure of trust which can be placed in the correctness of the information supplied by the localization system. For localization accuracy, three different localization levels have been studied in [EDMap, 2004] (Figure 1.7):

- ***WHAT ROAD***: this level is provided by most navigation systems. It determines the driving road and requires an accuracy of 5 to 10 meters.
- ***WHICH LANE*** : this level determines the lane where the vehicle is and requires additional information such as the number of lanes. the required accuracy is around 1 m.
- ***WHERE IN LANE***: this level determines the position of the vehicle in the assigned lane. It requires a localization error of 0.3 meter.

In addition to sensors, High Definition (HD) maps have been extensively used to achieve accurate localization systems. These maps allow to understand the environment beyond sensor visibility (sensor range) and provide high accurate, up-to-date and realistic environment representations. High Definition (HD) maps for autonomous driving are currently being developed by many commercial maps providers like TomTom [Tom, 2019], Here [Her, 2019] (Figure 1.8) as well as free mapping projects like OpenStreeMap (OSM) [Haklay and Weber, 2008].

A Coordinate systems for localization

Geodetic coordinates are typically used for vehicle navigations. They provide a 3D position of the vehicle using two different angles: *longitude* and *latitude* from the center of Earth and altitude. This reference system mostly used by GNSS (Global Navigation Satellite Systems) is commonly known as the *WGS 84 geodetic reference system*. The geometric manipulation of position in WGS 84 can be tedious because it uses angles instead of Cartesian coordinates.

ECEF (Earth-Center Earth-Fixed) is a Cartesian coordinate system centered at the center of the Earth (Figure 1.9.a). This reference system is not well suitable for vehicle navigation because its center is too far from the surface of the Earth where the vehicles are moving.

ENU (East North Up) is a Cartesian coordinate system whose center is on the surface of the Earth. It assumes a surface section of the Earth to be flat. The three axes

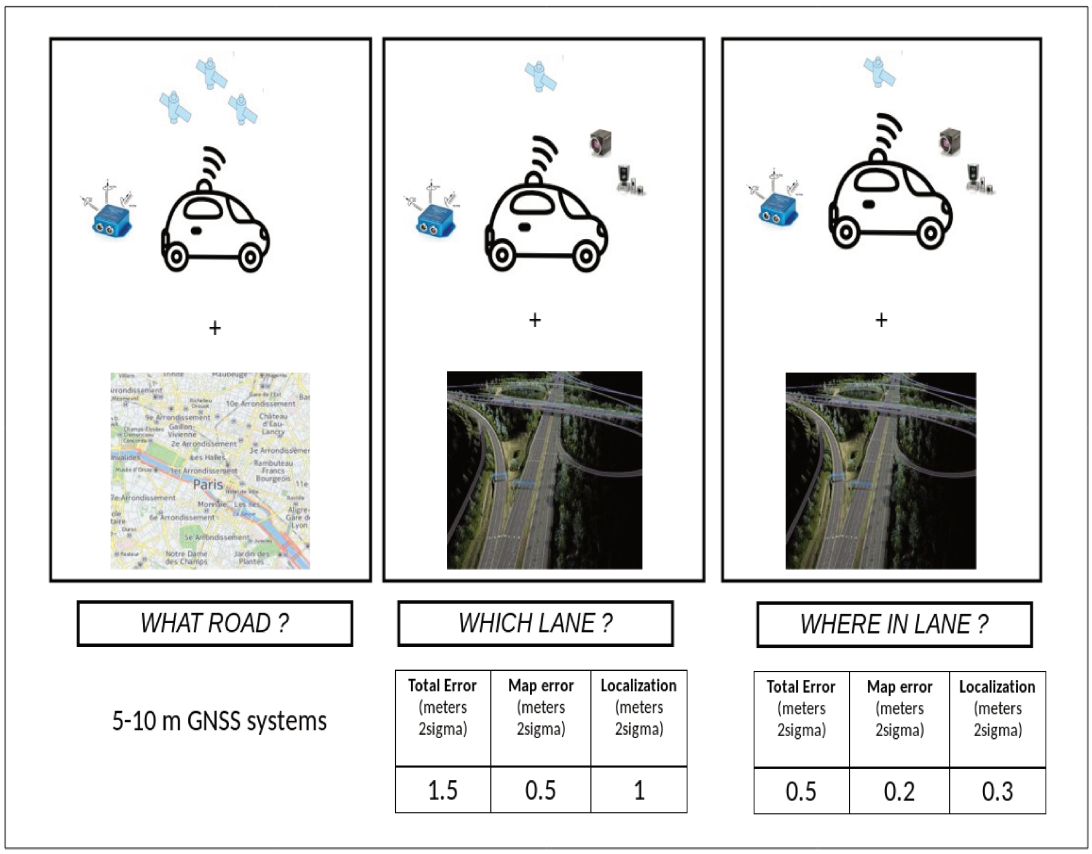
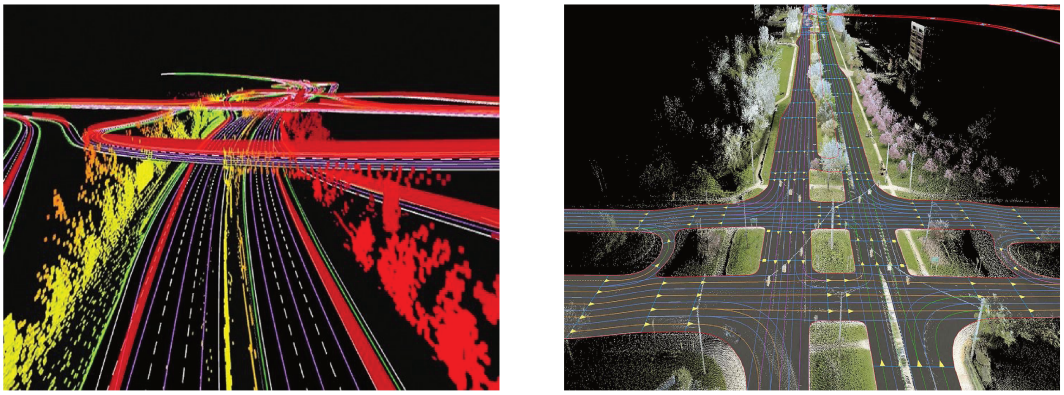


Figure 1.7: Localization levels as described in [EDMap, 2004]



(a) Example of TomTom HD map

(b) Example of Here HD map

Figure 1.8: TomTom and Here HD map representations

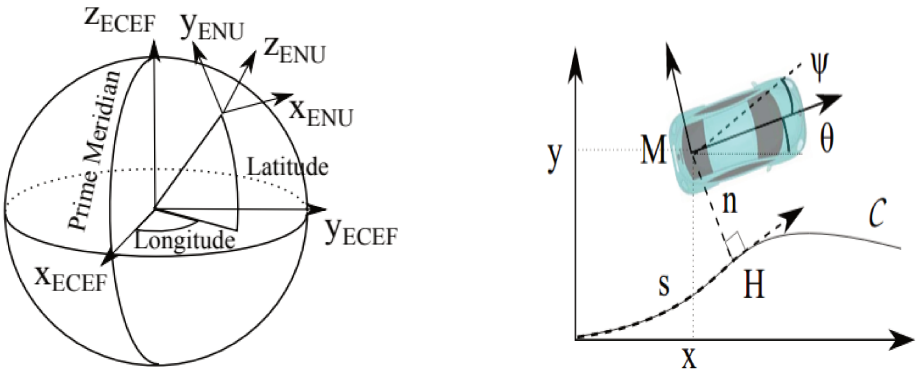


Figure 1.9: (Left) Illustration of ECEF and ENU coordinate systems. (Right) illustration of curvilinear coordinates. M : is the middle of rear axle point. H: is the projection of M to curve C, s and n are the curvilinear abscissa (along-track and cross-track). θ: the heading of the vehicle with respect to the x-y coordinates and ψ is the heading of the vehicle with respect to curve C. This image is taken from [Héry et al., 2018].

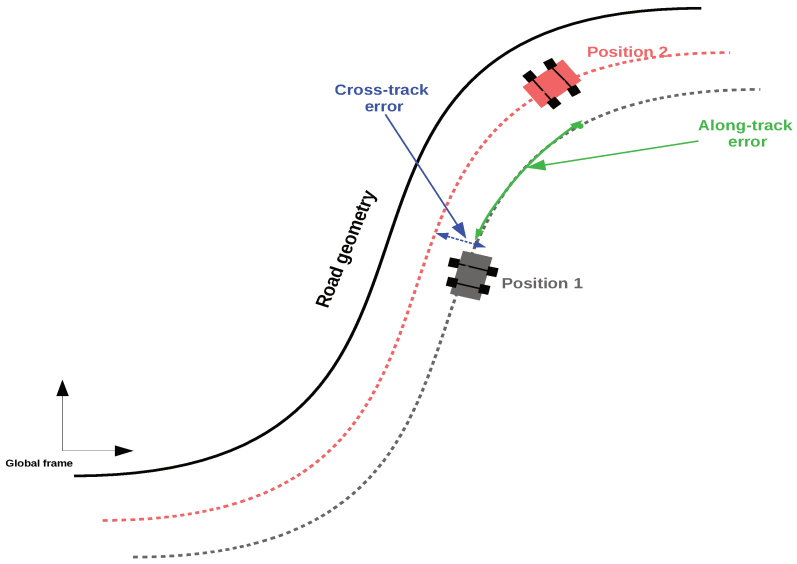


Figure 1.10: Illustrative example of the cross-track and the along-track errors. Position 1: current position and position 2 is true position

are aligned with the East, North and vertical directions and are tangent to the ellipsoid model of the Earth (Figure 1.9.a).

In automatic control, it is more convenient to know where the vehicle is with respect to the road. Hence, the curvilinear coordinate system is often used. The abscissa is defined along the center of the lane, the ordinate is the signed lateral distance and the heading which is the relative orientation with respect to the center of the lane (cf. Figure 1.9.b). Figure 1.10 depicts the calculation of cross-track and along-track errors assuming that we are given two different positions: Position 1 and position 2 and given road geometry. The calculation of these errors depends essentially on the road representation in the map (polyline, splines, polynomials). A good introduction to the subject is given in [Héry et al., 2018].

1.3 Objective of this thesis

The aim of this thesis is to develop a highly accurate localization system to enable highly automated driving functions (level 3 and above) on highway roads. The studied environment is very challenging. First, highway roads lack sufficient landmarks as often used in SLAM-based localization approaches. Second, because the vehicle speed is very high (up to 130 kph), problems related to data association and sensor frame rates are to be expected. Hence, we address the problem by defining a suitable architecture of the system, choosing the sensors (LiDAR and third party highly accurate map) and developing the required algorithm pipeline for perception and localization. Finally, we validate the developed solution by performing tests on test track and on a real highway road.

1.3.1 Main contributions

In this work, we develop a LiDAR-based localization approach using a highly accurate map. As previously discussed, LiDARs are suitable to get 3D accurate representation of the environment and they cover a large field of view. The main contributions of this work are summarized hereafter:

A LiDAR perception module

The proposed perception module uses LiDAR data to extract relevant features for localization. Feature detection process relies on shape-based and reflectance-based models. The perception module provides:

1. Detection and tracking of lane markings. The recognition of the type of the lane marking is not addressed.

2. Detection and tracking of Guardrails (on highways).

3. Detection of traffic signs. Here, only the locations of traffic signs are provided. Semantic recognition is not addressed.

4. Detection of guardrail reflectors. Indeed, on many highway roads, guard rails are usually equipped with reflecting markers to indicate their location in low light conditions.

B Improved localization module based on an optimized particle filter

The proposed localization module implements a particle filtering algorithm to localize the vehicle in a highly accurate digital map. The algorithm matches the data from the perception module with the data attributes in the map. The choice of the number of particles has an immediate effect on the convergence of the particle filter. If this number increases, more space is covered and the convergence of the filter is more ensured. However, increasing the number of particles results in a high computational resource and limits real time application. A key contribution of our work is the proposal of a modified version of particle filtering which keeps the number of particles constant during the experiments but re-distributes them in an optimized manner to maximize the particle space coverage around the true vehicle position. This version is called *constrained-update* particle filtering.

C Localization assessment under a variety of conditions

To validate the proposed solution, we collected data by driving a prototype vehicle in various conditions. At first, we conducted different experiments at different ego vehicle speeds from 30 kph to 110 kph with a step of 10 kph. Secondly, we collected data on a real highway road (A13 highway in Paris area) and on a Renault highway-like test track (CTA2) of 5 kilometers length. The collected data contains different driving maneuvers. Indeed, we have collected sequences with and without lane change maneuver. The evaluation was performed by comparing the outputs of our localization system to a ground truth provided by a GNSS/RTK receiver.

1.3.2 Structure of the manuscript

This manuscript is organized as follows. In chapter II, a state of the art review of different localization and mapping systems is given. In the mapping section, we describe different map representations in the literature then we highlight the current trends for autonomous vehicles. In the localization section, we discuss three different topics: local localization (or odometry) where we address two different sensors: camera and LiDAR sensors. In the global localization section, we focus on GNSS receivers and the map-based localization. Finally, we address the Simultaneous Localization And Mapping (SLAM) problem.

Chapter III details our proposed perception and localization solution. First, a literature review of different perception systems with a focus on cameras and LiDARs is given. Then, the proposed LiDAR-based road perception system is described. In addition, the Map Management System (MMS) which consists of a set of tools to communicate with the third party map is illustrated. A mathematical formulation of Markov localization is given in order to introduce to the particle filtering concept and to explain our implementation of the filter.

An experimental evaluation of the proposed system is discussed in chapter IV. This evaluation compares position outputs of the localization system with a ground truth, obtained by GNSS with RTK corrections. The evaluation of the perception system on its own is not addressed in this work since it is considered to be implicitly evaluated along with the localization system. Finally, conclusions and perspectives are given in chapter V.

1.4 List of publications

This work has led to a list of scientific publications in international IEEE conferences and a pending journal submission.

- **F. Ghallabi**, F. Nashashibi, G. El-Haj-Shhade, and M. A. Mittet, "LIDAR-Based Lane Marking Detection for Vehicle Positioning in an HD Map," in IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2018, vol. 2018-November, pp. 2209–2214.
- **F. Ghallabi**, M. A. Mittet, G. El-Haj-Shhade, and F. Nashashibi, "LIDAR-Based High Reflective Landmarks (HRL)s For Vehicle Localization in an HD Map," in 2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019, 2019, pp. 4412–4418.
- **F. Ghallabi**, G. El-Haj-Shhade, M. A. Mittet, and F. Nashashibi, "LIDAR-based road signs detection for vehicle localization in an HD map," in IEEE Intelligent Vehicles Symposium, Proceedings, 2019, vol. 2019-June, pp. 1484–1490.

Chapter 2

Mapping and localization: state-of-the-art

Contents

Résumé en français

Le présent chapitre est consacré à l'étude de l'état de l'art des approches de localisation et de cartographie. Dans la première partie, nous étudierons les différentes représentations de cartes à savoir: les cartes métriques et topologiques. Ensuite, nous étudierons les différentes approches de localisation qui sont conçues suivant la carte utilisée. Etant donné que la littérature sur la localisation est extrêmement large, nous nous focalisons dans ce chapitre aux travaux utilisant les LiDARs et les caméras.

2.1 Overview of mapping solutions

2.1.1 A taxonomy of map representations

Environment mapping has been widely addressed by the robotics research community since the 1980s. The complexity of the mapping problem is the result of several important factors. At first, the studied environment and its size. For instance, indoor and outdoor environments should not be similarly considered in the mapping approach: indoor environments are more controllable, less spacious and more structured than outdoor environments. The size of the environment is also important considering the limited detection ranges of sensors. Indeed, short range sensors would not be appropriate for large environments and vice versa. The second and the most important factor is the accuracy of perception and localization. Localization accuracy is important for global map consistency and perception accuracy is important for local map consistency. Under the restrictive assumption that the robot poses are known, the problem is known as *mapping with known poses* and is less complex than the general case where the poses are unknown and have to be estimated, namely the *Simultaneous Localization And Mapping (SLAM)* problem. SLAM approaches will be discussed further in this chapter. Here we focus on the first problem in which we still need to deal with perceptual errors and ambiguities. Perceptual errors are mainly due to raw sensor data errors and the errors of the processing unit itself (modeling errors, discretization, sub-sampling ...). The perceptual ambiguity is the problem of finding associations between objects

that look alike. This may happen, for example, when the robot revisits the same place twice.

The classification of mapping approaches is not an easy task since it may differ depending on the point of view. Mainly two different representations have been proposed: the metric maps [Moravec and Elfes, 1985] and the topological maps [Engelson and McDermott, 1992]. Metric maps provide the geometric properties of the environment with usually a fine grained resolution and in which the positions of objects such as obstacles are stored in a common reference frame. In topological maps the environment is represented by a set of distinctive places that are connected to each other to indicate different ways to navigate inside the environment [Kuipers and Byun, 1991]. The most widely used approach is the metric maps (which is also the type of map used in our approach). We mainly introduce two different metric map types: grid-cell map representations and Feature maps.

A Grid-cell map representations

The first metric map representation is the *occupancy grid map*. Early work dates back to the 1980s by Elfes and Moravec [Moravec and Elfes, 1985]. In the most simple formulation, the environment is discretized into a set of two-dimensional regular cells for which two different labels indicate if the cell is occupied or not. Obviously, using only two labels for the occupancy grid is not realistic because it does not account for the different type of errors: sensor data errors and position errors. A more realistic representation is to assign a probability value to a cell being occupied by an obstacle as shown in Figure 2.1.

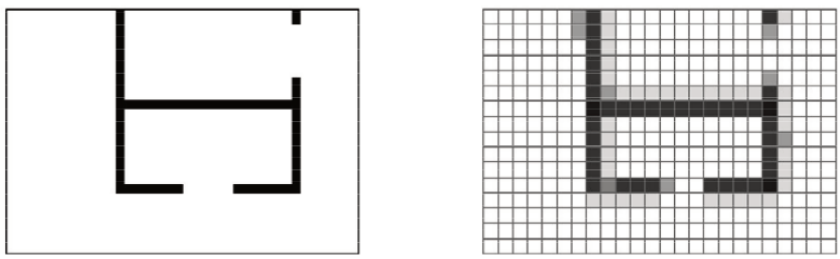
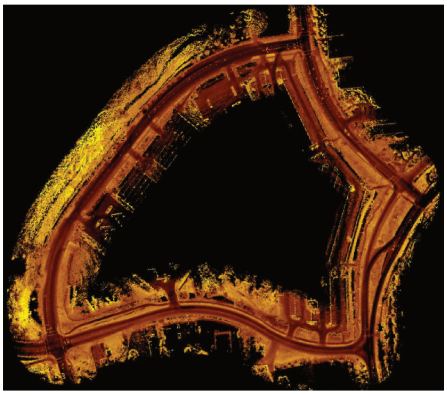
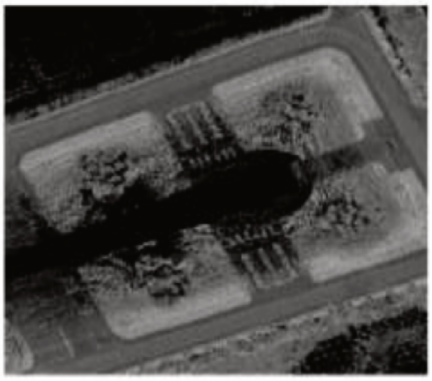


Figure 2.1: Illustration of an occupancy grid representation of a real environment

Other approaches adopted the grid-cell representation but chose to store different data types in each cell. For example, the proposed approach in [Levinson et al., 2008] uses a 3D laser scanner to build a reflectivity grid map (Figure 2.2a). Lidar reflectivity data provides information about the brightness (reflectance) of the objects in the scene.



(a) Reflectivity grid map [Levinson et al., 2008]



(b) Height grid map[Fu et al., 2016]

Figure 2.2: An example of a reflectivity & height grid maps

A laser scanner is also used in [Fu et al., 2016] to build height maps (Figure 2.2b) where each cell contains the mean height value of all the 3D points that fall within it.

The main advantage of these methods is that they can be directly constructed from raw sensor data without having to implement computationally expensive feature extraction algorithms. However, for large environments, grid-cell representations are memory-extensive. For example, mapping 20.000 miles of roads required a storage capacity of 200 GB in [Levinson et al., 2008].

B Feature-based maps

Feature-based or landmark-based maps store distinct features extracted from raw sensor data along with their positions. The approach is to detect distinguishable landmarks and to design descriptors to associate data between measurements and map attributes. In general, the nature of the extracted landmark highly depends on the used sensor and the information it provides. For example, visual features like points, lines, segments, corners have been investigated in [Lu and Song, 2015] and [Mur-Artal et al., 2015]. The feature may exhibit geometric characteristics such as points, lines, corners or intensity characteristics such as blobs. Visual feature association is often ensured by designing several feature descriptors such as SURF [Bay et al.,], SIFT [Low, 2004] and ORB [Rublee et al., 2011]. Laser scanners have also been investigated. A feature can vary from a single corner extracted from the point cloud [Borghi and Brugali, 1995] to a set of points scattered over object surfaces [Thrun, 2002b]. In contrast to grid-maps, feature-maps are light-weight, compact and are most often used in Kalman filter-based SLAM systems. The difference is, instead of using raw sensor data, a feature detection module should be designed.

2.1.2 Mapping for autonomous vehicles: current trends

Maps for autonomous driving contain both metric and topological representations. Metric layers are physical objects that constitute the structure of roads like lane markings, traffic signs, road sides, etc. Topological representation is usually represented by a road network layer which indicates how the different road segments are connected. This is sometimes known as *routing*. High Definition (HD) maps for autonomous driving are currently being developed by many commercial maps providers like TomTom [Tom, 2019], Here [Her, 2019] as well as free mapping projects like OpenStreeMap (OSM) [Haklay and Weber, 2008]. However, the main difficulty is the availability of a unified map representation format that takes into account the requirements of autonomous vehicles. On the contrary, several mapping approaches are tailored for particular applications. For example, the Route Network Definition File (RNDF) format was developed in the context of the 2007 DARPA Urban Challenge [Darpa, 2007]. In this format, each road is composed of a set of segments which comprise one or more lanes. A lane is a set of waypoints (points in the WGS84 (G1150) reference frame) along with a width parameter. In addition, other labels are attributed to some waypoints (entry and exit waypoints) to identify connections between segments. Similarly in [Bender et al., 2014], a map framework based on *lanelets* was developed for maneuver planning. A lanelet describes the actual lane with two bounds left and right that are modeled by polylines (Figure 2.3). This representation seems to be complete from the topological point of view but lacks details in the geometric representation of lanes as they are represented by a set of lines (polylines). Polylines may be considered as a weak primitive to represent high curvature roads such as roundabouts. To overcome this weakness, the enhanced map *Emap* was proposed in [Betaille and Toledo-Moreo, 2010]. Instead of using polyline representation, this work proposes to model the road by a series of straight lines, circles and clothoids. Clothoids fit the road shapes better and enable to decrease the amount of information (waypoints for instance) stored in the map.

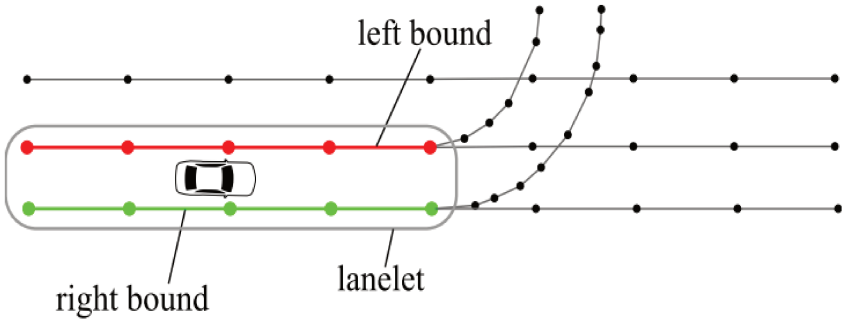


Figure 2.3: Lanelet map proposed in [Bender et al., 2014]

Although many publications cover maps for highly automated driving, there is still

a lot of work to be done towards formulating requirements for high definition maps and implementing a standard map format for autonomous vehicles. Though, recently, the problem has been tackled in the general robotic field by the *IEEE RAS Map Data Representation Working Group* [IEEE Robotics and Automation Society, 2015].

2.1.3 Concluding remarks

The literature overview of mapping approaches shows two different map representations: metric and topological. Metric representations model the physical world and the geometric properties of the environment. They have been essentially used in specific use cases such as localization. In topological or multi-layers topological maps, the world is modeled by accessible areas that are connected to each other via arcs. This representation is very useful for path planning and decision making tasks. Maps used for autonomous vehicles are a mixture of metric and topological representations. The represented geometric entities are in general: road segments, lanes, traffic signs, traffic lights, etc. The topological representation ensures the interconnections between road segments and the definition of the driving direction, etc. Although a lot of work has been done in map representation for autonomous driving, the development of standard requirements is still an important step to be done. In our approach, we adopted a metric map representation since we are aiming at developing a localization system. The used map mainly contains basic road features like road markings and traffic signs. In the following section, we give an overview of different localization techniques which are also as important as the mapping task.

2.2 Overview of localization techniques

Localization approaches addressed in the context of the autonomous vehicles are derived from the approaches that have been developed in the field of robotics. Robot localization is the task of estimating the robot position in a given reference frame. We distinguish between two different localization strategies: local and global. In local techniques, the initial robot position is assumed to be known and the current position is estimated from the previous position using a measure of its displacement using proprioceptive or exteroceptive sensors. This is also known as the *position tracking* problem. When the robot is required to build and update a map of an unknown environment while simultaneously localizing itself in it, the problem is called Simultaneous Localization And Mapping (SLAM). Global localization can localize a robot without prior knowledge of its initial position. This strategy is also known as the *kidnapped robot problem* or the *lost robot problem*. Position tracking, SLAM and global localization will be discussed in details in the following subsections.

2.2.1 Position tracking

Position tracking aims at estimating the robot displacement between two consecutive sensor readings. The displacement is computed by directly integrating proprioceptive sensor measurements like encoders, accelerometers and gyroscope. Thus, the displacement is deduced by solving the *dead reckoning navigation problem* [Park et al., 1998, Chung et al., 2001]. For exteroceptive sensors like cameras and laser scanners, the task is more challenging because the displacement cannot be inferred directly from raw sensor data, instead, some processing tools are required. For cameras, the task is usually known as *visual odometry* and for laser scanners, it is most often referred to as *point cloud registration*. Of course, other exteroceptive sensors could also be used such as radars [Schuster et al., 2016] and sonars [Ribas et al., 2008].

A Formulation of the problem

Let $S_{0:n} = \{S_0, S_1, .., S_n\}$ be a series of sensor readings at discrete time instants. For simplicity, the sensor local coordinate frame is assumed to be the same as the robot local coordinate frame. The goal is to calculate the transformation matrix $T_{k,k-1} \in \mathbb{R}^{4x4}$ which model the ego motion rotation $R_{k,k-1}$ and translation $t_{k,k-1}$ matrices between two sensor readings S_{k-1} and S_k:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

The pose of the robot P_k at time k is the concatenation of all subsequent motions $T_{k,k-1}$ and is given as follows:

$$P_k = T_{k,k-1} \times P_{k-1} \tag{2.2}$$

The position P_0 is assumed to be known. The full state position estimation is a 6 Degrees of Freedom (DoF) problem that consists of three translations and three rotation angles. For some mobile robot applications (including autonomous vehicles), the estimated position can be modeled by three parameters: a two-dimensional position and a heading angle. In the following paragraphs, different approaches proposed for camera-based and laser-based position tracking systems are described.

B Camera as the only exteroceptive sensor

The use of camera for ego motion estimation has been widely addressed in the literature and is commonly known as Visual Odometry (VO). Historically, this problem is known in the computer vision community as *structure from motion* (SFM) and dates back to the 1980s [Longuet, 1981], [Harris and Pike, 1987]. SFM focuses on the estimation of both the camera positions and the structure of the environment from the image set. VO is a particular case of SFM that focuses essentially on the estimation of camera

positions. There are two classes of VO algorithms: stereo VO and monocular VO. In stereo VO, the 3D positions of the features can be inferred directly from image data by means of triangulation. In contrast, in monocular VO, the motion of the camera can be recovered only up to a scale. However, the stereo VO can degenerate when the distance to the scene is larger than the baseline. In this case, the monocular camera is much more reliable. Whether it is a stereo or monocular VO, the general pipeline and the challenges remain nearly the same.

The general pipeline consists of four different steps (Figure 2.4). The first step is feature detection which consists in selecting some key points in the image data. Two different approaches are discussed in the literature: dense and sparse methods. In dense (or appearance) methods, the image is taken as a whole: all the pixels are features, whereas in sparse methods, only several distinguishable points are extracted from the image. The second step is feature matching or feature association where each feature detected in one image is matched with the corresponding feature detected in the other image. The third step is motion estimation where the transformation matrix T is computed from the associated feature points. The motion estimation step must perform outliers rejection in order to solve the motion system of equations based on inliers only. The last step is a local optimization which is known as the bundle adjustment problem [Triggs et al., 2000]. It consists in obtaining more accurate estimate of the local trajectory by means of iterative refinement over the last n frames.

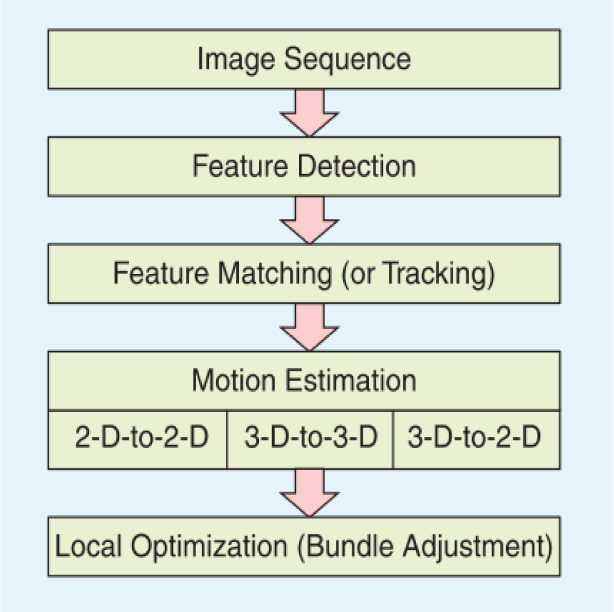


Figure 2.4: General pipeline of a VO system [Scaramuzza and Fraundorfer, 2011]

Early research on ego motion estimation using stereo cameras has been introduced by Moravec [Moravec, 1980]. The author proposed a single camera sliding on a rail that captured nine different images at equidistant intervals. Then, a corner detector

(*Moravec corner detector* [Moravec, 1977]) was developed to detect feature points in one picture and match them along the epipolar lines of the remaining eight pictures using normalized cross correlation. The system of equations of the ego motion estimation was solved using weighted least square minimization. Later, [Matthies and Shafer, 1987] proposed to use the *Moravec corner detector* for feature detection and improved the scalar representation of uncertainties in the triangulation of 3D points by setting, instead, a 3D normal distribution to represent the errors. As a result, the obtained trajectory is more accurate than the one proposed by Moravec [Moravec, 1980].

A dense stereo VO approach is proposed in [Lacroix et al., 1999] for planetary rovers. Feature points are selected by defining a similarity measure based on correlation scores of one pixel with each of its neighbors. In order to eliminate false matches, the authors implemented a tracking algorithm which checks if a corresponding feature remains in a small area around the initial feature. The size of the search area is defined according to the uncertainty of the estimated transformation from the robot internal (proprioceptive) sensors. Finally, motion is estimated by applying a constrained least square method to the associations. The displacement estimated by the algorithms was close up to 1% to the GPS positions.

A remarkable real-time approach was proposed by Nister [Nistér et al., 2004]. A Harris corner detector [Harris and Stephens, 1988] was used to detect features from the image. In contrast to previous work, the proposed feature matching does not include a tracking step over multiple frames, which limits the problem of feature drift over time. Motion estimation is solved using the 5-points algorithm. To get rid of outliers, the authors proposed to integrate the RANSAC outliers rejection algorithm into the motion estimation [Nist, 2003]. The best obtained accuracy is about 1.08% of the total trajectory.

Finally, in [Lefaudeux and Nashashibi, 2012], FAST corner detection and Lucas & Kanade tracker [Lucas and Kanade, 1981] are used to detect and track spatial features from stereo images. The ego-pose is estimated by standard photogrammetry techniques (namely SVD). The authors did not report quantitative accuracy results due to lack of ground truth data.

Monocular VO is particularly different from stereo because the relative motion and the 3D structure have to be computed from bearing-only data. The estimated trajectory is only valid up to a scale factor. Nister et al also tackled the case of monocular camera [Nistér et al., 2004]. In contrast to the stereo version where the triangulation is directly performed, the monocular version needs to track over multiple frames for 3D points triangulation. Tardif et al [Tardif et al., 2008] used an omnidirectional camera for visual odometry. Their key contribution is the decoupling of rotation and translation in the motion estimation step. The rotation and translation are respectively calculated using the vanishing points and a 3D landmark map that is built by triangulation of SIFT features over multiple keyframes. A RANSAC scheme is also implemented for outliers rejection. The VO system was run for 2.5 Km and the claimed accuracy was about 2.5%

of the total trajectory.

Combining visual and inertial measurements has been addressed in the literature to complement monocular cameras with metric scales (from IMU). This is known as visual-inertial odometry (VIO). Historically, there have been two different concepts for visual-inertial odometry: batch nonlinear optimization methods and recursive filtering methods. While batch optimization methods [Leutenegger et al., 2015, Qin et al., 2018] jointly minimize the errors originating from integrating the IMU and vision data, recursive algorithms [Chai et al., 2002, Roumeliotis et al., 2002] usually use the IMU for state propagation and updates from visual observations.

The above approaches tackled the case of unconstrained (full) motion estimation problem with 6 DoF. However, for autonomous vehicles it is common to use nonholonomic constraints to reduce the complexity of the motion estimation problem and, as a result, enhance the runtime performance. An example of a 2 DoF motion model is addressed by Scaramuzza et al [Scaramuzza et al., 2009]. It was assumed that the motion of a camera can be locally described with a circular motion and only 1-point algorithm was needed to recover the trajectory. The method was tested with different feature detectors: SIFT, Harris and KLT. Finally, two different approaches were proposed for outliers removal: *1-point RANSAC* and *histogram voting*.

All the approaches mentioned so far are subject to an unavoidable problem that is the drifting error over time. The drift is the natural consequence of the incremental path estimation process and must be kept as small as possible. To achieve this, several techniques like *the Bundle Adjustment (BA)* have been designed. BA is defined in [Triggs et al., 2000] as *the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates*. The main idea of BA is to optimize a cost function which is the re-projection errors of the 3D structure to obtain a very accurate model. This process involves a lot of matrix inversions yields a computational burden. A local version of BA, called *Local Bundle Adjustment (LBA)* or *windowed bundle adjustment*, was introduced in [Mouragnon et al., 2006] to side-step the computational load of the global method. Instead of optimizing over all the images, the author suggested to perform a local optimization only on a triplet of images. In addition to BA techniques, fusion with other sensors such as IMU [Konolige et al., 2007] and laser scanners [Zhang and Singh, 2014a] have also been proposed to cope with cumulative drift.

So far, the vast of majority of visual odometry state of the art approaches are developed under a standard pipeline that includes: feature extraction, feature matching, motion estimation and local optimization. Recent work focuses on developing an end-to-end visual odometry methods using deep learning techniques as in [Wang et al., 2017]. The aim is to avoid the standard pipeline and to train a deep learning model to optimally learn effective feature representation for the VO problem and to model sequential dynamics. As for other research fields, these methods have gained interest in the scientific community.

C Laser scanners as unique exteroceptive sensors

In this paragraph, we address the ego motion estimation by using laser scanner data (point clouds). In the literature, it is more commonly known as *point cloud registration* instead of *laser odometry*. Point cloud registration is very often solved by means of the well established *Iterative Closest Point (ICP) technique* [Besl and McKay, 1992]. ICP starts with two point clouds and an initial guess of their relative transformation, then it iteratively refines the transformation by minimizing a cost function (metric error). The general pipeline of an ICP algorithm is depicted in Figure 2.5.

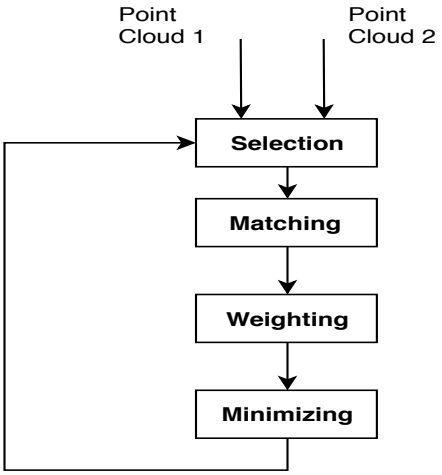


Figure 2.5: General pipeline of an ICP algorithm [Rusinkiewicz and Levoy, 2001]

The first step is to select some (or all) set of points in one or both point clouds. This is very similar to the feature detection step in visual odometry. The matching step is finding correspondences between the selected sets. The weighting step is assigning a relative weight to each of the associated points. Their relative weights should indicate how reliable the feature correspondences are. Finally, the minimization step is calculating the geometric transformation by minimizing a cost function that takes into account the set of feature correspondences and their relative weights. The typical problem faced in point cloud registration is the distortion caused by moving sensors.

In case of moving sensors, point cloud distortions are related to two main factors: the velocity of the agent carrying the laser scanner and the frame rate of the latter. The distortion is low for high frame rates, but can be significant for low frame rates. Several methods have been proposed for distortion correction, VICP uses a motion model to iteratively correct the measured points [Hong et al., 2010]. It is an ICP based on velocity estimation followed by distortion correction using the estimated velocity. Other methods use the reflectance images from the lidar and implement distortion correction by means of a visual odometry algorithm [Tong and Barfoot, 2013].

[Zhang and Singh, 2017] method, named LOAM (Lidar Odometry And Mapping), is considered as the state of the art solution for lidar odometry. The selected points for the ICP algorithm are points on sharp edges and planar surface patches. A KD-tree algorithm [De Berg et al., 2008] was used to find correspondences by searching for the nearest neighbor. The overall distances of feature correspondences are minimized using the Levenberg-Marquardt non-linear minimization method [Richard and Andrew, 2004]. In order to correct the distortion the authors modeled the robot motion with constant angular and linear velocities during the start and the end of the laser sweep. The proposed method have been evaluated using KITTI datasets [Geiger et al., 2013], the average position error is 0.88% of the traveled distance.

The same approach for distortion correction was adopted by Moosmaan and Stiller [Moosmann and Stiller, 2011a]. They constructed a 2D array of range measurements from which they extracted surfaces along with their normal vectors and a confidence value. Then, a de-skewing step consists in linearly transforming the extracted surfaces by using the estimated velocity from the previous iteration. Finally, an ICP algorithm was implemented to solve the motion estimation problem. This method has been tested for two different scenarios. In scenario 1, the position estimation error is 2.29 meters for a total length of 1.3 km, whereas in scenario 2, the position estimation error is 4.10 meters for a total length of 1.1 km.

In [Bosse and Zlot, 2009], a 2D laser scanner (SICK LMS291) is used to discretize the environment into a 3D grid of voxels. Planar and cylindrical surfaces were estimated for each voxel points, and the shape parameters were computed using the first- and second-order moments. Correspondences were found by searching for the nearest neighbor of each voxel in a 9D descriptor space (space of surface parameters). The reported accuracy of this method is less than 1% of distance traveled.

In [Biber and Strasser, 2003], another registration approach called the normal distribution transform (NDT) was proposed. The 2D plan was discretized into a set of cells (2D grid) for which a normal distribution was calculated from the corresponding points. The NDT represents the probability of measuring a sample for each position within the cell. The alignment of two different scans was ensured by calculating a score value of each point in the second scan by means of the normal distribution transform of the first scan. The authors did not report quantitative results on position errors, instead, they analyzed qualitatively the obtained map.

A generalization of the NDT-based scan registration method using 3D data was developed in [Magnusson et al., 2007]. The authors compared the performances of ICP-based registration and NDT-based registration. They concluded that 3D-NDT is more accurate than ICP. 3D-NDT is also faster because it avoids the data association step of the ICP. The proposed approach was tested for mobile robot applications, the authors compared the pose odometry results by varying the cell size. They concluded that a size of $2m$ is preferable and gives translation error less than $0.5m$ and a rotation error less than $0.01rad$.

D Combining cameras and laser scanners as exteroceptive sensors

Visual odometry and point cloud registration can be combined into one framework for robot motion estimation. The idea is to combine the advantages of both methods to improve the accuracy of the motion estimation problem. The proposed approaches either use point cloud registration as the main motion estimation pipeline with the aid of motion data from visual odometry, or use the visual odometry as the main motion estimation pipeline and complete with 3D data from laser scanners. Examples of the first category can be found in [Zhang and Singh, 2015] and [Pandey et al., 2011]. Zhang and Singh [Zhang and Singh, 2015] implemented an ICP algorithm by selecting points on sharp edges and planar surfaces as geometric features from a 2-axis laser scanner. The initial guess for the ICP was provided by a visual odometry system [Zhang and Singh, 2014b]. Also, in [Pandey et al., 2011], the initial guess for the ICP algorithm was provided by a vision system. The authors proposed to complete the lidar 3D points with high dimensional features (SIFT descriptors) from camera images and used these points in a RANSAC framework to obtain an initial guess of the alignment transformation. The authors claim that using visual descriptors is more efficient than using euclidean distances for the data association problem.

Most of the proposed approaches that consider visual odometry as the main pipeline for motion estimation use monocular cameras and complete visual data with 3D laser scanner data [Huang and Stachniss, 2018, Huang et al., 2019]. A 1-DoF ICP variant was proposed in [Huang and Stachniss, 2018] where a visual odometry system is implemented to compute the rotation and translation of the transformation between two frames of a monocular camera. Given that the translation is only valid up to a scale factor, 3D points from laser scanners are fed into an ICP algorithm which estimates the true scale factor then refines the visual odometry estimated translation in a final ICP step. The authors also proposed a constrained data association strategy for outlier rejection. Indeed, the estimated rotation from visual odometry is used to guide data association in all ICP iterations. This approach outperforms visual odometry from the monocular camera, since it provides accurate scale estimate, as well as laser-based ICP (more accurate orientation and translation).

A better accuracy was obtained with the direct approach proposed in [Huang et al., 2019]. This method used two registration stages: the first stage tries to find a proper initial pose estimate by jointly performing a coarse photometric pixel-alignments along with a geometric point cloud registration. The second stage is performed by aligning only pixel intensities at the finest image level. Good performance is obtained by an occlusion detection of sparse point cloud, which reduces the impact of outliers in the photometric alignment problem, that is which is based on the *constant image brightness* assumption. The authors tested their approach on KITTI datasets and reported an error of 0.6% of traveled distance on sequence 04. They also compared their approach to results of LOAM [Zhang and Singh, 2017]. Combining laser and camera outperforms lasers alone; e.g LOAM 1.4% .Vs. 1.0% on sequence 01 (which is a highway

environment).

2.2.2 Global localization

Contrary to local localization that is considered as a continuous position estimation problem, global localization is rather a discrete and qualitative problem that does not require any prior knowledge of the initial position of the robot. Global localization techniques are divided into two different categories. The first category is *GNSS-based* which uses the Global Navigational Satellite System as the main source of information. Most of the proposed solutions in the literature combine GNSS data with other (most often proprioceptive) sensors and a prior map. The second category is *Map-based* which uses exteroceptive sensors and a prior map as the main sources of information to infer the robot position. In the next two paragraphs, we respectively address GNSS-based and Map-based localization.

A GNSS-based solutions

GNSS has been the standard global localization technique for many years. Satellite-based positioning methods rely on the trilateration of different satellite signals in order to derive the receiver's position. Each satellite broadcasts a coded signal that the GNSS chipset receives and interprets. Using the time of arrival (TOA) and the speed of each signal (the speed of light), the GNSS sensor calculates its distance to the satellite, called *pseudo range*. The position of the receiver is deduced by trilateration from n different satellites. The minimum required number is four: three satellites to determine the longitude, latitude and altitude values and the forth is used to synchronize the time. There are several sources of errors, the first source of error is the difference between the pseudo range and the true range (Figure 2.6). This error is illustrated in the intersection area (uncertainty region) between the different pseudo range circles in (Figure 2.6). Another important problem is the *multipath error*. This error occurs when the receiver reads the same signal from different paths rather than from a direct line of sight (LOS). Mutliple paths are caused by reflections, for example, from high buildings in urban environments. Even in open sky environments such as in highways, the accuracy obtained by automotive GNSS receivers is not sufficient and is not better than 2-3 meters [Tgri,].

To improve the accuracy and to cope with GNSS shortcomings, the use of complementary navigational methods have been proposed. In [Obradovic et al., 2007], a Kalman filter fuses odometer and gyroscope sensors into a dead-reckoning process (DR). The estimated DR trajectory is corrected with high quality GNSS positions. A digital map is used to project the obtained trajectory by adopting a two-pattern feature vectors: straight-line and curve patterns. Similarly, the system proposed in [Fouque et al., 2008] uses a digital map, a dead reckoning process and GNSS measurements for a global localization of vehicles in a Extended Kalman Filter (EKF) implementation.

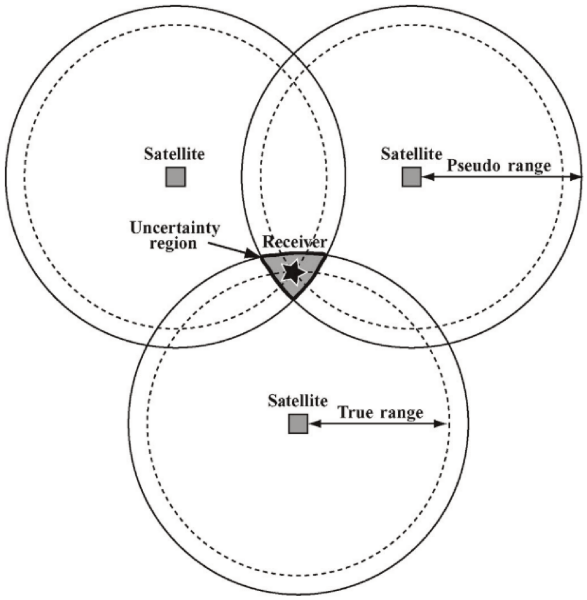


Figure 2.6: Trilateration from three different satellite systems [Skog and Handel, 2009]

DR is used for the prediction phase of the EKF and GNSS and map measurements are included in the correction phase of the EKF. The map is solely used to determine the heading of the vehicle. According to the paper, a DR drift is very dependent on the quality of the heading estimate, therefore, using the map as a source for heading information can be beneficial. An enhanced map *Emap* is used in [Toledo-Moreo et al., 2009] that contains the geometrical description of the center of each lane by series of clothoids. A particle filter is implemented to fuse the GNSS, dead-reckoning and the Emap for lane-level localization. The prediction step uses DR for particles motion and GNSS data and the Emap are integrated in the update phase of the filter.

B Map-based localization

In map-based localization, the idea is to use sensor data with a-priori built map in order to infer the position of the robot. The main advantages of using prior maps are the improved accuracy of the localization and the complementary information that can be provided to perception systems. Map-based localization are already categorized in section 2.1.2. In this paragraph, we focus on *metric map-based localization* as it represents the vast majority of implementations in the robotics field (and is also adopted in our study). We distinguish two different classes: Feature (or landmark) based localization which relies on Feature-based maps and Grid-based localization which relies on grid-cell map representation.

Feature-based localization relies on distinctive features (or landmarks) to infer the robot position. The map is represented by a set of 2D or 3D features and the localization is performed by matching extracted features from sensor data to the features

from the map. The choice of feature type depends on the studied environment and the used sensor. Various types of features have been proposed in the literature: either specific to the structure of the road such as lane markings [Suganuma and Uozumi, 2011a], corners of road marks such as arrows, speed limits [Ranganathan et al., 2013] and traffic signs [Li et al., 2010], or basic geometric features such as points [Wijk and Christensen, 2000], segments [Gomes-Mota and Ribeiro, 2000] and planar surfaces [Javanmardi and Javanmardi, 2017] or salient visual features of the environment like in [Caselitz et al., 2016, Se et al., 2005]. In any case, the appealing properties that a good feature should have are the availability (i.e can be frequently found in the environment) and the distinctiveness in order to minimize false matches. Given a feature-based map, the implementation of the localization process consists of three different steps:

1. *Feature extraction from sensor data.*
2. *Matching with map features.*
3. *Position error minimization.*

A vision-based system was proposed in [Ranganathan et al., 2013] to detect corners of specific road marks such as arrow and speed limits. The proposed system implements a visual odometry using calibrated stereo camera. At first, Harris corners are detected and tracked using a KLT tracker. Then, relative motion is computed using the 3-point algorithm [Richard and Andrew, 2004] with RANSAC estimator for robustness. A windowed bundle adjustment is used to adjust and smooth the noise due to visual odometry. A light-weight map that consists in road marks (corners and labels) augmented with GPS coordinates is used for online localization. The process incorporates the absolute coordinates of map attributes in order to reduce the drift in visual odometry. This method was compared to a Visual SLAM approach and proved to be less sensitive to significant lighting change. However, its major drawback is that it is only applicable to localization on roads with clearly painted visible markings. Also, it is difficult to generalize this method to different roads like highways because it relies on specific road mark types (stop, bike, turn arrow). Moreover, as the road marks are detected in an inverse perspective map (bird eye view) of the image, the pitch and roll variations may induce image distortions and detection errors.

In [Pink, 2008], a stereo camera and georeferenced aerial images are used for global localization. The localization process consists in matching the stereo images to aerial images using a feature-based technique. The used features are the centroids of lane markings computed for both type of images. Aerial images provide high pixel resolution (10x10cm per pixel) and are used to build a lane markings map (Figure 2.7). A Canny edge detector [Canny, 1986] is applied to stereo images to detect edge points that are clustered according to their proximity in pixel coordinates. The alignment between detected features and the corresponding feature map is achieved using ICP algorithm. Despite promising localization accuracies, using ICP may stick nearby local minima.

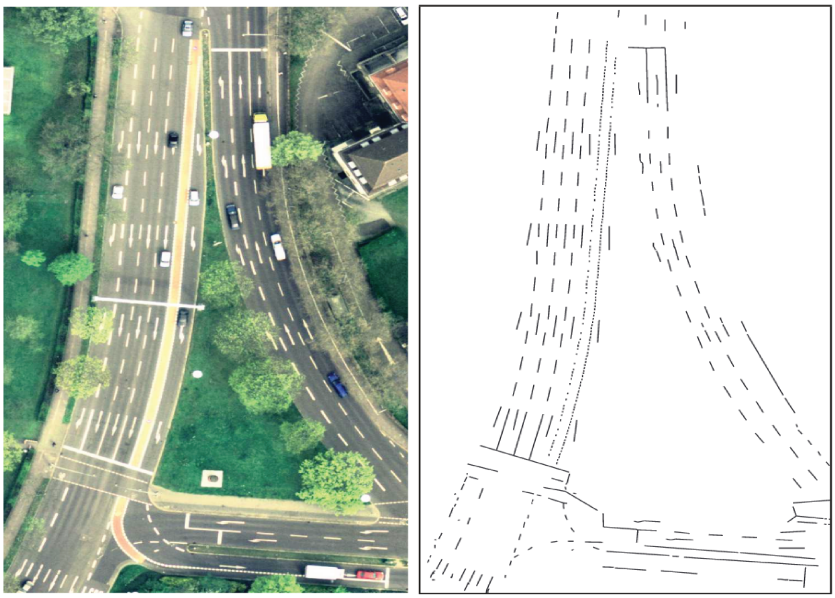


Figure 2.7: (Left) Part of the aerial image and (Right) corresponding feature map [Pink, 2008]

In [Javanmardi and Javanmardi, 2017], planar surfaces map were generated from a 3D point cloud. The idea is to exploit planar surfaces which are mostly available in urban areas, mainly ground and walls. The plane extraction is done by means of a RANSAC algorithm. Each plane is modeled by its center point, width, length, normal vector and a normal distribution that represents the uncertainty of the plane estimation. Those parameters are used for map matching in the localization process using a point-to-distribution variant of the normal distribution transform (P2D-NDT) [Magnusson et al., 2007] in order to match a 3D Lidar scan to planar surface map. It is also possible to use high level objects on the road such as lane markings and traffic signs. However, this method is mainly designed for urban environments where planar surfaces are very often

A lidar sensor was used in [Suganuma and Uozumi, 2011b] in order to estimate the lateral distance to lane markings based on reflectivity data. The calculated lateral distances and a digital lane marker map are input to a Kalman filter to correct the lateral drift error. The system was proved to be accurate in the lateral direction but still suffered from drift errors in the longitudinal direction.

Similarly, lane markings and traffic signs were used in [Li et al., 2010] where a particle filter algorithm was implemented for the localization system. In the correction step three different measurements were integrated: a GPS position, lane markings and traffic signs extracted from a vision-based system.

The grid-based localization problem can be formulated as finding a position that maximizes the correspondences between a local occupancy grid constructed from current and past sensor data, and a global map. Evidence grids were used in [Schultz and

Adams, 1998] for continuous localization (CL). The proposed method constructs a *mature* evidence grid by cumulating consecutive grids using odometry data. Then, the mature grid is aligned to the global grid using two different approaches: the first one is the iterative hill-climbing which looks iteratively in the vicinity of the current position for positions that maximize a matching score function. The second approach entails calculating the matching scores at different positions close to the estimated one and calculates a new position as the center of mass of the previous calculated ones weighted by their matching score values.

In [Thrun, 1998], a grid-grid matching using a differentiable correspondence function that measures the similarity between grids was adopted. A different grid was adopted in [Levinson and Thrun, 2010] where the reflectivity is the main data attributed to the grid cell. A likelihood function that measures the similarity between the local cumulated reflectance map and the global map is implemented within a histogram filter for online localization.

Likewise, the suggested approach in [Wolcott and Eustice, 2016] localizes a single monocular camera in a 3D prior reflectance ground map. Multiple synthetic views from different vehicle positions are extracted from the map. Then, a normalized mutual information image registration is implemented in order to find the best alignment between the current image and the synthetic views. To overcome the computational burden of the method, the authors used a GPU-based implementation to accelerate the process.

2.2.3 Simultaneous Localization And Mapping (SLAM)

So far, we have discussed the fundamentals of mapping and localization problems. On the one hand, environment mapping assumes that the robot positions are known, while on the other hand, some localization approaches need a prior map in order to estimate the position. In this subsection, a general overview of the Simultaneous Localization And Mapping (SLAM) is provided. As the name suggests, the aim of SLAM is to build a consistent map while simultaneously determining the robot position in it. A seminal work in SLAM is the research of Smith and Cheeseman in 1986 [Smith and Cheeseman, 1986]. Since then, intensive research has been conducted to solve and improve the SLAM problem. The difficulty of SLAM lies in its intersection with a variety of other fields such as: computer vision, geometry, dynamics, optimization and probabilistic estimation. The SLAM problem has been tackled in two different ways. The first is called EKF-based SLAM where an Extended Kalman Filter (EKF) is implemented to recursively estimate a Gaussian density over the current robot and landmark positions. Related methods are based in general on linearizing nonlinear processes (motion model and correction model), and quickly become computationally intractable when the number of landmarks grows significantly (e.g. in outdoor environments). In addition, the linearization has a direct impact on the consistency of the map as well as the estimated position. A thorough review of EKF- based SLAM tech-

niques can be found in the two-part tutorial in [Durrant-Whyte and Bailey, 2006] and [Bailey and Durrant-Whyte, 2006] that covers the *classical age* of SLAM from 1986 to 2004. The second way is known as smoothing and mapping (SAM). It is a variation of SLAM where the problem is formulated as a graphical model and is solved in a least squares framework. Recent reviews of modern SLAM systems have been proposed in [Bresson et al., 2017, Cadena et al., 2016]. Finally, a comparison between filtering and smoothing based SLAM is proposed in [Strasdat et al., 2010]. For convenience, we consider the smoothing and mapping SLAM variation and we simply refer to it as SLAM.

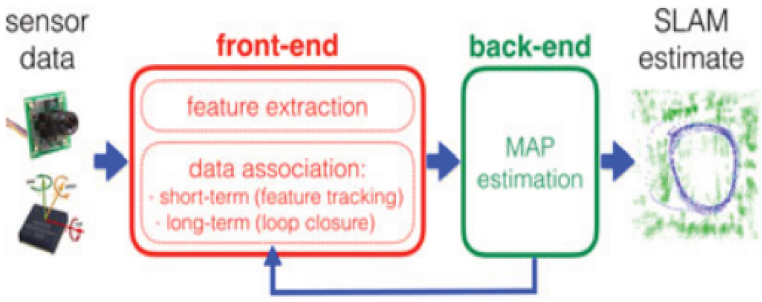


Figure 2.8: Front end and Back end SLAM modules [Cadena et al., 2016]

The general architecture of a SLAM problem is composed of two different modules: the *front end* and the *back end*, as depicted in Figure 2.8. The *front end* module involves inference of sensor data in order to extract relevant features or landmarks. It also involves the well-known data association problem which consists of two different tasks: feature tracking and loop closure. Features are tracked over time by searching the correspondences the current frame and previous frames. The extracted features mainly depend on the used sensor and the studied environment. For example, SLAM variants based on vision systems (Visual SLAM or VSLAM) have been widely investigated. An in-depth review of VSLAM approaches is presented in different survey papers such as [Fuentes-Pacheco et al., , Taketomi et al., 2017]. Other approaches use laser scanners (Laser SLAM) have also been proposed [Moosmann and Stiller, 2011b, Deschaud, 2018]. Feature extraction and tracking (also named *short term data association*) represent the odometry (localization) module of SLAM. Loop closure is what makes SLAM different from simple odometry. It ensures the global consistency of the map by reducing the localization error when revisiting known areas.

The *back end* module uses the extracted features along with the robot poses and formulates a maximum a posteriori (MAP) estimation problem. It takes as inputs a vector $\mathcal{X}$ of both the trajectory poses and the landmark positions and a vector of measurements $Z := \{z_k, k = 1..m\}$. A measurement z_k can be expressed as a function of a subset X_k of $\mathcal{X} := z_k = h_k(X_k) + \epsilon_k$ where ϵ_k is a random measurement noise. In

MAP estimation, $\bar{\mathcal{X}}$ is estimated by maximizing the posterior $p(\mathcal{X} \mid Z)$ according to the following equation:

$$\bar{\mathcal{X}} = \arg\max_{\mathcal{X}} p(\mathcal{X} \mid Z) = \arg\max_{\mathcal{X}} p(Z \mid \mathcal{X}) p(\mathcal{X}) \tag{2.3}$$

Assuming that the measurements are independent given the variable $\mathcal{X}$, the above equation can be written as:

$$\bar{\mathcal{X}} = \arg\max_{\mathcal{X}} p(\mathcal{X}) \prod_{k=1}^{m} p(z_k \mid \mathcal{X}) \tag{2.4}$$

$$\bar{\mathcal{X}} = \arg\max_{\mathcal{X}} p(\mathcal{X}) \prod_{k=1}^{m} p(z_k \mid X_k) \tag{2.5}$$

A factor graph is often used to solve the MAP estimation problem. A factor graph is a graphical model that encodes the dependence between the kth factors and the corresponding variable X_k. The nodes of the factor graph are the trajectory and landmark positions and the factors are the probabilistic constraints $p(z_k \mid X_k)$ and $p(\mathcal{X})$ depicted in equation 2.5. For simplicity, Gaussian noises ϵ_k with information matrices I_k are considered:

$$p(z_k \mid X_k) \propto \exp(-\frac{1}{2} \| h_k(X_k) - z_k \|_{I_k}) \tag{2.6}$$

Substituting 2.6 into 2.5, the maximization problem can be considered as a minimization of the negative log-posterior. Thus, the MAP estimate becomes:

$$\bar{\mathcal{X}} = \arg\min_{\mathcal{X}} \sum_{i=1}^{m} \| h_k(X_k) - z_k \|_{I_k} \tag{2.7}$$

This minimization problem is commonly solved by successive linearizations such as Gauss-Newton or the Levenberg–Marquardt methods. Successive linearizations transform non-linear problem into a set of linear equations called normal equations. Different libraries have been proposed to solve the linear equations of the MAP estimation: e.g GTSAM [Dellaert, 2012], g2o [Kummerle et al., 2011], iSAM [Kaess et al., 2008] and SLAM++ [Polok et al., 2013]. Most of them take advantage of the normal equation matrix to implement fast linear solvers.

2.3 Discussion

In this chapter, we presented two different categories of localization techniques: local localization (or position tracking) and global localization. In general, local localization is a sub-module of global localization techniques as shown in Figure 2.9. First,

input sensors can be either exteroceptive or proprioceptive. Second, the position tracking module ensures a continuous position estimation at a frame rate T_1 from sensor 1. In the literature, the main proprioceptive sensors for local localization are: wheel encoders, gyroscopes, accelerometers. They can be directly integrated to estimate the displacement of the vehicle (dead reckoning (DR)). For exteroceptive sensors, two different sensor technologies have been addressed: vision and lidar. Visual odometry (VO) is presented with two different variants: stereo VO and monocular VO. Stereo VO uses stereo vision and estimates motion using triangulated 3D points. Monocular VO only uses one camera hence suffers from an unknown absolute scale. Although the latter solved by stereo VO, it degenerates to the monocular case for distant features. Local localization is accurate for a short period of time. However it tends to drift for a long time period. When choosing a local localization technique, it is very important to take into account the studied environment and the use-cases. For outdoor environments (e.g. highway roads) visual odometry techniques have proven to be more accurate than laser odometry and IMU-based solutions. In addition, we claim that using visual odometry is more appropriate than using Visual SLAM as it is very unlikely to re-visit the same region twice (loop closure). The combination of visual-laser odometry outperforms visual or laser odometry alone since it combines the advantages of both methods. However, this assumes that both sensors are extrinsically calibrated (sensor-to-sensor and sensor-to-vehicle). In addition, due to the limitation of modern camera field of view (FoV), many 3D LIDAR will not be fully exploited as the overlapping of the 3D points cloud with the camera image will be restrained with its FOV. The second module is a map-matching algorithm that runs at a slower rate $T_2 > T_1$ and where sensor 2 is used to extract features or to build grids depending on the map representation (feature map or grid map). The goal is to reduce the cumulated drift from the position tracking module and to enhance the estimation of the system by matching map attributes to features/grids from input sensor.

2.3.1 Our proposed architecture

In the proposed approach, we used an inertial measurement unit for the position tracking module and a LiDAR sensor for environment perception (feature detection module). We adopted an IMU-based position tracking for two main reasons: IMUs do not depend on lighting conditions whereas vision sensors, for example, may encounter major difficulties in the absence of light. In addition, IMUs are already available in today's cars and are easy to integrate. A map-matching algorithm is implemented using a particle filter algorithm and a highly accurate light-weight map where primitive road features are stored: lane markings, traffic signs, guardrails, etc. Thus, we developed a road perception algorithm that uses LiDAR data to detect the same road primitives stored in the map. To validate our architecture, we compared the localization outputs to GNSS/RTK positions considered as ground truth. Different metrics are used for the evaluation: the euclidean metric to compute absolute errors and the curvilinear coor-

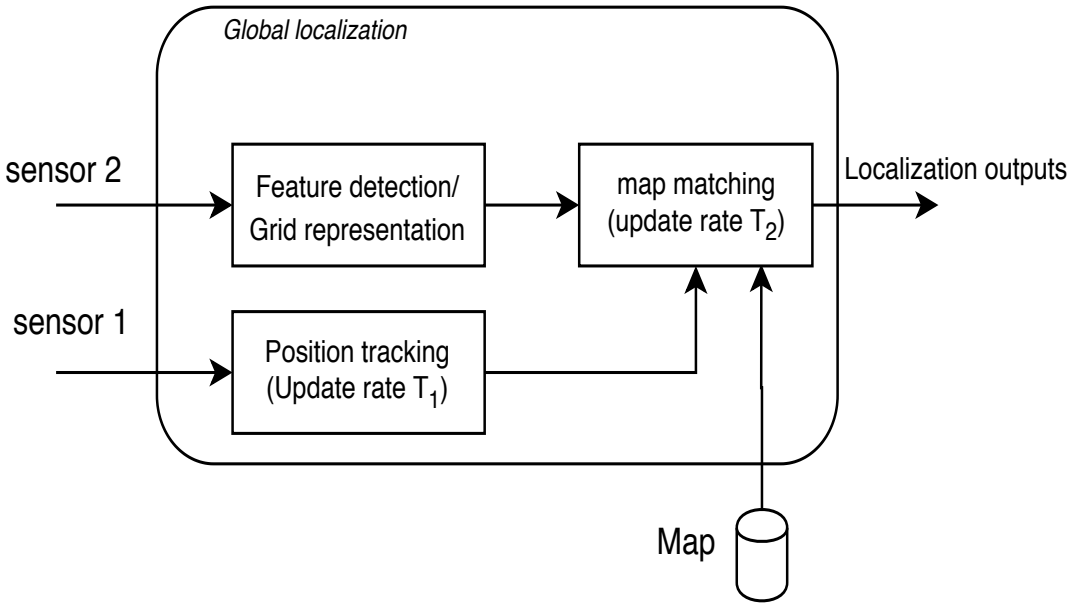


Figure 2.9: General architecture for global localization

dinates with respect to the map as described in [Héry et al., 2018]. The next chapter is dedicated to detail the proposed architecture.

Chapter 3

Design and evaluation of the proposed localization system

Contents

Résumé en français

Dans ce chapitre, nous détaillons notre architecture de localisation précise basée sur les capteurs LiDARs. Cette architecture est basée sur différents modules: le premier module est un module de perception qui permet d'interpréter les données du capteur afin d'extraire des amers liés à l'infrastructure routière tels que les lignes de marquages, les panneaux de signalisation et les barrières sur autoroute. Le deuxième module est un module de localisation ou "map matching" qui prend comme entrées les données du GNSS, les attributs de la carte et les données de perception et implémente un algorithme de filtrage particulaire pour estimer la position du véhicule. Enfin, on a un module d'évaluation et de validation qui évalue la précision de l'estimation de la trajectoire du véhicule en la comparant par rapport à une vérité terrain obtenue grâce à une GNSS/RTK.

3.1 Introduction

In this chapter, the proposed solution for perception and localization using LiDAR sensors is presented. The general architecture of the proposed solution is given in Figure 3.1. For online localization, we have three main modules: road perception that uses LiDAR sensors in order to extract road primitive features. In our case, we extracted lane markings, barriers, traffic signs and guardrail reflectors. The second module is the Map Management System that takes as inputs a rough vehicle position estimation (e.g GNSS data) and a highly accurate third party map in order to extract map features in the vicinity of the vehicle. The third module is a map-matching algorithm based on an improved version of particle filter. The evaluation of the localization system is performed with a reference to a highly accurate GNSS with RTK corrections. Different evaluation metrics will be discussed in the next chapter.

Before entering into the details of each module, we present, in the next paragraph, the setup of our prototype vehicle.

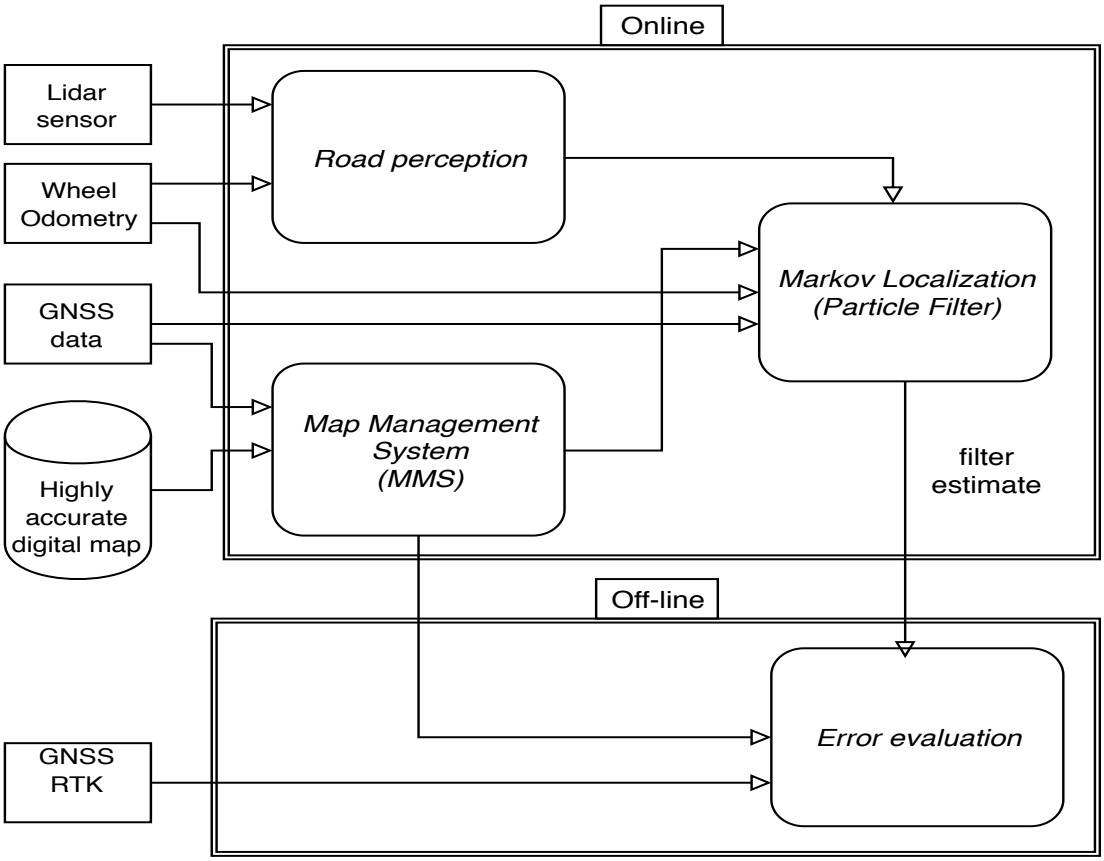


Figure 3.1: General architecture of our solution

3.1.1 Prototype vehicle: MELO

Our prototype vehicle "MELO" is a Renault Espace car that is equipped with 5 LiDARs: four roof-mounted LiDARs and one bumper-mounted LiDAR. The front LiDARs are Velodyne VLP32-C, and the rear LiDARs are Velodyne VLP-16. The vehicle is also equipped with GNSS receiver with automotive dead-reckoning from ublox (ADR78) and a GNSS/RTK for ground truth reference (ATLANS-C from iXblue). Finally, the central processing unit is an embedded PC that runs ROS (Robot Operating System) with the required modules to interface the sensors and insure time synchronization for data collection and replay. The prototype vehicle is shown in Figure 3.2.

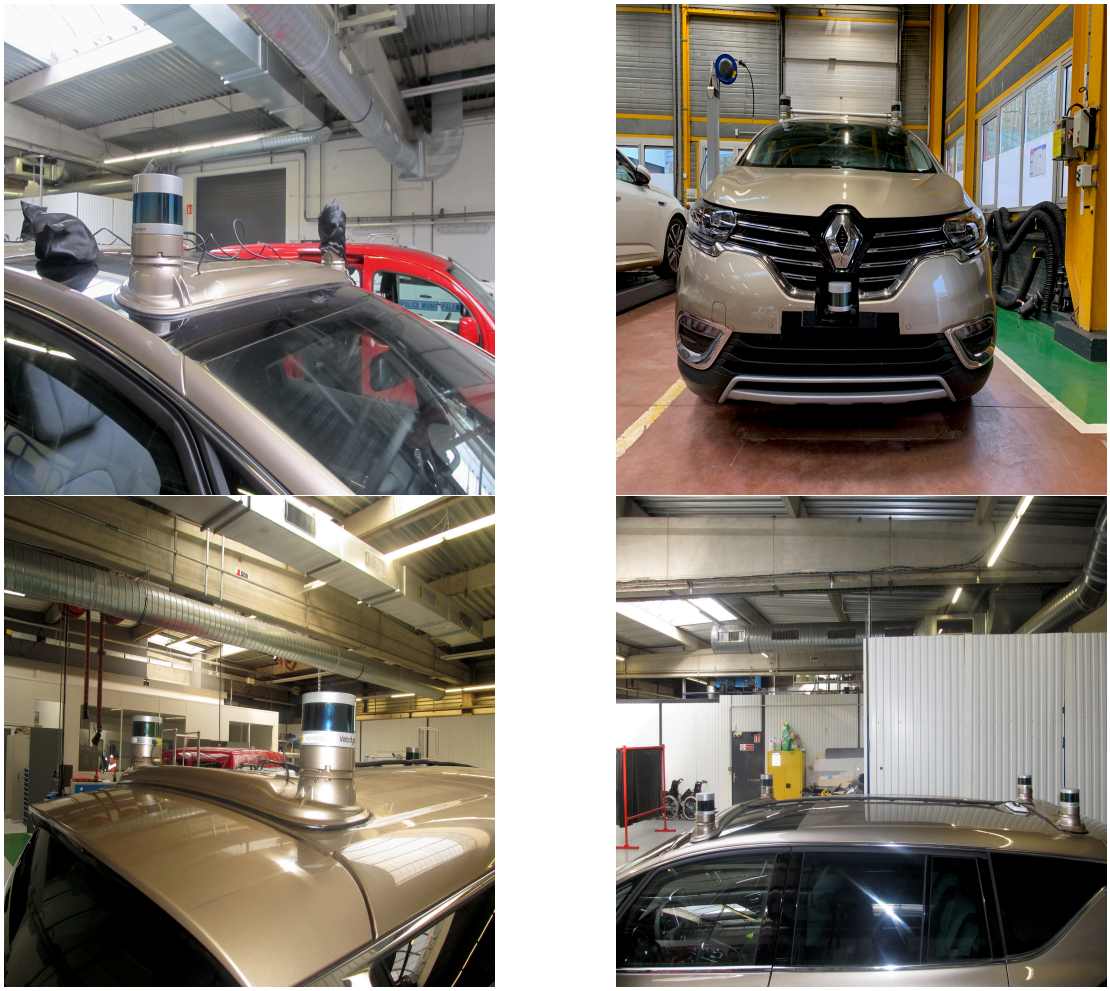


Figure 3.2: (Top left) Roof-mounted LiDAR. (Top right) bumper-mounted LiDARs. (Bottom right) rear LiDARs. (Bottom left) All roof-mounted LiDARs

3.2 Road Perception for Localization

Vehicle perception is a crucial task for self-driving cars to operate in real-world environments. For human drivers, road texture and color, road boundaries, traffic signs, lane markings and traffic participants (car, truck, motorcycles, cyclists, pedestrian) are the main perceptual cues for a safe driving experience. For autonomous vehicles, perception is very complicated due to many factors that are related to the environment and to the vehicle itself. Given the imperfect sensors, the unpredictable environment and the data processing time, there has not been a single perception system that can operate under all conditions and in all driving scenarios. Perception is by definition *a set of processing units that take as input sensor data and model, interpret and understand the surrounding environment*. According to the desired application, one may be interested in targeting some objects rather than others. For example, road perception focuses on the extraction of static road features such as road markings, the road surface, road boundaries and so on. Object detection and classification focuses on traffic participants such as vehicles (car, truck, motorcycles) and vulnerable road users such as cyclists and pedestrians. Sometimes, the detection of the absence of obstacle is more relevant than the obstacle itself (e.g. freespace detection or drivable area detection).

In this work, we focus on the detection of static road features for localization. Road perception modules are illustrated in Figure 3.3. The first module is the road detection and segmentation module where the road boundaries are detected. Detecting the road can be of several usages such as road mapping, detection of free (navigable) space, detection of lane markings, etc. The second module is the extraction of lane markings. Lane markings have been essentially utilized for assistant functions such as Lane departure Warning (LDW), Lane Keeping Assist (LKA) and also for more high level automated driving features such as localization and mapping. The third module is the detection and classification of traffic signs that aims at determining their 3D (or 2D) positions, shape and the sign information within.

3.2.1 Road perception: State-of-the-art

A Road detection & segmentation

Many cues have been reported in the literature to define road boundaries and to detect road surface. These cues can vary according to the used sensor. For lidars and stereo-vision, the use of geometric cues such as the planes, elevation, curbs, slopes, etc, often provide good accuracy in many scenarios: urban, semi-urban and highways. For example, in [Lombardi et al., 2005], the disparity map of a stereo-vision system is used for road detection. The road is detected as a 3D plane verifying some constraints in the disparity map. In [Pradeep et al., 2008], surface normals are used to generate piecewise planar models of the scene. Surfaces with similar normal directions are clustered and curbs are detected. The use of supervised machine learning techniques have also been investigated for road detection. In [Vitor et al., 2014], a multilayer perceptron (MLP)

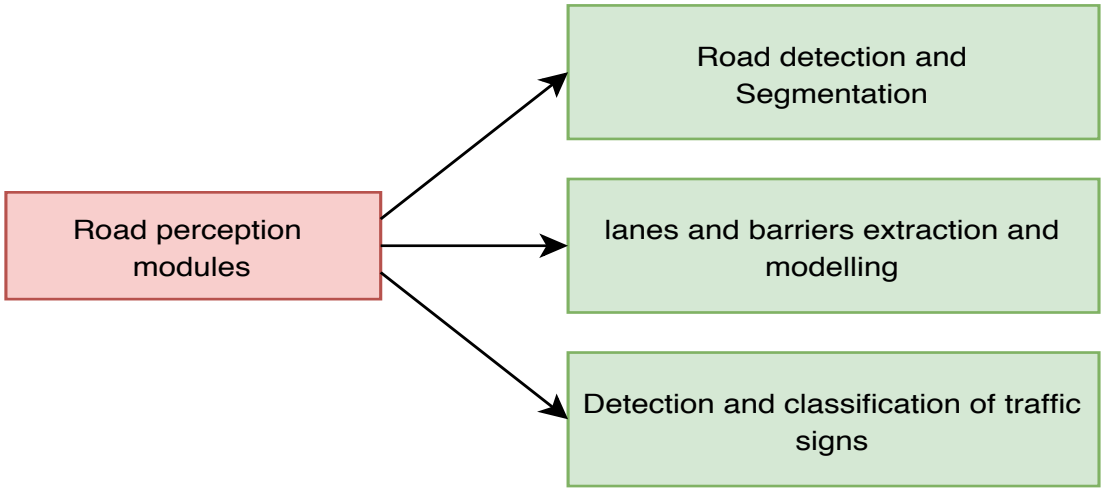


Figure 3.3: Road perception modules

is applied to learn road patterns resulting in the classification of the road recognition. LiDAR-based approaches use in general curbs, elevations and slopes. Very often, a grid representation is utilized. In [Hata and Wolf, 2014], the compression levels of consecutive rings are analyzed. Basically, the compression value is proportional to the obstacle slope. In [Kammel and Pitzer, 2008], an occupancy grid is used to classify curbs using the elevation difference. Indeed, a grid cell whose elevation difference exceeds a fixed threshold is labeled as a non ground cell. Similarly in [Chen et al., 2009], the elevation of the ground is estimated by averaging the elevations from the lasers directly striking the ground. By comparing point elevations with respect to ground elevation, an elevation difference value that is greater than $0.25m$ is considered to be a curb point. The same approach for detecting the ground is presented in [He et al., 2016].

In the absence of 3D data, many approaches for detecting the road surface are based on the appearance rather than the geometric elements. In this case, the road is assumed to have uniform appearance (texture, color) which is different from its surrounding. For example, in [Álvarez et al., 2007], a region growing algorithm is used to detect the road by converting the colored image into an illumination invariant image. In [McCall and Trivedi, 2006], a template matching is used to detect the road. The template is obtaining by applying an inverse-perspective warping to the image and cropping a local window that is assumed to belong to the road. In [Rasmussen and Korah, 2006], Gabor filters are used to detect dominant texture orientations, then, a vanishing point is detected through a voting scheme. Finally, the road is delimited by the two most extrinsic rays that pass through the vanishing point.

B Lane extraction and modeling

Detection of lanes is usually performed by seeking different types of lane features. This comprises two different stages: extraction of marking features or points and lane modeling. Most of state of the art techniques use vision-based systems for lane detection due to the variety of information that can be inferred from image data. Very often, lane markings are detected based on their shape and color. Their appearance is very different from the asphalt, therefore gradient-based approaches are applicable to extract a wide variety of lane markings. In [McCall and Trivedi, 2006], steerable filters are applied by computing three separable convolutional kernels in different directions of the image. This allows to measure the values and angles of the minimum and maximum responses. To detect lanes, the response in the direction of the lane should be near the maximum, and the minimum response should be low. Then, parabolic model is used in order to estimate the parameters of lane.

In [Aly, 2008], a real time detection in urban streets is proposed. At first, top view of the road is constructed by applying inverse perspective mapping technique. Then, a two dimensional Gaussian kernel is applied to filter the transformed image. Finally, a Hough transform is used to count the number of lanes and to indicate lane marking positions. As for lane modeling, RANSAC spline fitting is implemented. Thus, four control points are required to achieve the fitting process.

Road markings can also be detected using Lidar sensors. LiDAR reflectance measurements are very often used to achieve this goal. In [Hata and Wolf, 2014], road marking points are selected by analyzing the intensity histogram of road surface. It is shown that the histogram has bimodal shape; one mode for the asphalt and another for marking returns. Thus, separating asphalt from road markings consists in finding an appropriate threshold to separate the two modes of the histogram. The optimized threshold is calculated using Otsu thresholding technique [Otsu, 1979]. The authors in [Kammel and Pitzer, 2008] establish a lane marker intensity map to which the Radon transform [Beylkin, 1987] is applied to detect lane markers in the Radon plane. In [He et al., 2016], a convolutional neural network is applied to an intensity image obtained by projection of the intensities of road surface points.

C Detection and recognition of traffic signs

Extensive research has been conducted on the detection and recognition of traffic signs. ADAS systems based on this function are already in production and are known as TSR (Traffic Sign Recognition) systems. In the last decades, the use of vision systems for TSR has underpinned the trend in the automotive market. For vision systems, two different approaches have been addressed in the literature: color-based and shape-based approaches. For color-based approaches color segmentation and feature extraction are usually performed before detection, since the color and the shape of traffic signs are regulated by law in each country. In general, color segmentation applies thresholding to the input image in some color space. For example, in Figure 3.4 , a segmenta-

tion of the red color in HSI (Hue, Saturation, Intensity) space is illustrated[Møgelmose et al., 2012]. The HSI thresholding was also adopted in [Gil Jiménez et al., 2008] and [Vázquez-Reina et al., 2005] while others used adaptive RGB thresholding [Prisacariu et al., 2010] and [R. Timofte, K. Zimmermann, 2009]. The next step is feature extraction, edges are the most frequently used features [Houben, 2011, Ruta et al., 2011] . Differently, in [Vázquez-Reina et al., 2005], a boundary distance transform is used for feature extraction and an FFT of shape signatures are used in [Gil Jiménez et al., 2008].

Figure 3.4: Segmentation of red color in HSI space [Møgelmose et al., 2012]

Finally, the detection step consists in localizing the signs based on the extracted features. A very popular method that allows to detect specific shapes (circles, triangle, squares) is the Hough transform as done by [Ren et al., 2009]. Some papers propose the use of cascaded classifiers with Haar wavelet such as in [Baró et al., 2009]. A very good survey on traffic sign detection is presented in [Møgelmose et al., 2012].

Contrary to vision systems, the use of LiDARs for traffic sign detection is not very common and hardly addressed in the literature. The main challenge is the sparsity of the point cloud at larger distances. Although traffic signs are made of highly reflective materials which result high-intensity LiDAR returns, recognizing the shape and the information contained in the sign requires a very dense point cloud and is difficult to achieve in practice. Hence, camera and a lidar are sometimes used for the recognition task [Zhou and Deng, 2014]. The camera detects the traffic sign while the LiDAR allows to determine its 3D position. Lidar-only solutions solely rely on LiDAR reflectance in order to locate the traffic signs. For example, in [Riveiro et al., 2016], an intensity map is created, then, the pixels with high intensity values are grouped together using a clustering technique called DBSCAN [Martin Ester, Hans-Peter Kriegel, Jörg Sander, 1996]. In [Vu et al., 2013], a virtual scan image is created by projecting the point cloud on an image of a predetermined size. Then, principal component analysis (PCA) is applied to detect high-intensity planar surfaces corresponding to traffic signs.

D Concluding remarks

The state-of-art review shows that road perception is mainly used on vision systems for many reasons such as the information diversity provided by cameras (color, texture, shape, ...), the adapted algorithms for feature extraction and classification as well as integration and cost of camera sensors. However, many relevant information cannot be directly obtained from monocular vision such as depth, scale and the 3D coordinates of a point. Meanwhile, lidars recently gained attention since automotive-grade sensors already found their way into production for autonomous driving systems (e.g. SAE level 2+ for Audi A8). Combining LiDAR and vision enables the strength of each technology but also generates other challenges such as the geometric extrinsic calibration, time synchronization, as well as increasing the cost of the overall system. In all cases, there is no perfect solution for the road perception problem. The use-case is very important in choosing the sensor and the method to be implemented. The study of the state of the art leads us to a generic architecture for lidar-based road perception systems that is composed of two different steps (see the illustration in figure 3.5). The first step is to localize the region of interest (ROI) for the desired road feature, in order to limit the search space in the lidar data point cloud. For example, lane markings are located on the road, hence the segmentation of the road surface is typically performed beforehand. As for traffic signs, they are often located on the roadside and have regulated altitudes and shapes. The prior knowledge of the infrastructure can be used to target more precisely the area of interest. The second step is refinement where some geometric and reflectance cues are utilized to select the most likely candidates. In general, it consists in a model-based fitting with the objective of searching for well-defined geometric shapes like planes, lines, spheres. When the object of interest is very reflective (as in our case study), the LiDAR reflectance is used to strengthen the refinement step.

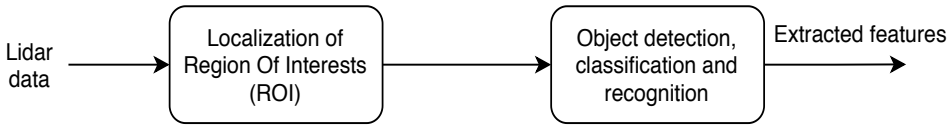
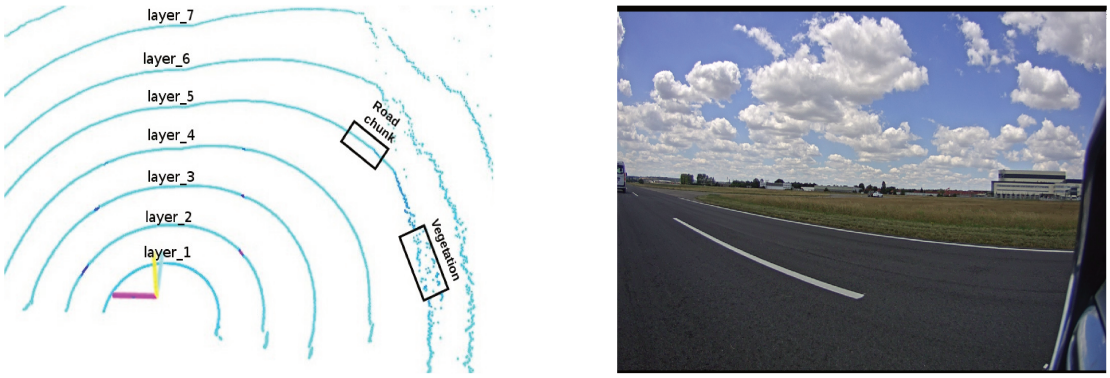


Figure 3.5: Road object detection and classification

Our proposed road perception system detects distinctive features on highway roads that are: lane markings, traffic signs, barriers and barrier reflectors. At first, we propose a road segmentation approach followed by a lane markings detection and tracking algorithm. Then, we use a reflectivity front grid map for the detection of traffic signs and barrier reflectors. This representation allows us to quickly identify the region of interest indicating the locations of the traffic signs and the barrier reflectors. Finally, a 2D horizontal height map is used to detect barriers.

3.2.2 Detection and segmentation of the road surface

In this subsection, we propose a road segmentation approach based on geometric analysis of the set of the returned points from one layer of a 360 ° multilayer rotating LiDAR (Velodyne VLP32-C). The observation of the returned 3D scan in one single frame shows that the rotating layers draw circular arcs when hitting the road. This is explained by the flatness and the smoothness of highway road surfaces. In contrast, outside road boundaries, the layer points are scattered as they encounter non flat and non smooth surfaces such as vegetations, gravel, etc. This is depicted in Figure 3.6. In Figure 3.6a, we highlight two different layer chunks, one for road surface and another for non road region (vegetations). It is apparent that the road chunk is geometrically more structured and close to a circular arc shape than the layer chunk that corresponds to vegetations. Figure 3.6b illustrates the corresponding context camera image. The goal is to design a score function that takes as input a subset of points from the lidar layer and finds a score value that indicates to which extent the described geometry fits a circular arc shape. The design of such function is described in the next paragraph.



(a) raw LiDAR data **(b)** raw camera image

Figure 3.6: Illustration of lidar layer dispersion outside road boundaries

A Calculation of a circular arc similarity measure

Let us take the example of one LiDAR layer identified by its vertical angle θ. All the calculations hereafter are also applicable to the rest of the layers. Let $\mathcal{L}$ be the set of points obtained from a complete rotation of the layer:

$$\mathcal{L} = \{p_1, p_2, ..., p_N\} \tag{3.1}$$

The coordinates of a point p_k are given in the spherical coordinate system:

$$p_k = (r_k, \theta, \phi_k) \tag{3.2}$$
$$0 \leq \phi_k \leq 2\,\pi \tag{3.3}$$

θ is the vertical beam angle and is constant at a given lidar layer, r_k is the detected point range and ϕ_k is the azimuth angle. The first step is to group the set $\mathcal{L}$ by subdividing the azimuth angle (ϕ) into a set of chunks s_j. A chunk is defined by a start angle ϕ_{start}, an end angle ϕ_{end} and an angular resolution ϕ_{end} that defines its size (i.e. the number of chunk points). Consequently, the set $\mathcal{L}$ can be defined as follows:

$$\mathcal{L} = \begin{cases} s_j & = \{p_i = (r_i, \theta, \phi_i) \mid \phi^j_{start} \leq \phi_i \leq \phi^j_{end}\} \\ \phi^j_{start} & = j \times \phi_{res} \\ \phi^j_{end} & = (j+1) \times \phi_{res} \; and \; j \in [0, ..N_c = \lfloor \frac{2\pi}{\phi_{res}} \rceil] \end{cases} \tag{3.4}$$

where the symbol $\lfloor x \rceil$ is the integer part (truncation) of the float x and N_c is the number of chunks. The second step is re-structuring where the azimuth space is divided into four quadrants: $Q_1 := [0, \frac{\pi}{2}]$, $Q_2 := [\frac{\pi}{2}, \pi]$, $Q_3 := [\pi, \frac{3\pi}{2}]$ *and* $Q_4 := [\frac{3\pi}{2}, 2\pi]$. Intuitively, we can assume that there exist at most one road boundary in each quadrant region. Hence, the road segmentation algorithm can be executed for the four quadrants in parallel, therefore, to reduce the computational time.

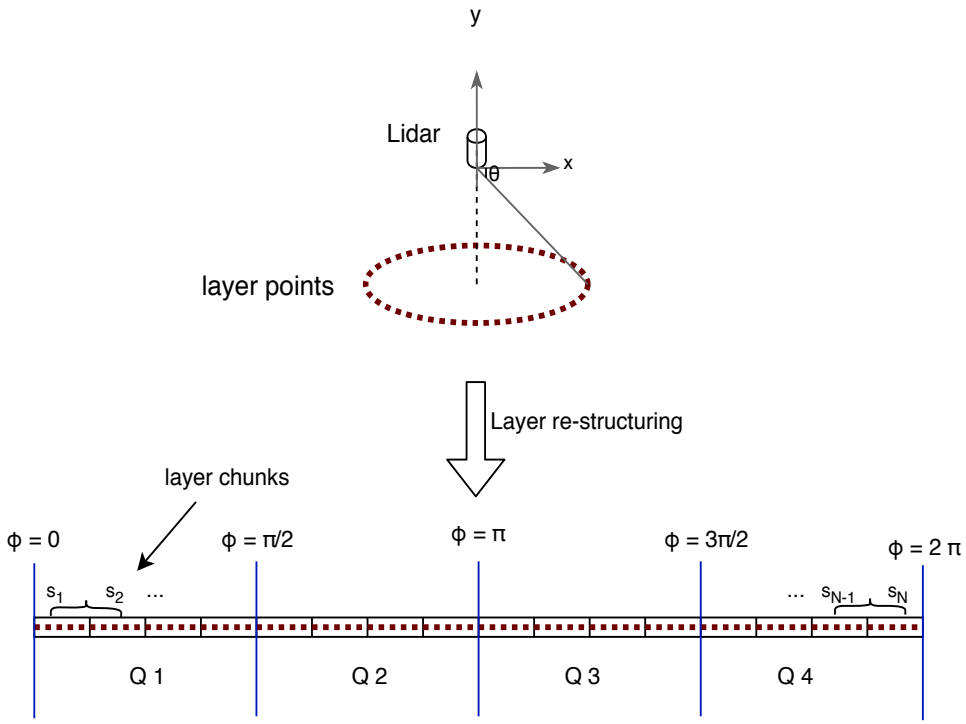


Figure 3.7: Re-structuring of layer points into contiguous chunks

The classification of layer chunks into road and non road is obtained by fitting the chunk s_j into a circular arc model. Hence, a similarity measure (score) $\lambda(\theta, s_j)$ is designed for this purpose. Except for the intrinsic parameters of the LiDAR, our method does not require any other parameter such as the mounting height of the LiDAR as in [Hata and Wolf, 2014]. Indeed, all the points of a road chunk lie within the same horizontal plane, which is the road surface plane as illustrated in Figure 3.8. We denote by z_μ the height of the plane which s_j belongs to, and we estimate it as follows:

$$z_\mu = \frac{1}{N_s} \sum_{p:=(r,\theta,\phi)\in s_j} z_p, \text{ where, } z_p = r \times sin(\theta) \tag{3.5}$$

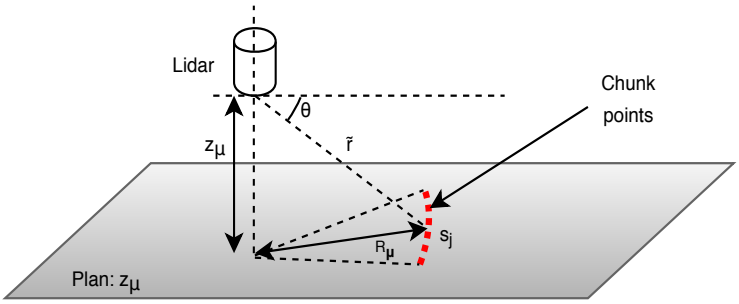


Figure 3.8: Theoretical distribution of a road chunk points

N_s is the number of chunk points. We also denote by $\mathcal{C}_j$ the expected circular arc described by s_j if it is completely on the road surface. The first parameter of $\mathcal{C}_j$ is z_μ. The second parameter is the radius R_μ which is calculated with respect to the beam vertical angle θ (Figure 3.8):

$$R_\mu = z_\mu \times \cot(\theta) \tag{3.6}$$

where cot denotes the cotangent function. For each point $p_i = (r_i, \theta, \phi_i) \in s_j$ there exists a corresponding point $\tilde{p}_i = (\tilde{r}_i, \theta, \phi_i) \in \mathcal{C}_j$ and for which we have:

$$\tilde{r} = \sqrt{R_\mu^2 + z_\mu^2} \tag{3.7}$$

The similarity measure λ is calculated by computing the normalized sum of the distances of p_i to $\tilde{p}_i$ and is given as follows:

$$\lambda(\theta, s_j) = \frac{1}{N} \sum_{p \in s_j} dist(p, \mathcal{C}_j) \tag{3.8}$$

where *dist* is the distance in the spherical coordinates and is given by:

$$\|p - \tilde{p}\| = \sqrt{r^2 + \tilde{r}^2 - 2r\tilde{r}} \tag{3.9}$$

Consequently:

$$dist(p, \mathcal{C}_j) = \sqrt{r^2 + R_\mu^2 + z_\mu^2 - 2r\sqrt{R_\mu^2 + z_\mu^2}} \tag{3.10}$$

B Implementation of Forward-Backward iterative search

For simplicity, the notation $\lambda(\theta, s_j)$ is simply substituted by λ_j. We stored the λ values in four different arrays (one array per quadrant). For each array, we implemented an iterative search process in order to keep only road chunks. As mentioned, we assumed that within each quadrant, the layer intercepts the road boundaries once at most. Consequently, the quadrant chunks can be split into two different adjacent sections: road surface and non road surface. For Q_1 and Q_3, we implemented a *forward-search* strategy, starting from the first chunk and iterating until crossing the first chunk whose λ value is greater than a threshold T. Inversely, for Q_2 and Q_4, we implemented a *backward-search*, starting from the last chunk and reversely iterating over chunks until crossing the first value of $\lambda > T$. An illustration of this process is given in Figure 3.9.

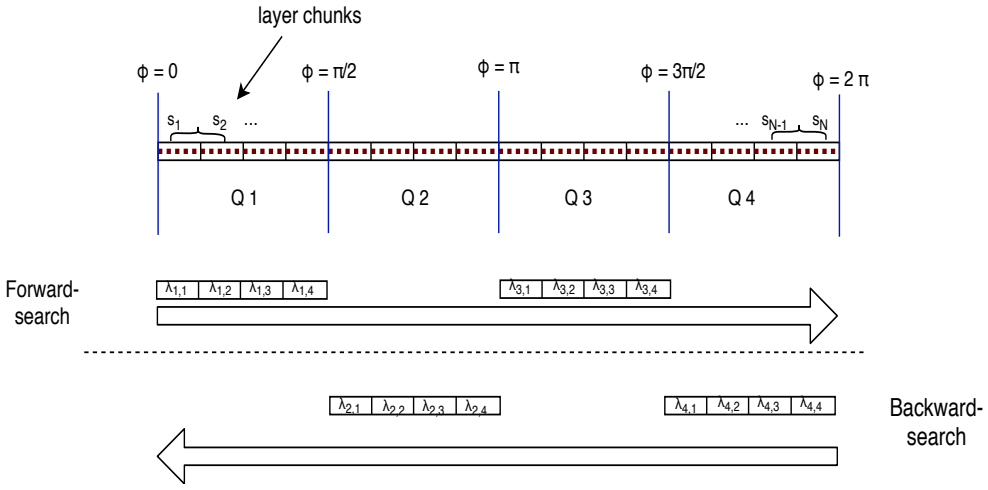


Figure 3.9: Forward and Backward search process

In order to set a proper value of T,we compute a histogram of λ values for all layers that hit the road surface. Assuming that the majority of layer chunks lies within the road surface, the histogram peak can be used to set the value of T. In Figure 3.10, the histogram shows that the peak is reached for $\lambda \leq 0.1$. This value can be used to set the threshold T.

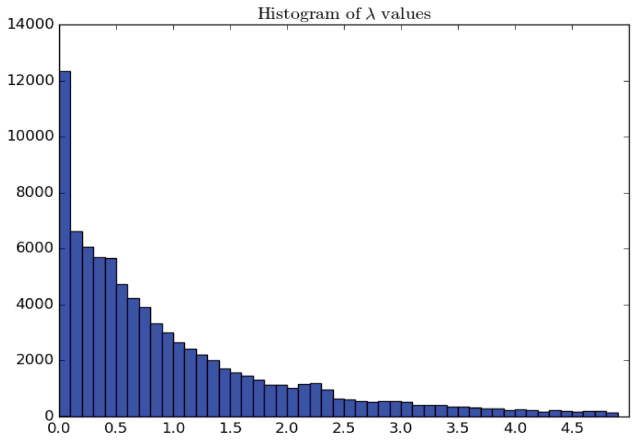


Figure 3.10: λ-value histogram for one LiDAR frame

3.2.3 Detection of lane markings

A Creation of a reflectivity grid map

A two-dimensional horizontal reflectivity grid map is created by projecting the reflectivity data of the segmented road surface points. The grid map is centered around the ego vehicle frame and has a cell size equals to $0.1m \times 0.1m$. Since the main goal is to use the lines to perform lateral localization, the detection range can be limited in space as we only need detections in the vicinity of the ego vehicle. Consequently, a grid map dimensions of := $(x, y) \in [-10\ m, 10\ m] \times [-10\ m, 10\ m]$ is considered to be sufficient. The attributed value to each grid cell is the mean reflectivity value from all the points that fall in it. Then, a binary image is generated by applying a threshold to the reflectivity grid map. The main purpose of the binary image is to detect lane markings by extracting straight lines. For our use case, the line assumption is locally (at a short range) very strong since highway road profiles are designed to have low curvatures.

Lane markings are detected by applying the Hough Transform (HT) to the binary image. This technique is described in details in the following paragraph. The Hough transform is a very simple, yet efficient, technique to detect shapes (line, circle, planes, etc). However, unlike image data, the obtained binary image can be very sparse due to the relatively low resolution of the LiDAR and to some missing reflectivity returns. Since the main idea of HT is based on calculating scores to extract lines, the sparsity of the binary image may highly impact the accuracy and the robustness of the detection process. To mitigate this problem, we applied dilation operator which is a shift-invariant (translation invariant) operator, equivalent to Minkowski addition. This operation usually uses a structuring element for probing and expanding the shapes contained in the input (binary) image. To illustrate the dilation principle, a 3x3 struc-

tural element with one value is applied to an input binary array in Figure 3.11. As we can see, the applied kernel tries to connect isolated points and to fill the gaps between them. This is very practical when we want to homogenize a continuous shape for a better detection output. The dilation can be applied as much as necessary, thought, too many iterations would result in overgrowing the shape in the image. For example, too many repetitive dilations to the input image in Figure 3.11 will turn all white pixels into black pixels.

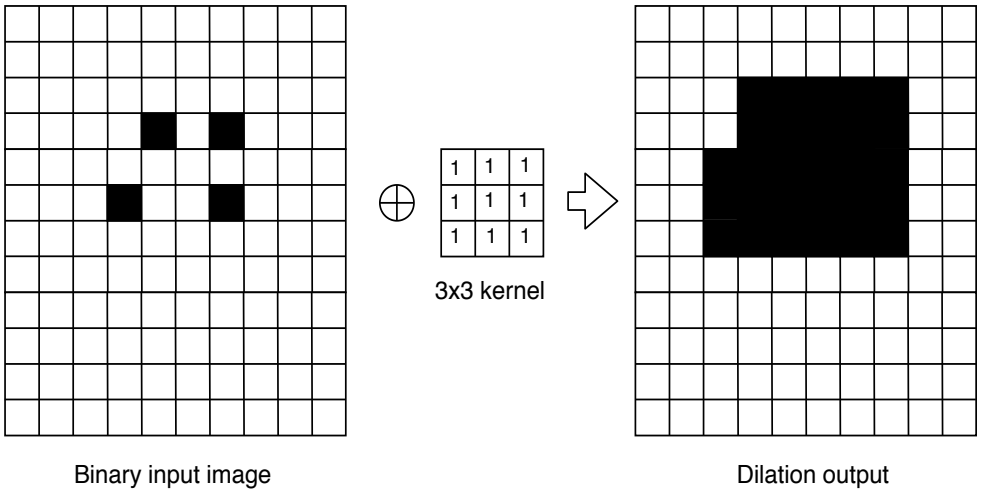
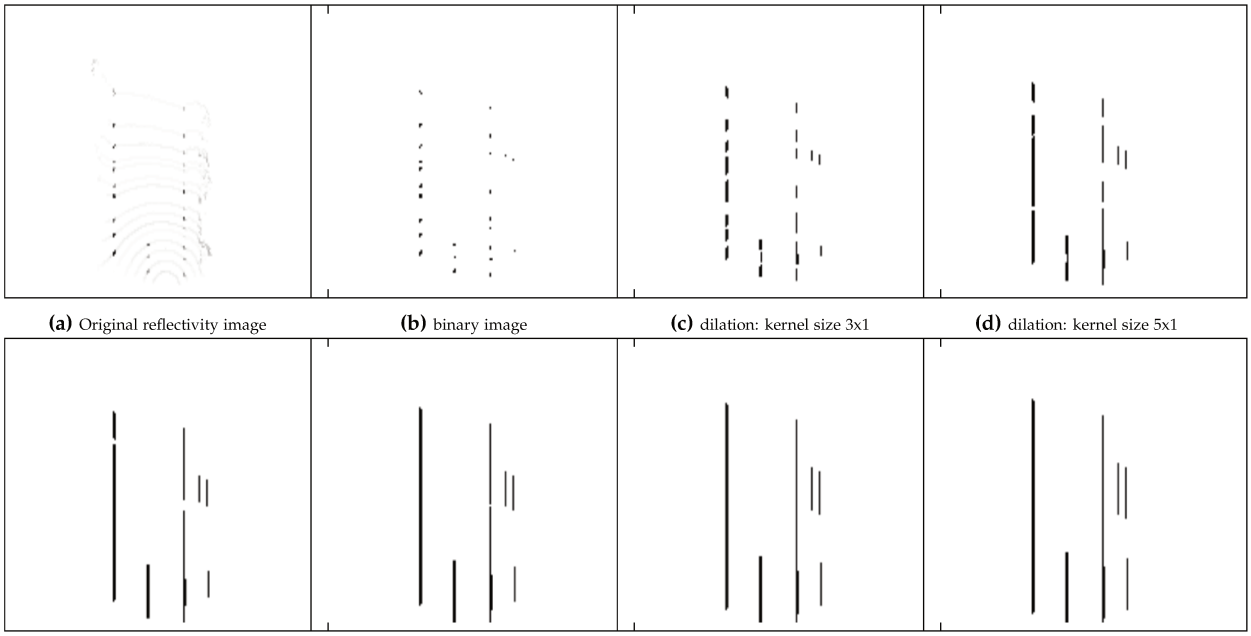


Figure 3.11: Dilation Output for 3x3 Structural element

In some cases, we want to dilate a specific direction in the image. It is therefore more appropriate to apply one-directional kernels instead of two dimensional ones. For instance, in our application, we want to look for vertical lines considering that lane markings are parallel to the driving direction.Thus, we applied a $n \times 1$ kernel to the binary image in order to improve the quality of the lines that we want to detect. Choosing a large kernel size allows to connect very distant pixels together but may also extend isolated points into segments, which can potentially result in false detected lines. The results of the dilation operation which was repeated twice with different kernel sizes, and applied to reflectivity grid map are illustrated in Figure 3.12. As we can be seen, kernel sizes larger than 5×1) cause the appearance of many line segments that were originally isolated points. Having tested different kernels, we concluded that the kernel size 5×1 enhances the line quality without adding false detections. Finally, Hough transform is implemented to extract the straight lines.

B Line detection using the Hough Transform (HT)

The Hough transform is a global method for extracting straight lines in a 2D image. The principle mainly relies on the difference between Cartesian and polar forms of a line. It has been shown that a line in the Cartesian space can be mapped to a point



(a) Original reflectivity image **(b)** binary image **(c)** dilation: kernel size 3x1 **(d)** dilation: kernel size 5x1

(e) dilation: kernel size 7x1 **(f)** dilation: kernel size 9x1 **(g)** dilation: kernel size 11x1 **(h)** dilation: kernel size 13x1

Figure 3.12: Results of dilation for different kernel sizes.

in the parameter space (r, θ). Given a single point (x,y), the set of all straight lines through this point are mapped to unique sinusoidal curve in the (r, θ) plane. Varying (x, y) along one straight line will produce sinusoids which intersect at the (r, θ) for that line. This intersection point is the polar representation of the line. The difference between the parameter and the Cartesian spaces is shown in figure 3.13.

This description is reflected into what is called an accumulator. An accumulator is a 2D matrix where the rows stand for the discretized r values and the columns for the discretized θ values. The accumulator is often initialized with zeros. A pixel of value equals to one votes for all the lines that pass through it. By doing this to all the pixels in the image, the resulted matrix is filled with numbers that indicate the score of pixels that are mapped to the the same accumulator cell. The existing lines are detected by searching maximum values in the accumulator. This principle is used in our binary image to detect lane markings. Since not all detected lines can be considered as a lane marking, we only kept the lines that are nearly parallel to the driving direction and that are parallel to each other. We also imposed a minimum distance between two extracted maximum locations. Practically , this avoids having multiple detections for the same line. Finally, the set of retained lines are given in polar coordinates as follows:

$$l_i = \{(r_i, \theta_i), i \in [1..N]\} \tag{3.11}$$

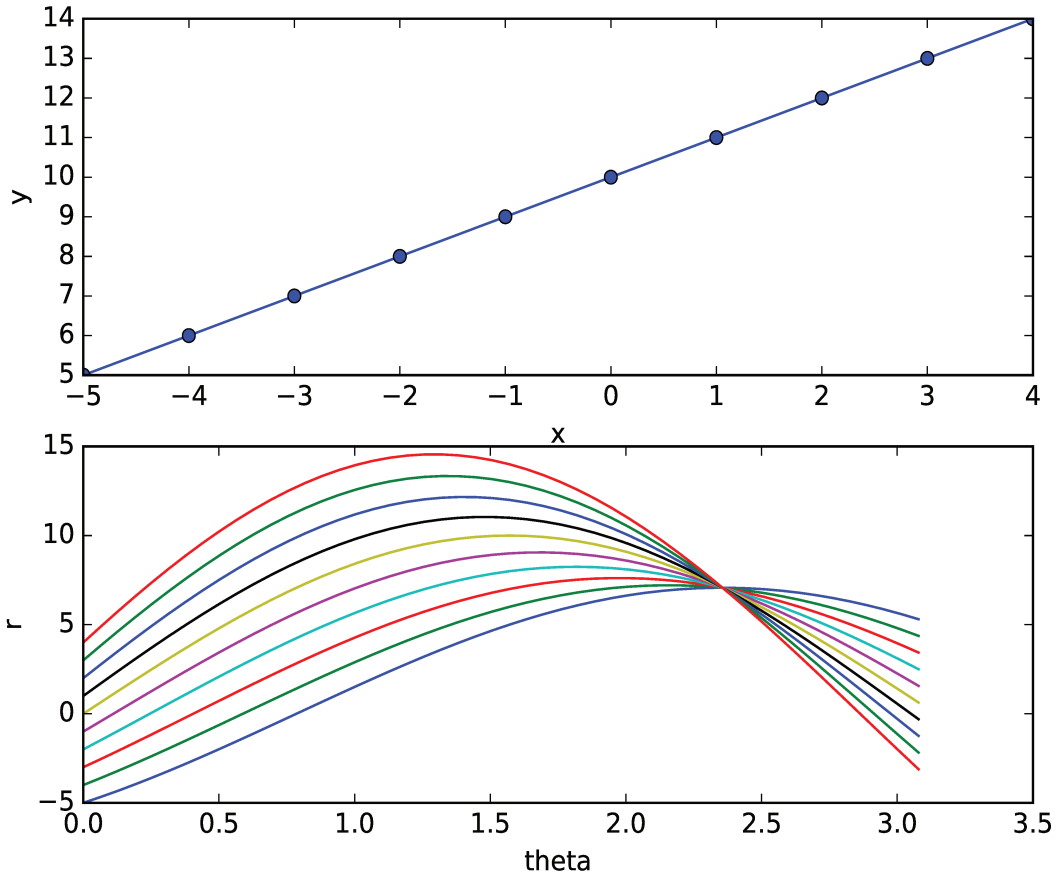


Figure 3.13: Line mapping from Cartesian space to parameter space

C Tracking with Kalman filter

In order to deal with missed detections and occlusions, we implemented a tracking algorithm based on a standard Kalman filter [Kalman, 1960]. The algorithm is a composed of a prediction step and a correction (or update) step. In the prediction step, a motion model is used to update the state of the tracked object. In the correction step, the prediction is updated on the basis of the difference between measured and predicted states. Let $\mathbf{x_k}$ be the state vector of a tracked object at time k be $\mathbf{x_k}$. It is assumed that the system evolution functions are linear with normally distributed additive noise given as below:

$$P(\mathbf{x_k} \mid \mathbf{x_{k-1}}, \mathbf{u_k}) = A \times \mathbf{x_{k-1}} + B \times \mathbf{u_k} + \epsilon_{motion} \tag{3.12}$$

where A and B are the transition matrices that govern the state vector $\mathbf{x}$ and the command vector $\mathbf{u}$, respectively. ϵ_{motion} is a normal distribution with a zero-mean covariance matrix $\epsilon_{motion} \sim \mathcal{N}(0, \Sigma_{motion})$. Moreover, the equation of the observation $\mathbf{z_k}$ is given as:

$$P(\mathbf{z_k} \mid \mathbf{x_k}) = H \times \mathbf{x_k} + \nu_{obs} \tag{3.13}$$

where H is the observation matrix and ν_{obs} is a normal distribution with a zero-mean covariance matrix $\nu_{motion} \sim \mathcal{N}(0, \Sigma_{obs})$. The standard implementation of kalman filter to estimate the state vector and covariance matrix $(\mathbf{x_k}, \Sigma_k)$ is described as follows:

$$\begin{cases} \mathbf{x_k} & = A \times \mathbf{x_{k-1}} + B \times \mathbf{u_{k-1}} \\ \Sigma_k & = \Sigma_{k-1} + \Sigma_{motion} \\ \mathbf{K_k} & = \Sigma_k \times H^T \times (H \times \Sigma_k \times H^T + \Sigma_{obs})^{-1} \\ \mathbf{x_k} & = \mathbf{x_{k-1}} + \mathbf{K_k} \times (\mathbf{z_k} - H \times \mathbf{x_k}) \\ \Sigma_k & = (I - \mathbf{K_k} \times H) \times \Sigma_k \end{cases} \tag{3.14}$$

where $\mathbf{K}$ is called the Kalman gain and I is the identity matrix.

In our case, the inertial inputs are the velocity v and the yaw rate w. They are provided by an inertial measurement unit (IMU) from u-ublox B78-ADR. The command vector is given as $\mathbf{u} = [\Delta y, \Delta\theta]$ where Δy and $\Delta\theta$ are respectively the lateral displacement and the change of heading between two time steps. the state vector is the polar representation of the line $\mathbf{x} = [r, \theta]$. The command vector u is a function of v and w and is given by:

$$\Delta y = v \times dt \times sin(w \times dt) \tag{3.15}$$

$$\Delta\theta = w \times dt \tag{3.16}$$

In addition, the adopted motion model is given by:

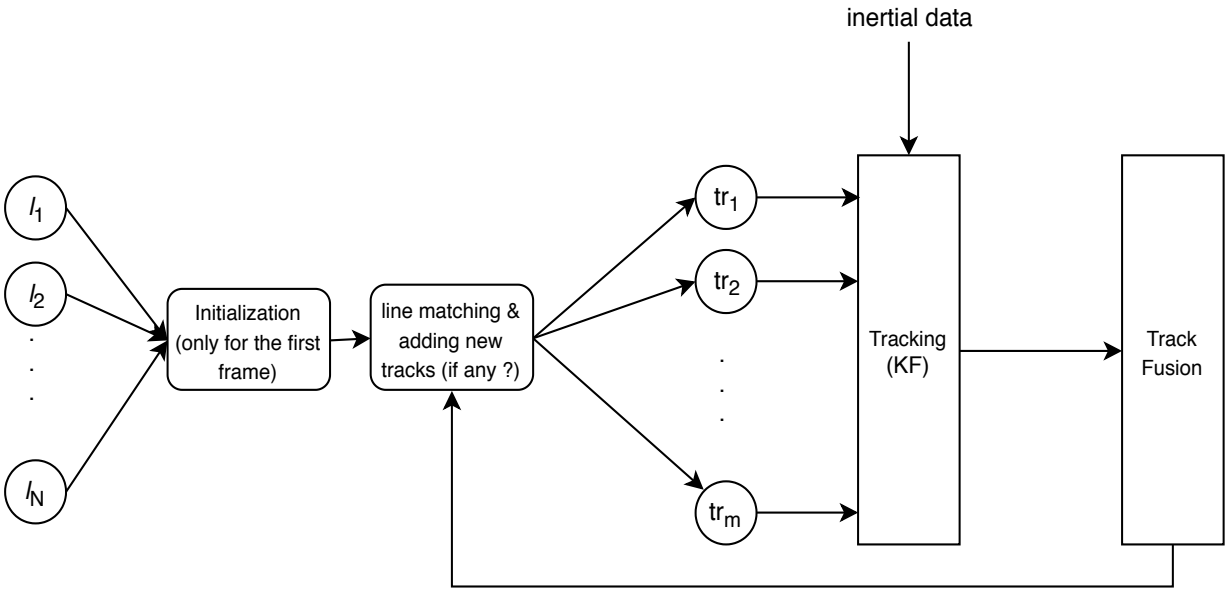


Figure 3.14: Line tracking architecture

$$r_k = r_{k-1} - \Delta y \tag{3.17}$$
$$\theta_k = \theta_{k-1} - \Delta\theta \tag{3.18}$$

The values of r and θ are interpreted as the ego vehicle lateral distance and heading in relation to the line, respectively. By undergoing the movement described by (v, w), the lateral vehicle displacement in the lateral direction can be given by $\Delta y = v \times dt \times \sin(w \times dt)$. This displacement induces a change in the value of r given by Equation 3.17. Similarly, the yaw rate w produces also a change of heading $\Delta\theta = dt \times w$ given by Equation 3.18. The command vector was not set directly to (v, w) in order to keep a linear representation of the motion model. Otherwise, the motion model will include non linearities. Nevertheless, non-linear versions of Kalman filter could be used where the motion model is often approximated by the first degree of the Taylor expansion could be used (e.g. the Extended Kalman Filter (EKF) [Julier and Uhlmann, 1997]). We assume that, in our case, the linear version is sufficient given the driving conditions on highway roads. The observation matrix is equal to identity which means all the state variables are observable. To summarize, the prediction and observation matrices are given by:

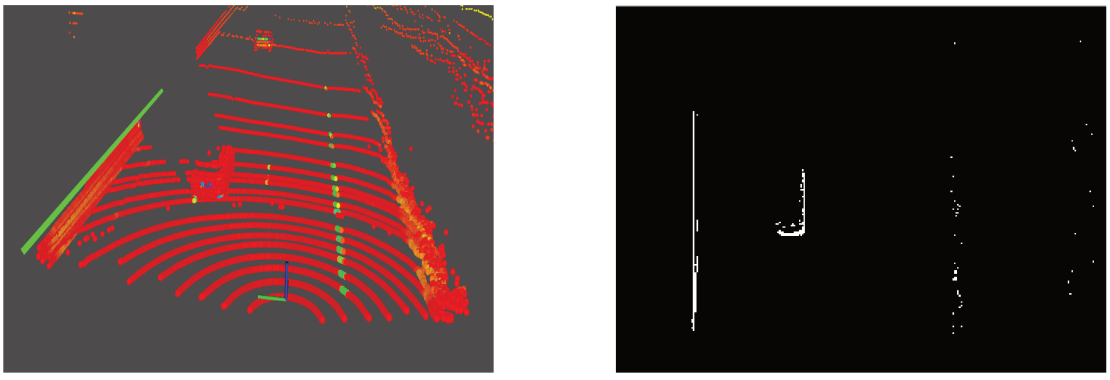
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} -1 \\ -1 \end{bmatrix},\ H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.19}$$

Figure 3.14 shows an overview of the different steps in our tracking algorithm. In the initialization step, a «track» is created for each detected line. A track is an implementation of the above equations of Kalman filter. In the second stage, a track-to-measurement association is implemented by calculating the distances between track

state vectors and detection state vectors. The goal is to find correspondences between the new measurements and the set of active tracks. A new measurement is associated to a given track if the relative distance is less than half a lane width. If no association is found, the track is kept for a fixed number of consecutive frames until finding a measurement correspondence, otherwise it is discarded. In addition, any new detection that was not tracked before is added with a new track initialization. In the last stage, the track fusion permanently checks if the states of any two tracks are getting close to each other, in order to combine them into one single track (if it is the case). This typically occurs in cases of fork or road insertion. In the latter case, the corresponding tracks (lines) are fused into one track by simply calculating the mean of their state vectors and covariance matrices.

3.2.4 Detection of highway barriers

One key characteristic of highway roads is the permanent presence of median safety barriers. These barriers are designed to safely separate both sides of highways. Geometrically, barriers have higher heights with respect to the road, which make them easily detectable by LiDAR. In the proposed method, we used LiDAR mounted on the bumper as it provides more density in that area. We created a 2D horizontal height map whose cells store the height difference between the highest and the lowest altitudes of 3D points that fall within each cell. An example of a height map is depicted in Figure 3.15b and is constructed from the point cloud in Figure 3.15a. The constructed height map is centered around the ego vehicle frame and is limited in the x and y directions, i.e $|x| \leq x_{max} = 20\ m$ and $|y| \leq y_{min} = 15\ m$. The assumptions made to detect the median barrier are the following:



(a) Result of barrier detection

(b) Constructed height map

Figure 3.15: Detection of highway barrier

- The barrier can be modeled by a straight line in the height map. Again, we applied the Hough transformation to detect straight lines and kept the same line representation (r, θ).

- It is located at the left hand side of the driving direction, i.e: $r \geq 0$ (but can be adapted when the road has barriers on both sides).
- It is considered as the longest line in the height map.

Sometimes, the third assumption can be violated by moving obstacles (vehicles, trucks, etc) that occlude the median barrier (Figure 3.15b). Thus, we implemented the Kalman Filter tracking algorithm to filter out the temporary presence of these obstacles. The result of the median barrier detection is depicted by the green line in Figure 3.15a.

A Lane index inference from detected barrier

The main use of the median barrier detection is to estimate the lane number which the driving vehicle belongs to. Assuming that the vehicle distance to the barrier is r and that the lane width is w, the lane index can therefore be approximated by:

$$Lane_{index} = \lfloor \frac{r}{w} \rceil \tag{3.20}$$

Indeed, in actual highway roads the distance to the barrier is the sum of the distance to the far left lane marking r_{lfar} and the left highway strip width w_{st}

$$r = r_{lfar} + w_{st} + \sigma_{det} \tag{3.21}$$

Where σ_{det} is the barrier detection error. The lane index inferred from the detected barrier is therefore:

$$Lane_{index} = \overbrace{\lfloor \frac{r_{farl}}{w} \rceil}^{true\ index} + \overbrace{\lfloor \frac{w_{st} + \sigma_{det}}{w} \rceil}^{error} \tag{3.22}$$

Where $\lfloor \frac{r_{farl}}{w} \rceil$ is the true lane number and $\lfloor \frac{w_{st}+\sigma_{det}}{w} \rceil$ is as an error term. On one hand, the strip width on French highways is at maximum 2.5 m and the lane width is approximately 3.5 m. On the other hand, the value of σ_{det} is considered to be small since the accuracy of LiDAR points is in the order of a few centimeters. As a result, we can assume $w_{st} + \sigma_{det} < w$ which yields $\lfloor \frac{w_{st}+\sigma_{det}}{w} \rceil = 0$.

3.2.5 Detection of traffic signs and guardrail reflector

For traffic signs and guardrail reflectors, we use the same principle as for the detection of lane markings. However, instead of creating a horizontal reflectivity grid map, we created a frontal reflectivity grid map. A frontal grid is a discretization of the space (θ, ϕ) in the spherical coordinates. Let respectively be (θ_{res}, ϕ_{res}) the corresponding angular resolutions and $[\theta_{min}, \theta_{max}]$ and $[\phi_{min}, \phi_{max}]$ the vertical and horizontal field of views . The angular position of a grid cell $m_{i,j}$ is consequently given by:

$$\phi_i = \phi_{min} + i \times \phi_{res},\ i \in [\ 0\ ..\ \lfloor \frac{\phi_{max} - \phi_{min}}{\phi_{res}} \rceil\ ] \tag{3.23}$$

$$\theta_j = \theta_{min} + j \times \theta_{res},\ j \in [\ 0\ ..\ \lfloor \frac{\theta_{max} - \theta_{min}}{\theta_{res}} \rceil\ ] \tag{3.24}$$

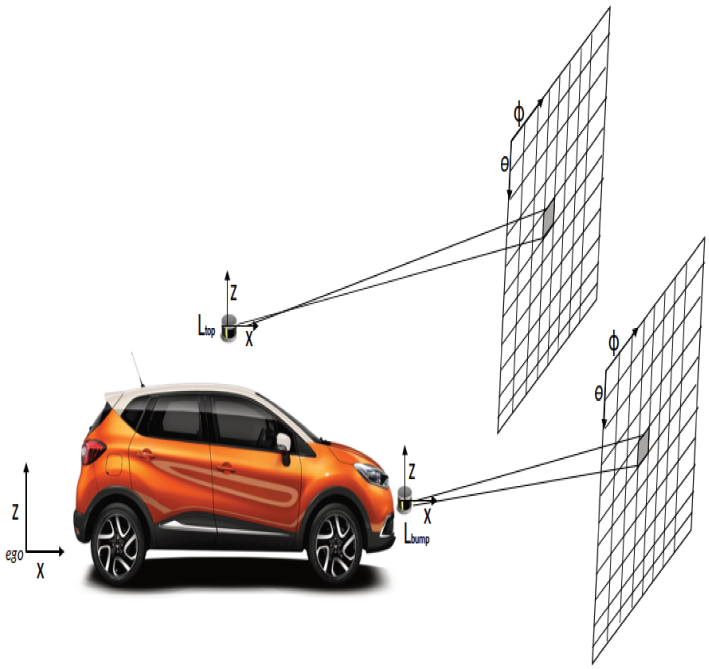


Figure 3.16: Projection of LiDAR data into polar front grids

Two different types of data are stored in each grid cell $m_{i,j}$: the 3D points returned from LiDAR layers that overlap $m_{i,j}$ and their average reflectivity value. First of all, it is important to note that this grid representation allows us to easily extract the set of 3D points of a local window in the grid without having, each time, to iterate over the whole point cloud. Second, a 2D image can be created by mapping the cell reflectivity values into image pixels. Since traffic signs and guard guardrail reflectors are made of very reflective materials, the frontal reflectivity grid map helps identifying the regions where these objects are located.

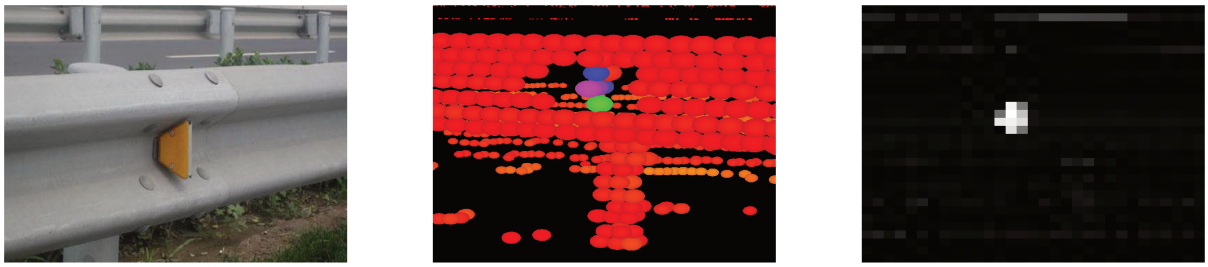
Because the value of θ_{max} is constrained by the maximum vertical angle of the LiDAR (2° for Velodyne VLP32-C), the bumper-mounted LiDAR is not able to sufficiently detect the traffic signs. The bumper-mounted LiDAR covers the lower part of the space (near the ground), hence it is more adapted to detect guardrail reflectors. On the contrary, roof-mounted LiDAR provides more dense data in the upper part of the space, therefore, we used it to detect traffic signs. We created two different front grid maps,

one for each LiDAR, we call them for simplicity the top and bumper grids. We did not adopt to fuse the two LiDARs into one grid in order to avoid time synchronization and the extrinsic geometric calibrations of the sensors. Figure 3.16 illustrates the mounting positions of the used LiDARs.

A Detection of regions of interest (ROI)s

We adopted two different techniques to detect regions of interest for traffic signs and guard guardrail reflectors. For traffic signs, we created a binary image by thresholding the top grid. Then, a closed contour detector is implemented to locate the ROIs. In image processing, a contour is a set of continuous connected pixels with the same intensity level which defines a shape boundary. However, unlike image data, the sparsity of LiDAR data induces unfilled gaps (pixels without any attributed reflectivity value) in the road sign shape on the constructed binary image. To overcome this problem, we applied a dilation operation with a 3×3 kernel to enhance the binary image, then, we used an OpenCV implementation of the method proposed in [S.Suzuki, 1985] to detect closed contours.

As for guardrail reflectors, the detection technique is different than the one used for traffic signs. The reason behind this lies essentially on their (i.e. guardrail reflectors) small sizes compared to traffic signs. Indeed, applying a dilation followed by closed contour detector may introduce false detections due to isolated reflective points. Instead, we implemented a template matching technique using Normalized Cross Correlation (NCC) derived from Cauchy-Schwarz inequality to measure the similarities as described in [Sarvaiya et al., 2009]. An example of a guardrail reflector in our highway-like test track and the corresponding 3D points are respectively illustrated in Figure 3.17a. The used template is a well-chosen image patch centered around a reflectivity reflector area as depicted in Figure 3.17c. In the following, we present a short overview of template matching techniques with particular focus on the NCC.



(a) Retro-reflector marker **(b)** Reflector point returns **(c)** Reflector template

Figure 3.17: Illustration of a retro-reflector in real highway road as well as in point cloud data.

Template matching is a technique used to find a known pattern in an input image. It is widely used in the field of object detection and recognition. Its mathematical formulation is based on the design of a function that measures the similarities between

the input image and the used template. There are mainly two different approaches for template matching: Feature-based approaches [Manjunath et al., 1992] and template-based approaches [Sarvaiya et al., 2009]. The choice of the matching technique depends on the nature of the image and the problem to be solved. Feature-based techniques attempt to find correspondences between features such as points, curves and edges. A similarity measure is applied to feature descriptors such as SIFT, SURF [Bay et al.,] and ORB [Rublee et al., 2011] in order to find correspondences.

Template-based matching operates directly on the pixel values by comparing intensity values of the template with intensity values of the input image. Let $I(x, y)$ be the input image and $t(u, v)$ the image patch. Many similarity measures are used in template matching. For example, a first option is to compute the sum of squared differences (SSD):

$$SSD = \sum_{x,y} [I(x + u_0, y + v_0) - t(x, y)]^2 \tag{3.25}$$

The template matches are obtained by searching for minimum values of the SSD. A second option is the cross-correlation. In the case of template matching, it is often calculated by sliding the template over the input image and searching for the maximum values which correspond to potential template matches. The formulation of cross-correlation is given as follows:

$$CC = \sum_{x,y} I(x + u_0, y + v_0) \times t(x, y) \tag{3.26}$$

One problem with cross-correlation occurs when a template is matched to an input image under varying illumination intensities. An improved version of cross-correlation is to normalize the pixels in the windows before comparing them. This is known as Normalized Cross-Correlation (NCC). The mathematical formulation of NCC is given as follows:

$$NCC = \sum_{x,y} \frac{(I(x + u_0, y + v_0) - I_0) \times (t(x, y) - t_0)}{\sqrt{\sum_{x,y}(I(x + u_0, y + v_0) - I_0)^2} \sqrt{\sum_{x,y}(t(x, y) - t_0)^2}} \tag{3.27}$$

where,

$$I_0 = \frac{1}{N} \sum_{x,y} I(x + u_0, y + v_0) \tag{3.28}$$

$$t_0 = \frac{1}{N} \sum_{x,y} t(x, y) \tag{3.29}$$

N is the number of pixels in the image template. The idea here is to normalize the pixels inside the local window by subtracting the mean pixel values (denoted by I_0 and t_0) and dividing by their standard deviations. Feature-based approaches are

very often used when the structural information is more relevant than the intensity information. For template matching, the intensity data is more relevant. In our case, since the sought reflector is more distinguished by its intensity values than its geometric form, we applied a template matching algorithm based on normalized cross correlation. However, in practical situations, many false detections can occur due to passing cars (license plates), isolated reflective points, etc. To further refine the set of detections, we used a model-based geometric fitting to discard ROIs that do not match a specific shape. The adopted method is described in the next paragraph.

B Refinement of ROI candidates

Unlike image data where colors and texture can be exploited, the geometric approaches remain the most reliable to search for specific shapes within LiDAR point clouds. For traffic signs and guardrail reflectors, we assume that they can be modeled by planar surfaces. The idea is to extract, for each detected ROI, the corresponding set of the 3D points from the grid maps. We apply a RANSAC-based plane fitting algorithm in order to only keep plane surfaces. RANSAC is the abbreviation of Random Sample Consensus. It is a parameter estimation algorithm proposed by [Fischler and Bolles, 1981] that is robust in the presence of a large proportion of outliers. The parameter estimation is performed by the minimum required number of points to construct a plane model. Thus, we need at minimum three non-coplanar points. Then, a score function is used to check to which extent the estimated parameters fit to the remaining points. For instance, one can use the number of inliers as a score value . The outline of RANSAC is given in in Algorithm 1:

Data: 3D point cloud: P
Result: Best plane parameters that fit P
1- Initialization: let N be the number of iterations ;
2- **while** *num of iteration* < N **do**
    2.1) Randomly select a sample of three points from P;
    2.2) Fit a plane model to these points ;
    2.3) Compute the distance of all other points to this model;
    2.4) Construct the inlier set (i.e. count the number of points whose distance from the model < d);
    2.5) Store these inliers;
    2.6) increment *num of iteration*;
end
3) The set with the maximum number of inliers is chosen as a solution to the problem ;
4) Estimate the model using all the inliers;

Algorithm 1: RANSAC-based plane estimation algorithm

The number of iterations N that is necessary to calculate a correct solution is com-

puted by:

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.30}$$

where s is the minimum number of points which the model can be estimated from, ϵ is the tolerated percentage of outliers in the data and p is the requested probability of success.

	Detection of traffic signs	**Detection of guardrail reflectors**
Minimum number of inlier (points)	70	50
Maximum detection distance (meters)	30	30
The normal vector: $n = (n_x, n_y, n_z)$	Parallel to the driving direction: $n_y \approx n_z \approx 0$	Perpendicular to the driving direction: $n_x \approx n_z \approx 0$

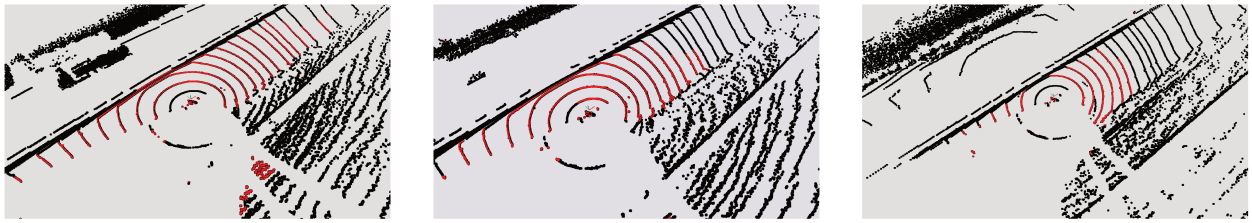
In our case, the requested probability is $p = 99\%$, the percentage of outliers is $\epsilon = 50\%$ and the minimum number of required points is $s = 3$. The minimum number of iterations is therefore equal to $N = 35$ iterations. The inlier points are firstly used to calculate the normal vector of the plane surface and, secondly to calculate the centroid point. The first selection criterion is applied to the estimated normal vectors of the estimated plane surface. Indeed, the normal vectors for traffic signs are assumed to be parallel to the driving direction, whereas guard guardrail reflectors are assumed to belong to planar surfaces perpendicular to the driving direction. This is shown in the last column of Table **??**. The second selection criterion concerns the calculated centroid point of the plane. Finally, to improve the reliability of the detections, we fixed a detection range of 30 meters. Indeed, the LiDAR point cloud is more dense at closer distances to the sensor..

3.2.6 Qualitative results and discussion

A Road surface segmentation

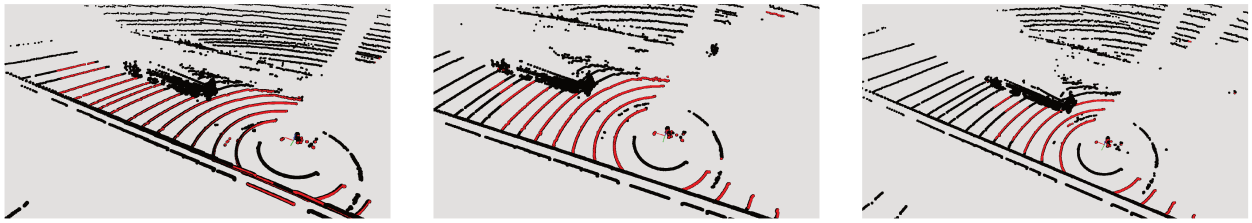
We present in Figures 3.18 and 3.19 results of the proposed road surface detection in different scenarios: an obstacle-free and with moving truck scenarios. We tested our algorithm with different chunk resolutions ϕ_{res}: 2, 5 and 9 degrees. On one hand, increasing the size of the chunk improves the model fitting score (i.e the chunk contains significant number of points) and is therefore less sensitive to false detections. However, this allows to eliminate large chunks that may be classified as non road but containing a significant amount of points from the road surface. On the other hand, choosing a small chunk resolution may result in false chunk classification because the fitting score is not reliable (i.e not enough points for the fitting process). Figures 3.18a

and 3.19a show the segmentation results for $\phi_{res} = 2°$. Apparently, many misclassified points lie on vegetations as well as guardrail, whereas for $\phi_{res} = 9°$, these misclassification are corrected.



(a) $\phi_{res} = 2°$ (b) $\phi_{res} = 5°$ (c) $\phi_{res} = 9°$

Figure 3.18: Results of the proposed road surface detection in an obstacle-free scenario.

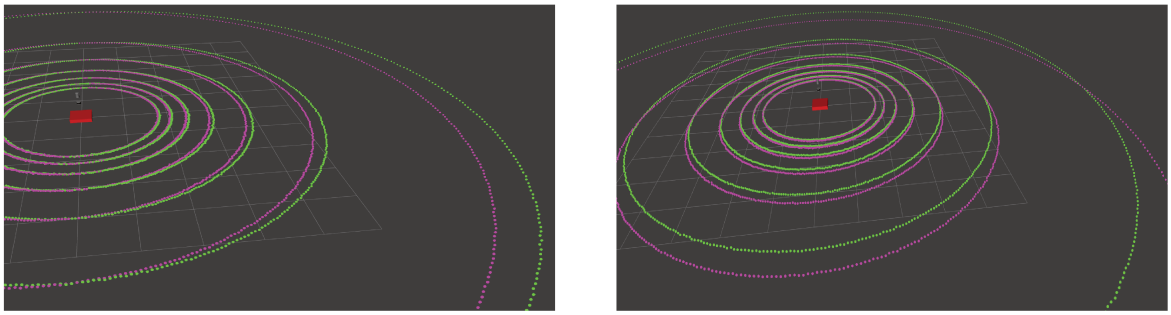


(a) $\phi_{res} = 2°$ (b) $\phi_{res} = 5°$ (c) $\phi_{res} = 9°$

Figure 3.19: Results of the proposed road surface detection in a moving truck scenario.

A second problem that affects the segmentation results is related to small pitch and roll variations. In practice, LiDARs mounted on moving vehicles may be subject to small bumps inducing therefore small variations in pitch and roll angles. We simulated the impact of small pitch and roll variations on LiDAR layers using Gazebo environment (see Figure 3.20). Visibly, this behavior has significant impact on distant layers (i.e small tilting angles) rather than layers that are close to the vehicle. This is expected since the impact of angle variations are seen at longer ranges. In Figures 3.18c and 3.19c, segmentation results with $\phi_{res} = 9°$ are acceptable only for the first four layers of the LiDAR. Starting from layer 5, the proposed method has poor performances. The same remark is valid for $\phi_{res} = 2°$ but starting from layer number 11.

Figure 3.19 illustrates the segmentation results in the presence of a moving truck. Our algorithm is capable to remove non road points related to moving objects that are filtered in most of state of the art approaches using elevation data. Nevertheless, one limitation of the approach is inherited from the iterative search process which stops for the first non road chunk. Indeed, our approach will stop the search process the first time it encounters the moving truck and will therefore omit the remaining part of the road. Fortunately, the impact of short-term occlusions are handled by the line tracking algorithm in lane markings detection module.



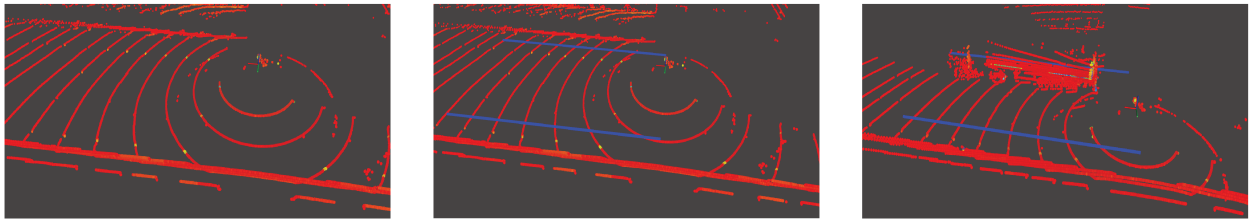
(a) Slight variation of pitch angle

(b) Slight variation of roll angle

Figure 3.20: LiDAR ring distortions caused by small variation of pitch and roll angles. Green and purple layers are the simulation results with and without introducing angle variations, respectively.

B Detection of lane markings

Our proposed approach has been tested for both LiDAR configurations: roof-mounted and bumper-mounted LiDARs. For a given object, the reflectivity of the LiDAR depends on its mounting height and the beam incidence angle. Road surface reflectivity data in Figure 3.21a is different from that in Figure 3.22a. Nevertheless, the lane marking detection process is not affected and the results are very similar (Figures 3.21b and 3.22b). This is one of the advantages that can be derived from the road surface extraction module. Indeed, it is possible to choose a very low reflectivity threshold since the road asphalt is not reflective at all.

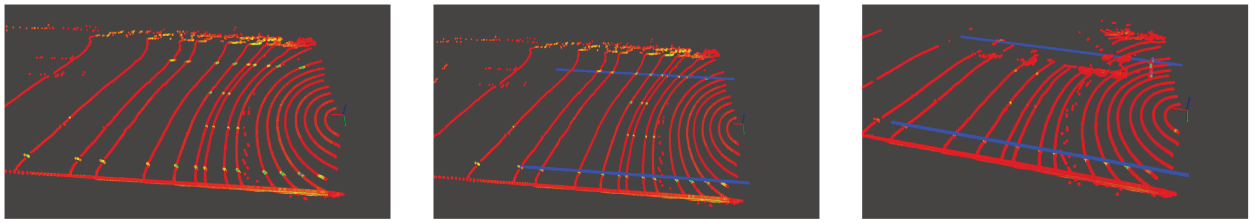


(a) Point cloud data

(b) Detected lines:obstacle-free scenario

(c) Detected lines:moving truck scenario

Figure 3.21: Detection of road markings using the roof-mounted LiDAR.



(a) Point cloud data

(b) Detected lines: obstacle-free scenario

(c) Detected lines:moving truck

Figure 3.22: Detection of road markings using the bumper-mounted LiDAR.

The illustrated results also show some missed road marking detections. The missed detections correspond in the majority of cases to dashed road markings. Indeed, because LiDAR data is sparse, only few pixels in the reflectivity grid map are obtained from dashed markings. Considering that we have fixed a minimum line length (i.e. minimum number of pixels corresponding to the line) in the Hough line extraction process, dashed markings are not detected. Decreasing the minimum length threshold would result in having false detections from small segments mainly due to the implementation of dilation operator. Our detection strategy is to allow having missed detections rather than having false detections. In other cases, missed detections are caused by temporary occlusion. Although, Kalman tracking is implemented to cope with this problem as in Figures 3.21c and 3.22c, the issue may persist if the line remains occluded for a long period of time (e.g. in traffic jams).

For convenience, since the reflectivity grid map is centered around the ego vehicle position, detection of road markings is implemented with signed values of the ranges r in the Hough parameter space. Negative r value means that the line is on the right side of the vehicle and positive r value means that the line is on the left side of the vehicle. Consequently, we can set the values of θ within the interval $[0, \frac{\pi}{2}]$ (i.e. default interval is $[0, \pi]$). Finally, for the rest of this manuscript, detected lines are given by the following set:

$$\{l_1 = (r_1, \theta_1),\ l_2 = (r_2, \theta_2),\ ..,\ l_n = (r_n, \theta_n)\} \tag{3.31}$$

C Detection of guardrail reflectors and traffic signs

Detection algorithms of traffic signs and guardrail reflectors have been tested in CTA2 test track and A13 highway road. Figures 3.23 and 3.25 show the detection results of traffic signs whereas Figures 3.24 and 3.26 show the detection results of guardrail reflectors. Detection of traffic signs is particularly sensitive to high reflective plates of moving objects (Figure 3.23). Indeed, the geometric criteria used to select valid traffic sign candidates are also satisfied in the case of license plates (i.e planar surface shape and the orientation of the normal vector). This is one of the limitations of using LiDARs which do not provide more information than the reflectance as provided by camera images. To cope with this problem, we used traffic signs stored in the map to further filter false detections. The idea is to check if a map traffic sign is located nearly at the same location as the detected traffic signs. Obviously, because detections are expressed in local LiDAR frames and map attributes are expressed in the global frame (WGS 84), we need to use an estimate of the vehicle position to transform local detections to the global frame. In this case, the accuracy of the used estimate is of major concern.

Figures 3.24 and 3.26 depict detection results for guardrail reflectors where NCC results are shown as colored bounding boxes. In contrast to traffic signs detection, it is less likely to find reflective planar surfaces that are perpendicular to the driving direction, thus, the NCC matching is robust to vehicle license plates. Also, NCC matching

is robust to the change of the environment. Indeed, although the used reflector template for NCC is a sample from CTA2 test track, it has proven to be robust on the A13 highway road.

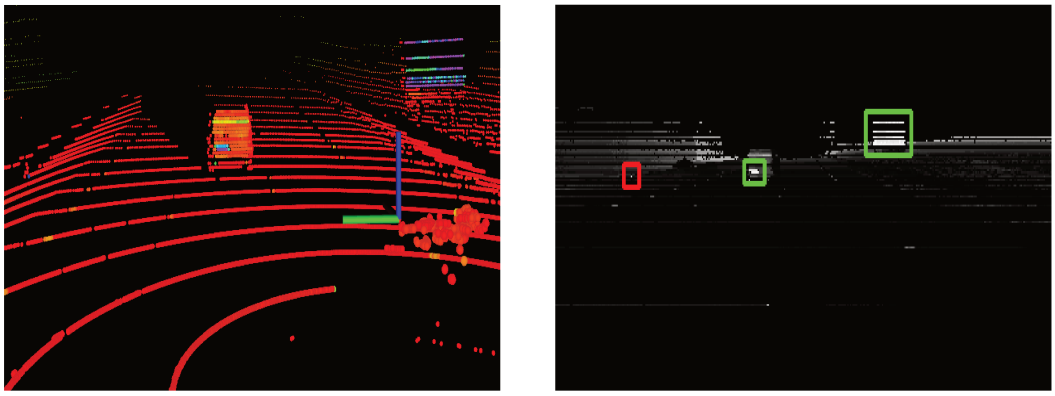


Figure 3.23: Traffic signs detection using the roof-mounted LiDAR on CTA2 test track (presence of a moving vehicle). (Left) LiDAR point cloud data. (Right) detection results on front reflectivity grid map, red boxes:discarded detection, green boxes:retained detection.

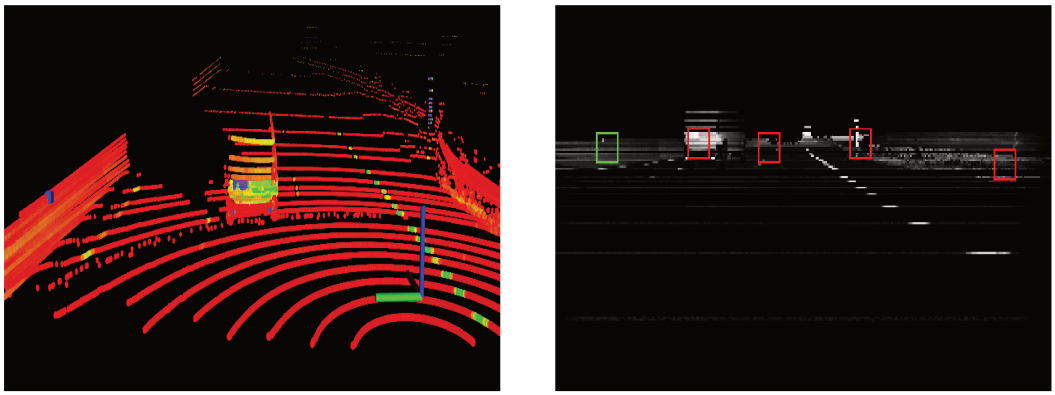


Figure 3.24: Guardrail reflector detection using the bumper-mounted LiDAR on CTA2 test track (presence of a moving vehicle). (Left) LiDAR point cloud data. (Right) detection results on front reflectivity grid map, red boxes:discarded detection, green boxes:retained detection.

In the rest of this manuscript, detected traffic signs and guardrail reflectors are given by the following set:

$$\mathcal{P}_k = \{l_1^k, l_2^k, .., l_m^k, s_1^k, s_2^k, .., s_n^k, r_1^k, r_2^k, .., r_p^k\} \tag{3.32}$$

with l for lane markings, s for traffic signs and r for guardrail reflectors.

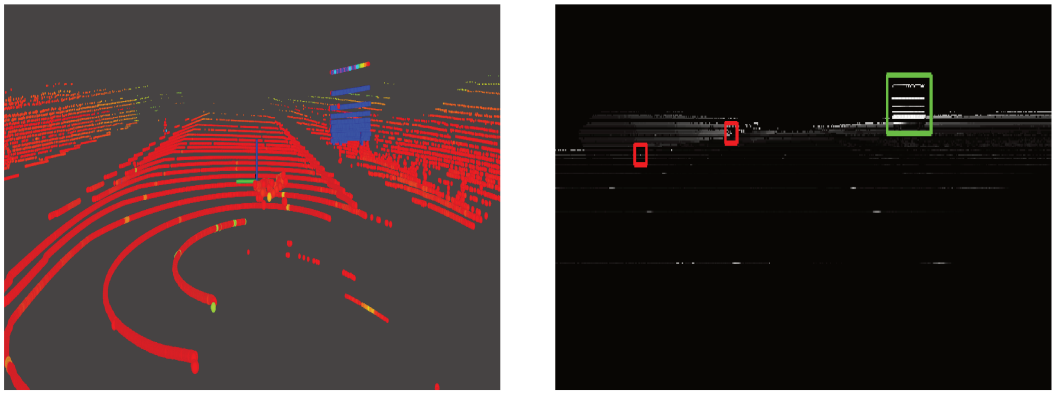


Figure 3.25: Traffic signs detection using the roof-mounted LiDAR on A13 highway road. (Left) LiDAR point cloud data. (Right) detection results on front reflectivity grid map, red boxes:discarded detection, green boxes:retained detection.

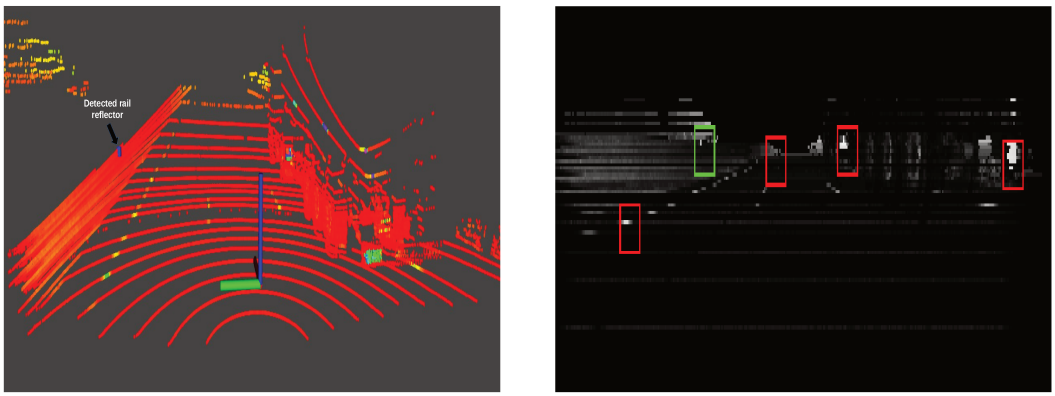


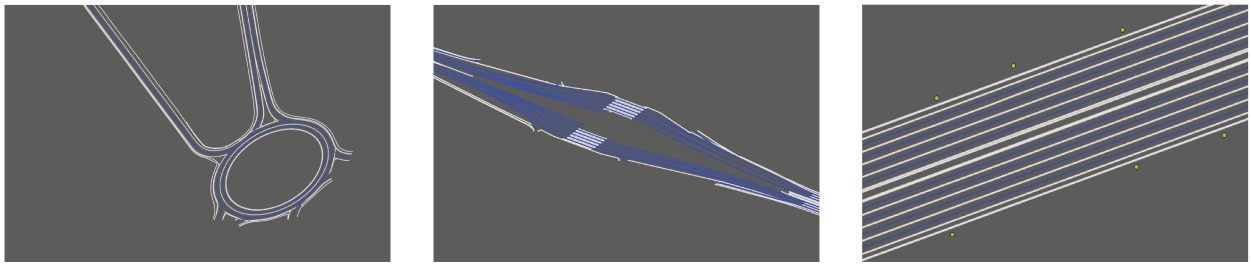
Figure 3.26: Guardrail reflector detection using the bumper-mounted LiDAR on A13 highway road (presence of a moving vehicle). (Left) LiDAR point cloud data. (Right) detection results on front reflectivity grid map, red boxes:discarded detection, green boxes:retained detection.

3.3 Map Management System (MMS)

In this section, we present the Map Management System (MMS) module that is a set of tools designed for extracting and managing map attributes. We start by a short description of the map followed by a discussion on the set of functions developed for the MMS module.

3.3.1 Description of the prototype map

In this study, we used a third party map that was originally designed for Renault's Urban Mobility Advanced Platform (UMAP) project. It was built with an absolute accuracy of 5cm stored as an SQLite database and has physical layers (tables). We present here the layers that we used in our study.



(a) Roundabout **(b)** Toll area on Highway **(c)** Road signals

Figure 3.27: Different map layers. Blue lines are the center lanes and white lines are the lane markings. Green dots represent the traffic signs (or guard rail) reflector map layers.

A Links

Links are described as polylines and are displayed as the centerlines of the drivable lane (Figure 3.27). They contain the geometric model of the road where vehicles are allowed to drive. The geometric description is modeled by using polylines defined with a specific type called *linestring* and that is composed of a set of consecutive points expressed in the WGS 84 coordinate system. A link also has other attributes: the link type (e.g. intersection, roundabout, parkings, etc.), the speed limit and the start and end points of the link. Also, a unique ID is attributed to each link to facilitate their retrieval, insertion and identification from the map table. Moreover, because a lane is often limited by two road markings, a link has two different IDs indicating the left and right lane markings. Finally, lane markings are stored in an independent table called: linkBorder.

B Link Borders

Similarly to the link representation, lane markings are represented by polylines of type *linestring* (Figure 3.27). Link borders are identified by unique IDs and store the lane marking type (e.g. continuous, dashed, pavement, etc), width and distance to the centerline (Link).

C ShapePoints

ShapePoints are the most basic elements in the used map. They consist of 3D points expressed in WGS 84 coordinate system and are attached to other high level objects such as Links.

D Road signals

Road signals can be of different types: traffic light, traffic signs, stop and yield signs, etc. A road signal is a 3D point expressed in the local Cartesian frame (Figure 3.27). It is characterized by a unique ID and other information such as: the attributed Link ID and the type ID.

3.3.2 MMS module functions

The developed tools in the MMS module are described below:

A Map interface functions

The map interface functions establish the communication with the map database and allow to extract map elements within a region of interest. A region of interest is a circular region centered around a rough position of the vehicle (e.g GNSS position) and whose radius is a fixed parameter.

B Extraction of road signals

The road signals table contains 3D points expressed in the WGS 84 coordinate system. Thus, we used the haversine formula to calculate the distance between the GNSS position and the road signal points which are expressed as longitude/latitude values.

C Extraction of Link Borders

At first, we extracted all the links in the region of interest using their relative shapepoints. Second, we filtered out all the links in the opposite side to the driving direction using the vehicle heading measure from GNSS receiver. Finally, for each link, we extracted the left and right link borders (i.e. left and right lane markings). Instead of executing the latter steps each time a GNSS position is available, we constructed a link

border tree where we exploited the connectedness property of linestring type. A link border is directly connected to another link border if the end point of the first is equal to the start point of the second. Consequently, the extraction steps are only executed for the initialization of the tree.

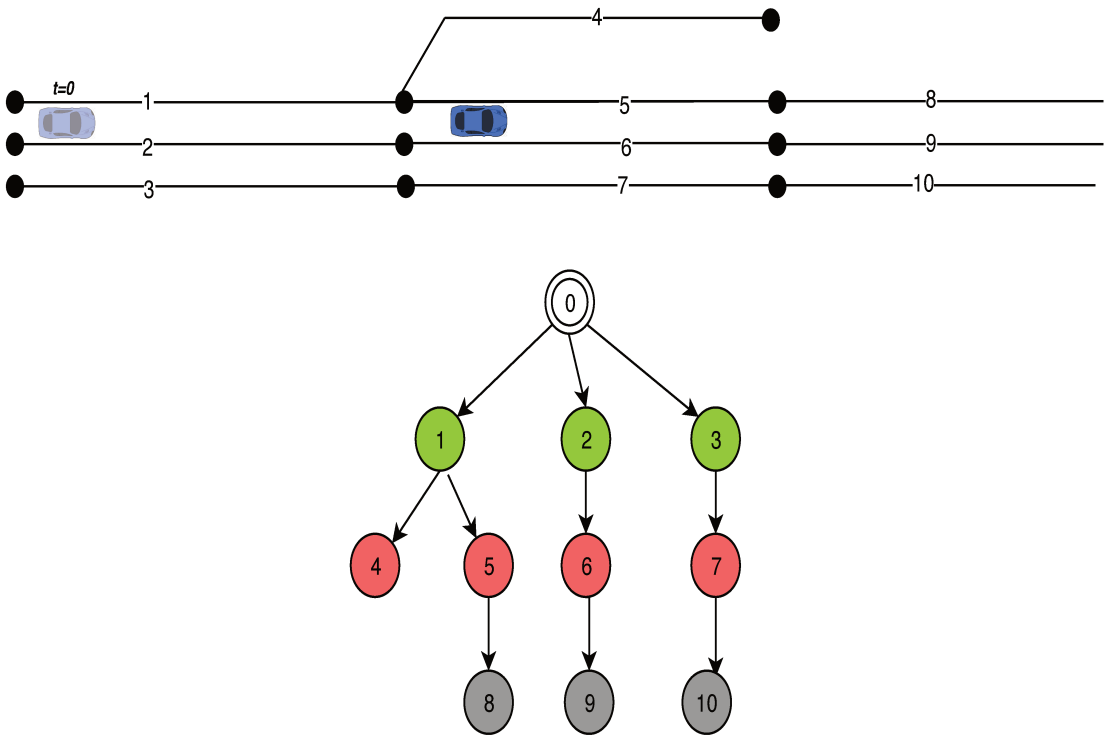


Figure 3.28: Link border tree. At time $t = 0$ the initialized link borders are (1, 2 and 3) and are shown as green nodes in the tree. As long as the vehicle is moving, the next nodes (red nodes) can be expected without querying in the map. This allows rapid navigation between map link borders.

Figure 3.28 shows an example of a set of link borders and the corresponding tree. At $t = 0$, the initialization of the tree yields three different nodes "1", "2" and "3". After a certain amount of time, the vehicle should switch to nodes "4", "5", "6" and "7", etc. Let $p_s = (px_s, py_s)$ and $p_e = (px_e, px_s)$ be the starting and ending points of link borders X. To switch from link border X to link Border Y, the following condition should be met:

$$(\overrightarrow{p_v px_s}.\overrightarrow{u}) \times (\overrightarrow{p_v px_e}.\overrightarrow{u}) > 0 \; and \; (\overrightarrow{p_v py_s}.\overrightarrow{u}) \times (\overrightarrow{p_v py_e}.\overrightarrow{u}) < 0 \tag{3.33}$$

Where p is the vehicle GNSS position, $u = (cos(heading), sin(heading))$ and $heading$ is the vehicle heading.

D Map Tracker Point

A second important function in the MMS module is the calculation of a map tracker point. The usefulness of this concept will be revealed in the localization part. A map tracker point is the projection point q of a given point (the vehicle position for example) p_v to a link border. To do that, we iterate over the contiguous segments of the linestring set of points := $\{p_1, p_2, .., p_N\}$. For each pair of points (p_i, p_{i+1}), we calculate the projection point of the vector $\overrightarrow{p_i p_v}$ into the line defined by (p_i, p_{i+1}), say it is q_i. The retained point q_i is the point that is inside the segment $[p_i, p_{i+1}]$. Let the vectors $\overrightarrow{u}$ and $\overrightarrow{v}$ be defined as follow: $\overrightarrow{u} = \overrightarrow{p_i p_v}$ and $\overrightarrow{v} = \overrightarrow{p_i p_{i+1}}$. The normalized vectors are computed as:

$$\vec{u} = \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}, \ \vec{v} = \frac{\overrightarrow{v}}{\| \overrightarrow{v} \|} \tag{3.34}$$

The projection of the point p_v is inside the line segment $[p_i, p_{i+1}]$ if the following condition is met:

$$0 \leq \vec{u} \,.\, \vec{v} \leq \frac{\| \overrightarrow{u} \|}{\| \overrightarrow{v} \|} \tag{3.35}$$

The projection point q_i can be calculated by the following:

$$\overrightarrow{p_i q_i} = \frac{(\overrightarrow{u} \,.\, \overrightarrow{v})}{\| \overrightarrow{u} \|} \times \overrightarrow{u} \tag{3.36}$$

In the rest of this manuscript, we express the set of extracted road signals (traffic signs and guardrail reflectors) and link borders are given by:

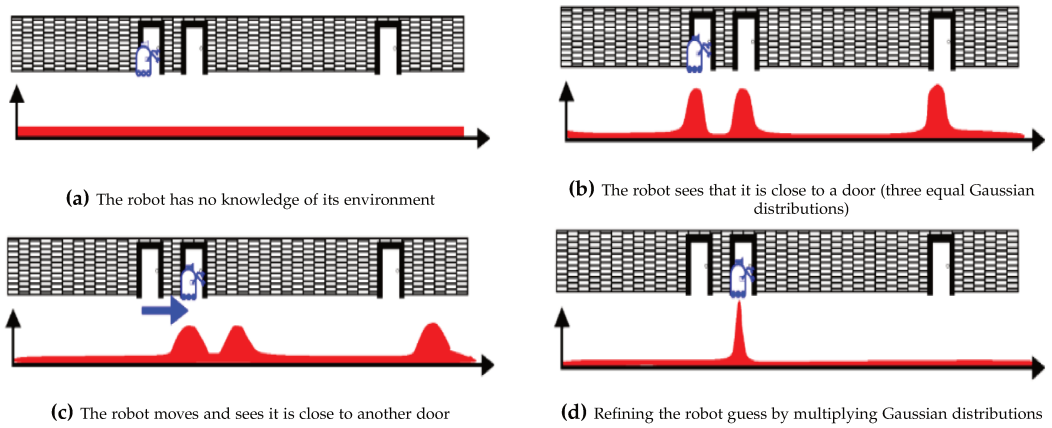
$$\mathcal{M}_k = \{ls_1^k,\ ls_2^k, ..,\ ls_n^k,\ rs_1^k,\ rs_2^k, ..,\ rs_m^k\} \tag{3.37}$$

where ls_i stands for the extracted link border linestrings (polyline) and rs_i the extracted road signals.

3.4 Markov Localization

3.4.1 Basic concept and mathematical formulation

In general, Markov localization addresses the problem of estimating the state of a mobile robot from sensor data. The robot is assumed to navigate in a static environment and that the only variable affected by sensor data is the robot state. This is commonly known as the *Markov assumption*. Instead of handling a single state, Markov localization maintains a probability distribution over the space of all hypotheses. This representation allows to weight these hypotheses according to the observed sensor data.



(a) The robot has no knowledge of its environment

(b) The robot sees that it is close to a door (three equal Gaussian distributions)

(c) The robot moves and sees it is close to another door

(d) Refining the robot guess by multiplying Gaussian distributions

Figure 3.29: Basic idea of Markov localization [Fox et al., 1999]

Markov localization is very well presented and formulated in [Fox et al., 1999]. At the beginning, we assume a one-dimensional robot location that can move horizontally (cf. Figure 3.29). Before having any sensor data, the robot can be anywhere in the environment and the state probability distribution is uniformly distributed over the environment (Figure 3.29a). Then, the robot reads sensor data and gets to know that it is close to a door, consequently, the probability distribution is equally risen next to each door location (Figure 3.29b). After a one meter translation of the robot, the probability distribution is also shifted by one meter (Figure 3.29c). Getting new sensor data that indicate that the robot is always close to a door, the result of the new probability distribution can be calculated as the product of the probabilities in Figures 3.29b and 3.29c. The resulting probability is therefore centered around the true location of the robot (Figure 3.29d).

We consider the robot location as a three-dimensional variable: $\mathbf{x}_k = (x_k, y_k, \theta_k)$, where (x_k, y_k) are the two-dimensional coordinates in a local Cartesian frame and θ denotes its heading. Let $\mathbf{z}_{0:k}$ be the set of observation data from time 0 to time k. For localization, we are interested in constructing the posterior density of the current state given by : $p(\mathbf{x}_k \mid \mathbf{z}_{0:k})$. This probability density function is recursively computed at each time step in two different phases: prediction and update.

A The prediction step

In the prediction step, a motion model is used to update the probability density $p(\mathbf{x}_k \mid \mathbf{z}_{0:k})$ only taking into account the motion data. It only depends on the previous state estimate $\mathbf{x}_{k-1}$ (Markov) and on a control input $\mathbf{u}_{k-1}$. The control input is most often the inertial data of the robot. The motion update is given by the following:

$$p(\mathbf{x}_k \mid \mathbf{z}_{0:k}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{z}_{0:k-1}) d\mathbf{x}_{k-1} \tag{3.38}$$

where the motion model is described by the conditional probability density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1})$

B The update step

In the update step, the observation data is incorporated from the sensors to obtain the posterior probability density. The measurement $\mathbf{z}_k$ at time k is supposed to be conditionally independent of earlier measurements $\mathbf{z}_{0:k-1}$. The measurement model $p(\mathbf{z}_k \mid \mathbf{x}_k)$ expresses the probability of seeing the measurement $\mathbf{z}_k$ when being at position $\mathbf{x}_k$. The posterior density over $\mathbf{x}_k$ is obtained by the Bayes theorem:

$$p(\mathbf{x}_k \mid \mathbf{z}_{0:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1})}{p(\mathbf{z}_k \mid \mathbf{z}_{0:k-1})} \tag{3.39}$$

The state estimation is processed by recursively computing the prediction and the update steps. At time $t = 0$, the initial state $\mathbf{x}_0$ is modeled in the form of a density $p(\mathbf{x}_0)$ that could be a uniform density over the allowable positions. Moreover, a prior map M is integrated as follows: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k}, M)$.

To summarize, Markov localization recursively computes two different phases: prediction and update. The prediction phase is based on the definition of a motion model defined by the probability density: $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ and the update phase requires the definition of a measurement model defined by the probability density: $p(\mathbf{z}_k \mid \mathbf{x}_k)$. The following sections describe different approaches of Markov-based localization. We start with a specific case, the Kalman filter, where motion and measurement can both be modeled by normal distributions. Then, we describe other approaches for nonparametric estimation: grid-based and sampling-based localizations.

3.4.2 The Kalman Filter (KF)

The Kalman Filter (KF) is a special case of Markov localization where motion and measurement can be modeled by Gaussian density and the initial state is also specified as a Gaussian. Under these assumptions, the probability density $p(\mathbf{x}_k \mid \mathbf{z}_{0:k})$ remains Gaussian in all the steps of the filter. One of the key advantages of Kalman Filter is the ability to evaluate Equations 3.38 and 3.39 in closed form solutions. In addition, the posterior density can be described in a very concise form (the mean and covariance matrices of the Gaussian). The implementation of KF is computationally quite efficient, for a matrix size of $d \times d$, the complexity of matrix inversion is approximately $O(d^{2.4})$. Although many extensions of the Kalman filter are available (Extended Kalman Filter, Unscented Kalman Filter), the Gaussian assumptions inherited by the KF are not always appropriate in case of non-linear and non-Gaussian motion and measurement models. The Kalman filter is classified among the family of parametric filters. In the following section, we discuss two variants of nonparametric filters: *Grid Localization* and *Monte Carlo Localization*. Both algorithms differ from to the Kalman Filter in their ability to handle raw sensor data and solve the global localization problem.

3.4.3 Grid-based Localization

Grid localization is a very popular technique for global localization. It uses a grid decomposition of the pose space. The posterior density is a collection of discrete probability values, $\{p_{i,k}\}$ where i stands for the cell id and $k$for the time step, and each probability value is defined over a grid cell. It requires as input the discrete probability values $\{p_{i,k-1}\}$, the new measurement z_k, the control and the map and outputs the updated probability values $\{p_{i,k}\}$. A pseudo code for grid localization is given in Algorithm. 2.

Data: $\{p_{i,k-1}\}$, z_k, u_k and M
Result: Updated probability values: $\{p_{i,k}\}$
1 **for** *i iterating over all grid cells* **do**
2 Prediction step:

$$\tilde{p_{i,k}} = \sum_j p_{j,k-1} \times \overbrace{f(x_{i,k}\ ,\ x_{j,k-1},\ u_k)}^{motion\_model}$$

Update step:

$$p_{i,k} = \alpha \times \tilde{p_{i,k}} \times \overbrace{g(z_k\ ,\ x_{i,k}, M)}^{measurement\_model}$$

3 **end**

Algorithm 2: Grid localization pseudo code

A key parameter for grid localization is the resolution of the grid. A finer grained representation, also known as *metric* representation, allows to have better localization accuracy, but at the expense of computational time. In, a coarse resolution, known as *topological* representation, reduces the computational time but results in a loss of the localization accuracy. That said, the motion and measurement models can also be highly affected by the grid resolution. For example, adopting a topological representation (e.g. a grid cell resolution of 1 m) with a high accurate measurement model will cause a drastic variation within each cell. Similarly for a motion model with a robot speed of $0.1m/s$, the robot will not be able to change cell for many consecutive frames.

3.4.4 Monte Carlo Localization (MCL)

The second nonparametric filter is the very popular Monte Carlo Localization based on Particle Filter. This powerful technique is adapted for many localization problems: local and global. The basic idea is to model the posterior density by a set $\mathcal{X}$ of weighted particles or samples as follows:

$$\mathcal{X}_k = \{(x_k^{[1]}, w_k^{[1]}), (x_k^{[2]}, w_k^{[2]}), ..., (x_k^{[N]}, w_k^{[N]})\} \tag{3.40}$$

The variable $x_k^{[i]}$ is a pose hypothesis and the variable $w_k^{[i]}$ is the corresponding importance weight. One of the key advantages of the particle representation is the ability to model nonlinear transformations of random variables. The variable N is the number of particles and is a parameter of the algorithm. This parameter is very important as it directly affects the computational time and the accuracy of the filter convergence. Increasing N allows accurate convergence but also increases the run time of the filter.

Data: $\mathcal{X}_{k-1}$, z_k, u_k and M
Result: $\mathcal{X}_k$
1 $\bar{\mathcal{X}}_k = \mathcal{X}_k = \emptyset$;
2 **for** *i = 1 to N* **do**
3 | sample $x_k^{[i]} \sim p(x_k \mid u_k, x_{k-1}^{[i]})$;
4 | $w_k^{[i]} = p(z_k \mid x_k^{[i]}, M)$;
5 | $\bar{\mathcal{X}}_k = \bar{\mathcal{X}}_k \cup (x_k^{[i]}, w_k^{[i]})$;
6 **end**
7 **for** *i = 1 to N* **do**
8 | draw i with probability $\sim w_k^{[i]}$;
9 | add $x_k^{[i]}$ to $\mathcal{X}_k$;
10 **end**

Algorithm 3: The particle filter algorithm [Thrun, 2002a]

Algorithm 3 is a pseudo code to describe the basic Monte Carlo algorithm based on particle filter. Line 3 generates a hypothesis state based on the previous particle state and the control u_k by means of the state transition distribution:

$$p(x_k \mid u_k, x_{k-1})$$

Line 5 computes the importance factor $w_k^{[i]}$ by incorporating the recent measurement z_k and the map M in the measurement model:

$$p(z_k \mid x_k, M)$$

The tricky step of the Particle Filter goes from lines 7 to 10. It is called the *resampling* or *importance resampling* and consists in drawing new particles with replacement N particles from the temporary set $\bar{\mathcal{X}}_k$. For the initialization step, all the particles have the same weight and are uniformly distributed over the pose space. The probability of drawing a particle is given by its importance weight. Importance resampling generates a new set of particles by replacing bad particles (i.e. with low weights) with good particles (i.e. with higher weights). Thus, the new set is likely to have many duplicates.

3.4.5 Discussion

Kalman and particle filters are both two different variants of Bayes filters that recursively estimate the state vector of a given process (e.g. position estimation in robot localization). Kalman filter has become the most popular and most used tool for state estimation. Its strength lies in its simplicity and its computation efficiency for small-size state vectors. This simplicity is due to the fact that it represents the state vector by a multivariate Gaussian distribution. In addition, it approximates the state (prediction step) and measurement (update step) transitions by linear equations. However, two major limitations to Kalman filtering approaches are often encountered: first, the linearization in EKF is approximated by linear Taylor expansions. Given that in most robotics problems, the state and measurement transitions are nonlinear, linearization may not be sufficient in some points of the space. A second limitation is observed in SLAM-approaches or map-based approaches where the state vector is not only limited to the position of the robot but also includes landmark positions. Since the complexity of Kalman filters is $O(K^2)$ where K is the size of the state vector, adding landmarks may increase the complexity and reduces the time efficiency.

Particle filter has also been used for state estimation. It approximates a distribution by a set of particles drawn from this distribution. Unlike Kalman filter, it can represent a broader space of distributions than Gaussians. Another advantage of particle filter is its ability to model nonlinear transformations of random variables. In addition, for landmark-based localization approaches, its time efficiency can be controlled by the number of particles. Indeed, for M particles and K landmark, the computational time is $O(MK)$ (e.g. $O(K^2)$ for Kalman filter). In our approach, we adopted a particle filter implementation in virtue of their advantages over Kalman filter for map-based localization. The proposed implementation is illustrated hereafter.

3.5 Proposed Particle Filter implementation

The proposed particle filter implementation follows the same basic steps as a Markov localization. The filter inputs are: extracted map elements M_k, inertial data, velocity v and yaw rate $\dot{\Omega}$, GNSS position and LiDAR perception features. Let $\mathcal{X}_k$ be the set of samples at time k:

$$\mathcal{X}_k := \langle \mathbf{x}_k^{[1]}, w_k^{[1]} \rangle, ..., \langle \mathbf{x}_k^{[N]}, w_k^{[N]} \rangle \tag{3.41}$$

The state space vector $\mathbf{x}$ consists of the 2D position (x, y) and the heading γ:

$$\mathbf{x} = [x, y, \gamma] \tag{3.42}$$

For the initialization step, the samples are uniformly drawn within the ellipsoid error of the GNSS position. The integration of a rough position estimation (such as a

GNSS position) is very useful to reduce the computational cost of the initialization, especially in large-scale outdoor environments. In the following subsections, we describe the proposed implementations of particle filter (PF) fundamental steps (prediction, update and resampling).

3.5.1 The prediction step

The motion model adopted for our approach is a constant turn rate and velocity (CTRV) model, as it can be applied in highway driving conditions. The following system describes the motion update equations:

$$\gamma_k^{[i]} = \gamma_{k-1}^{[i]} + \dot{\Omega}_k \times dt + \nu_\gamma \tag{3.43}$$

$$x_k^{[i]} = x_{k-1}^{[i]} + v_k \times dt \times cos(\gamma_k^{[i]}) + \nu_x \tag{3.44}$$

$$y_k^{[i]} = y_{k-1}^{[i]} + v_k \times dt \times sin(\gamma_k^{[i]}) + \nu_y \tag{3.45}$$

Where v_k and $\dot{\Omega}_k$ are respectively the vehicle velocity and yaw rate. The motion model error distributions are zero mean normal distributions and are modeled by ν_γ, ν_x and ν_y.

3.5.2 Different weighting strategies

In order to update the particle weights, we used the perception data $\mathcal{P}_k$ and the extracted map elements M_k and we propose four weighting strategies described in the following paragraphs:

A Weighting from lane markings

Let $\{\overbrace{l_1^k,\ l_2^k,\ ...,\ l_{m_1}^k}^{Perception},\ \overbrace{ls_1^k,\ ls_2^k,\ ...,\ ls_{m_2}^k}^{Map}\}$ respectively be detected lane markings and extracted linestring at time k. The polar coordinates of each linestring ls_i^k are calculated by using the relative map tracker point. For a tracker point $q_k^{[i]} = (x_{q_i^{[i]}}, y_{q_i^{[i]}})$, the polar coordinates are given by:

$$r_k^{[i]} = \|\mathbf{x}_k^{[i]} - q_k^{[i]}\| \tag{3.46}$$

$$\theta_k^{[i]} = atan2(y_{q_i^{[i]}} - y_k^{[i]}, x_{q_i^{[i]}} - x_k^{[i]}) - \gamma_k^{[i]} \tag{3.47}$$

By applying the above equations to all linestrings, the result is given by the following set:

$$\{ls_1^{k,i} = (\bar{r}_1^{k,i}, \bar{\theta}_1^{k,i}), ls_2^{k,i} = (\bar{r}_2^{k,i}, \bar{\theta}_2^{k,i}), .., ls_{m_2}^{k,i} = (\bar{r}_n^{k,i}, \bar{\theta}_{m_2}^{k,i})\} \tag{3.48}$$

The calculated polar coordinates are also expressed with signed r values, r is negative if the linestring is to the right of the particle position and positive otherwise. The relative sub-weight w_l is calculated as below:

$$\begin{cases} f(l_q^k\ ,\ ls_j^{k,i}) & = e^{\frac{-(\tilde{r}_j^{k,i}-r_q^k)^2}{2\sigma_r^2}} + e^{\frac{-(\theta_j^{k,i}-\theta_q^k)^2}{2\sigma_\theta^2}} \\ w_{l,k}^{[i]} & = \sum_{q=1}^{m_1}\ \sum_{j=1}^{m_2} f(l_q^k\ ,\ ls_j^{k,i}) \end{cases} \tag{3.49}$$

Where σ_r and σ_θ are two measurement noise variables. The maximum value of the function f is reached when $l_q^k = ls_j^{k,i}$ which means that the value of w_l is maximum when the polar coordinates of the projected linestrings are very similar to the polar coordinates of the detected lane markings. As a result, the maximum w_l value is obtained for the particles that are laterally positioned as the true position.

A typical problem can occur with missing lane marking detections. For example, when two lane markings are detected (the lines of the ego lane) on a highway road with three lanes (four lane markings), the calculated value of w_l has three equal peaks (one in each lane), In this case, the particle filter output will give three different equally distributed particles in all the lanes (cf. Figure 3.30). To remove this ambiguity, we use the median barrier as explained in the next paragraph.

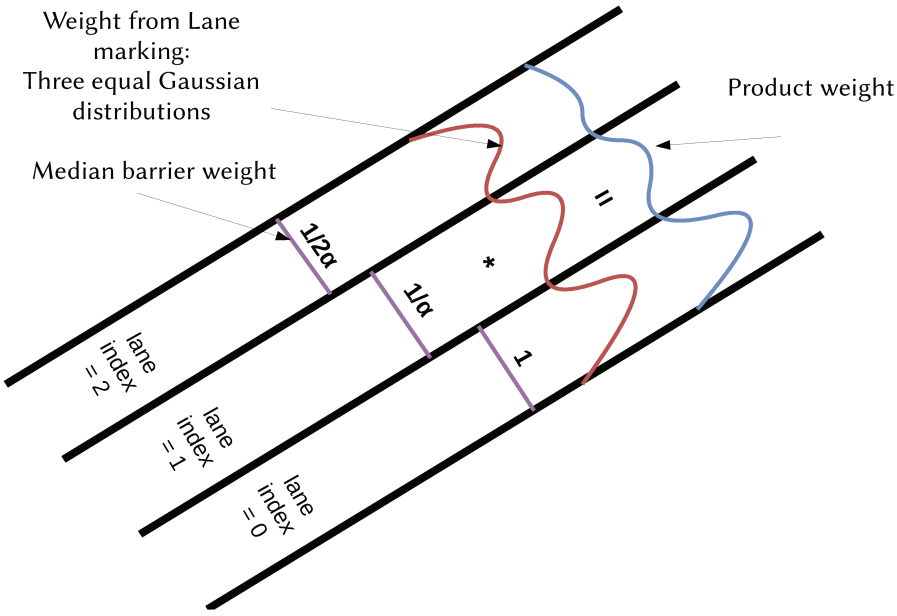


Figure 3.30: Multiple hypothesis problem. When only the lines of the ego lane are detected, the computed weight from lane markings gives three equal Gaussian distributions and is therefore not sufficient to decide which lane the ego vehicle belongs to. Using the median barrier allows to mitigate Gaussian distributions related to false lane assignments and keeps the Gaussian distribution related to the right lane assignment.

B Weighting from median barrier

The main use of the median barrier is to calculate the number of the lane where the ego vehicle is. Consider that the calculated lane number from the detected barrier is $L_{index} = \frac{r}{w}$, where the value w is directly extracted from the map. For each particle we use the map polylines (map lane markings) to also attribute a lane index for each particle $L^{i,m}_{index}$. The calculated weight from median barrier is therefore given by:

$$w_{barrier} = \frac{1}{1 + \alpha \times |L_{index} - L^{i,m}_{index}|} \tag{3.50}$$

Where α is a penalty parameter ($\alpha = 100$ in our case). The interpretation of this equation is straightforward, all the particles that are within the same lane as the true lane of the ego vehicle share the same lane index $L^{i,m}_{index}$ which is also equal to L_{index}. In this case, the value of $|L_{index} - L^{i,m}_{index}| = 0$ and the calculated weight from the barrier is equal to one. Assume that the ego vehicle is on the lane indexed (lane_index = 0) (cf. Figure 3.30). Particles on lane number 0 (i.e. the true ego vehicle lane) will have $w_{barrier} = 1$. Particles on lanes number 1 and 2 (lane_index = 1 and 2) will respectively have $w_{barrier} = \frac{1}{\alpha}, \frac{1}{2\alpha}$. Multiplying w_l and $w_{barrier}$ for the three lanes gives the following values $(w_l, \frac{w_l}{\alpha}, \frac{w_l}{2\alpha})$.

C Integration of GNSS data

Although the positions estimates from commercial GNSS receivers are not accurate enough for global localization, they can be exploited to define a bounded error of the localization system.

Very often, the distances from particle positions to the GNSS position are calculated and compared to threshold distance ρ_{max} to decide whether to keep the particle or not. This proves to be very useful when the particles tend to get far from the GNSS position [Chausse et al., 2005]. The proposed method is similar in the idea but different in the formulation of the adopted metric. The use of the $\mathcal{L}_2$-norm is very common but propagates the GNSS position error in both $x - y$ directions and strongly affects the lateral positioning of the vehicle. Alternatively, we computed the average of the relative distances D_i between a particle calculated MTPs and the GNSS calculated MTPs and compared it to a threshold distance ρ_{max}. Figure 3.31 shows an example of the calculation of D_i, a map composed of two lane markings, a GNSS position with its relative MTPs and three particle hypothesis with their relative MTPs. Although the three particles are not in the same position, their relative MTPs coincide. As a result, all the particles will have the same distance $D_i = \frac{s_1+s_2}{2}$, where s_1 and s_2 are the curvilinear lengths, thus the lateral position will not be affected by this process.

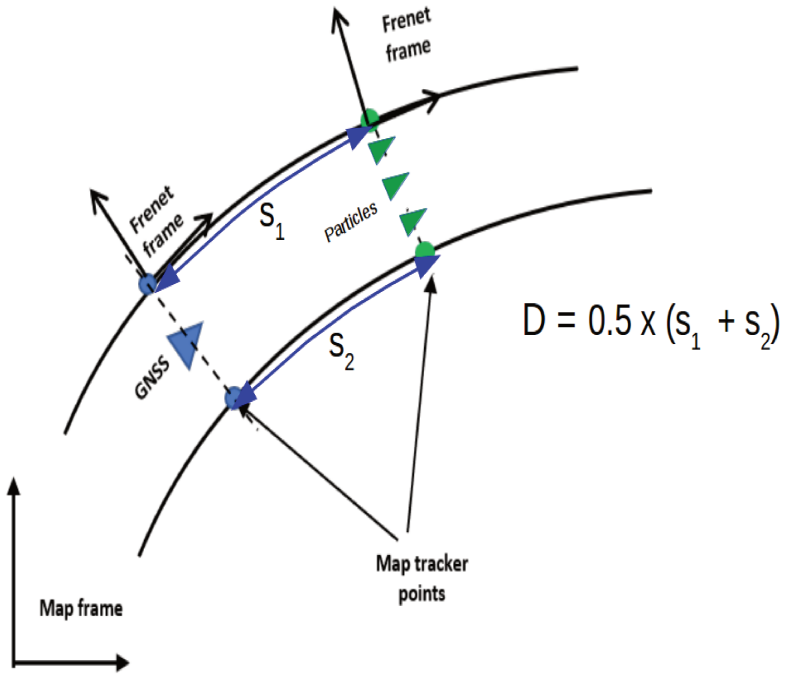


Figure 3.31: Illustration of GNSS update using map tracker points. Green triangles depict particle positions. Blue triangle is the GNSS position, circles are the corresponding map tracker points. D is calculated using the curvilinear lengths s_1 and s_2.

D Weighting from traffic signs and guardrail reflectors

Detected traffic signs and guardrail reflectors are mainly used to correct the error in the longitudinal direction. For each particle $\mathbf{x}^{[i]}$, we first transform the coordinates of map road signals from the map frame to the particle local frame. Let the transformed set be:

$$\{rs_1^{k,i} = (\bar{x}_1^{k,i}, \bar{y}_1^{k,i}), rs_2^{k,i} = (\bar{x}_2^{k,i}, \bar{y}_2^{k,i}), .., rs_{m_2}^{k,i} = (\bar{x}_{n_2}^{k,i}, \bar{y}_{n_2}^{k,i})\} \tag{3.51}$$

In addition, detected road signals (traffic signs and guardrail reflectors) are given by:

$$\{s_1^k,\ s_2^k,\ ...,\ s_{n_1}^k\} \tag{3.52}$$

To calculate the weight we start by searching for associations between the detected road signals and the transformed set of map road signals. Given the couple $(rs_m^{k,i}, s_n^k)$, the weighting function is given by the following expression:

$$g(s_n^k\ ,\ rs_m^{k,i}) = e^{\frac{-(\|s_n^k\| - \|rs_m^{k,i}\|)^2}{2\sigma_d^2}} \tag{3.53}$$

Where σ_d is a detection measurement error variance. Finally, the calculated weights for guardrail reflectors and traffic signs are:

$$\begin{cases} w_r & = \sum_{q=1}^{p_1} g(s_q^k\ ,\ rs_q^{k,i}),\ if\ s_q^k \text{ and } rs_q^{k,i} \text{ are guardrail reflector} \\ w_s & = \sum_{q=1}^{p_2} g(s_q^k\ ,\ rs_q^{k,i}),\ if\ s_q^k \text{ and } rs_q^{k,i} \text{ are traffic signs} \end{cases} \tag{3.54}$$

E Discussion

Lane markings are crucial for lateral positioning but do not have a significant contribution in the longitudinal direction. Road curvatures can be utilized to reduce the longitudinal drift [Suganuma and Uozumi, 2011a] but require high curvature values to be efficient, which is not the case of highway roads. In contrast, road traffic signs and guardrail reflector contribute in the longitudinal direction. Traffic signs may be absent for hundreds of meters on the highway, whereas guardrail reflector are repetitive but not guaranteed to be installed in all highway roads. The worst case is the simultaneous absence of traffic signs and guardrail reflector. In this case, the particle filter is only relying on lane markings and GNSS data.

$$w_k^{[i]} = \begin{cases} w_{k-1}^{[i]} \times (w_{k,l}^{[i]} \times w_{k,s}^{[i]} \times w_{k,r}^{[i]} \times w_{k,barrier}^{[i]}),\ if\ D_i \leq \rho_{max}. \\ 0,\ else. \end{cases} \tag{3.55}$$

Weights should be normalized to model the probability density. The normalization is given by:

$$\tilde{w}_k^{[n]} = \frac{w_k^{[n]}}{\sum_{i=1}^{N} w_k^{[i]}} \tag{3.56}$$

Finally, different options have been considered to calculate the filter estimate from the set of weighted particles. The standard option, which we use in our study, is to compute the weighted mean:

$$\bar{\mathbf{x}}_k = \sum_{i=1}^{N} \tilde{w}_k^{[i]} x_k^{[i]} \tag{3.57}$$

There is also another option which consists in taking the best particle, that is, the particle with the maximum weight:

$$\bar{\mathbf{x}}_k = x_k^{[p]},\ \text{where } p = \arg\max_i (w_k^{[i]}) \tag{3.58}$$

3.5.3 The constrained update step

One of the contributions of this work is the proposal of a modified version of the update step which increases the performance of the filter by optimizing the particles coverage of the map. This modification is designed to tackle a limitation of particle filtering which is called *particle deprivation*. This occurs when the number of particles is too small to cover the most likely regions which may result in the absence of particles in the vicinity of the correct position. In our case, this problem gets worse on some highway road structures when no guardrail reflector is available and in the temporary absence of traffic signs. Of course, this implicitly assumes the permanent availability of road markings. In the worst case, if the road markings are also not detected, the filter only relies on GNSS data. The latter situation is very unlikely on highways. Figure 3.32a illustrates the problem with a standard particle filter implementation and the result with the constrained update strategy.

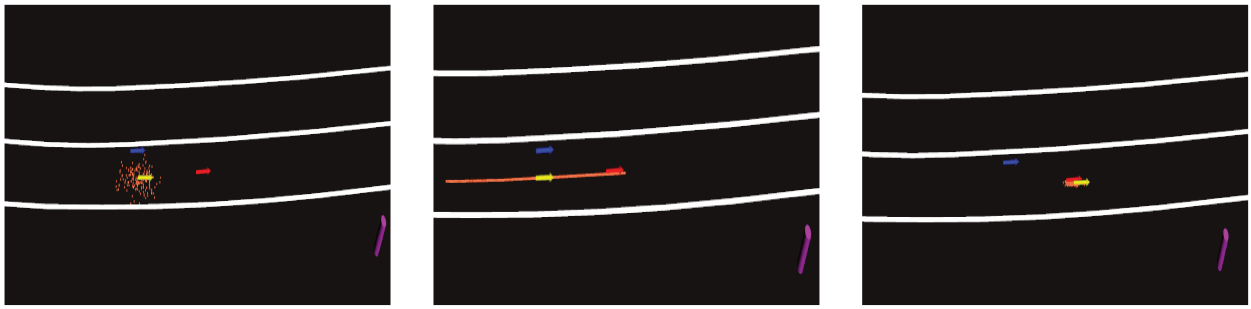


Figure 3.32: Constrained update strategy: (a) particle deprivation: the particles (orange) have drifted away from the ground truth position (red). The area around the true position is empty. (b) proposed regeneration of particles taking into account the shape of the map. (c) result of the constrained update strategy. The filter has converged in the vicinity of the ground truth position.

Implementing the weighting strategy from section D would result to low w_s values (w_s exponentially tends to zero) because of the discrepancy between the ground truth position and the particle positions. In this case, the calculated weights w_s have no contribution to the localization accuracy. To characterize this behavior, we calculate the maximum weight obtained from traffic signs as follows:

$$w_{s,max} = \arg\max_{i}(w_k^{[i]}) \tag{3.59}$$

If $w_{s,max}$ is below a threshold $w_{s,th}$ then the particle distribution on the map is not sufficient to ensure filter convergence. One possible way to solve this problem is to increase the number of particles by populating more regions in the map and therefore increasing the density of particles around the true position. Nevertheless, this reduces

runtime performance as the number of particles grows significantly. In addition, randomly generating particles is not efficient since we know that some regions of the map are unlikely to be populated by particles. We propose a more elegant way to generate particles by relying on the lateral position of the filter estimate from the previous filter estimate and from the geometry of the road inferred from the map (*constrained update*). The idea is to re-distribute the sample set more efficiently while maintaining the same number of particles. Thanks to road markings, the filter estimate is often well localized in the lateral direction. A new set of particles is placed along a generator curve $\mathcal{C}$ that is parallel to map road markings and has the same lateral offset as the previous filter estimate (Figure 3.32b). Thus, the new set of samples covers more space in the longitudinal direction and the weighting step improves the accuracy. Figure 3.32c shows the result obtained with the constrained update strategy. The filter is able to recover the correct position without having to increase the number of particles to populate more regions in the map.

3.5.4 The resampling step

A major problem is amplified through repetitive resampling: after some iterations the particles tend to concentrate into an identical particle copy, this is known as *particle degeneracy*. Particle degeneracy would eradicate particle diversity which is a crucial characteristic of the filter. As a general rule, the resampling step increases the variance of the particle filter as an estimator. On one hand, too much resampling increases the risk of losing diversity, while on the other hand, too little resampling puts many particles in regions of low probability. According to [Thrun, 2002a], there are two main strategies for *variance reduction*. The first strategy is to perform the resampling step according to the value of the variance of the importance weights. If it is high, the resampling should be done, otherwise, no resampling should be performed. Consequently, the resampling frequency is decreased. The second strategy, which is the adopted implementation in our filter, is known as *low variance sampling*. The pseudo code is given hereafter:

Data: $\mathcal{X}_k$
Result: $\bar{\mathcal{X}}_k$
1 $\bar{\mathcal{X}}_k = \emptyset$;
2 r = rand(0; N^{-1}) ;
3 $c = w_k^{[1]}$;
4 $i = 1$;
5 **for** m = 1 *to* N **do**
6 $\quad U = r + (m-1).N^{-1}$;
7 $\quad$ **while** $U > c$ **do**
8 $\quad\quad i = i + 1$;
9 $\quad\quad c = c + w_k^{[i]}$;
10 $\quad$ **end**
11 $\quad$ add $x_k^{[i]}$ to $\mathcal{X}_k$;
12 **end**

Algorithm 4: Low variance sampling algorithm [Thrun, 2002a]

Instead of sampling N particles independently, this strategy computes a random number and select particles according to this number and with a probability proportional to the corresponding weights. The random number is highlighted in line 2 of Algorithm 4. Then, the algorithm generates iteratively a fixed amount (= $\frac{1}{N}$) to r. The while loop in the algorithm searches for the first particle index whose cumulative sum of the weights is larger than U value (computed in Line 4 of Algorithm 4) as follows:

$$i = \arg\min_j \left(\sum_{m=1}^{j} w_k^{[m]} > U \right) \tag{3.60}$$

Figure 3.33 depicts the principle of low variance sampling. The particle selection is proportional to the relative weight. Indeed, particles with larger weights are selected more than one time and particles with smaller weight may not even be selected.

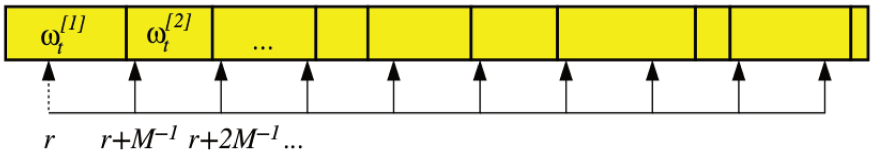


Figure 3.33: Principle of the low variance sampling [Thrun, 2002a]

Low variance sampling has many advantages. First, if the importance weights are all identical then the constructed new particle set remains the same as the input set.

Consequently, no particle is lost if no observation is available (i.e. no update step). Second, the dependent samples cycles through all particles rather than choosing them independently. As a result, larger subsets of samples are covered.

3.6 Conclusion

In this chapter, we presented a road perception module and a map-based localization approach. The road perception detects different highway primitive features such as lane markings, traffic signs, barriers and guardrail reflectors. These features are extracted from 3D LiDAR point clouds by coupling geometry-based and reflectivity-based approaches. Lane markings are detected by searching straight lines in a 2D horizontal reflectivity grid map, whereas barriers are detected by searching straight lines in a 2D height map. Traffic signs are detected using closed contour detector applied to a frontal reflectivity grid map. To eliminate false detections, we executed a RANSAC plane estimation to keep planar surfaces that satisfy certain criteria (orientation of the normal vector and the number of points (inliers) constructing the plane). We showed that our algorithm gives promising detection results but mistakenly detects vehicle plates as traffic signs. Finally, guardrail reflectors are detected by applying template matching using normalized cross correlation (NCC) to a frontal reflectivity grid map. Similarly, RANSAC plane estimation is used to filter false template matches. In contrast to traffic signs, the fixed geometric criteria to detect guardrail reflectors are not affected by license plates. According to qualitative results, we concluded that using the bumper-mounted LiDAR is more suitable to detect: barriers, guardrail reflectors and lane markings. The roof-mounted LiDAR is more appropriate for traffic signs detection.

The localization system is implemented using a particle filter where the detected features are matched to a third party highly accurate digital map. This map stores geometric road primitives: lane markings, traffic signs, road connections, etc. We proposed an improved version of Particle Filter (PF) which keeps the number of particles constant during the experiments but re-distribute them in an optimized manner to maximize the particle space coverage around the true vehicle position. This version is called *constrained update* particle filter. *Constrained* means that particles are generated taking into account the road map geometry. To evaluate our approach, we have conducted many experiments in a Renault highway-like test track and in a real highway environment in France (French A13 highway). We discuss the results in the up-coming chapter.

Chapter 4

Experimental results

Contents

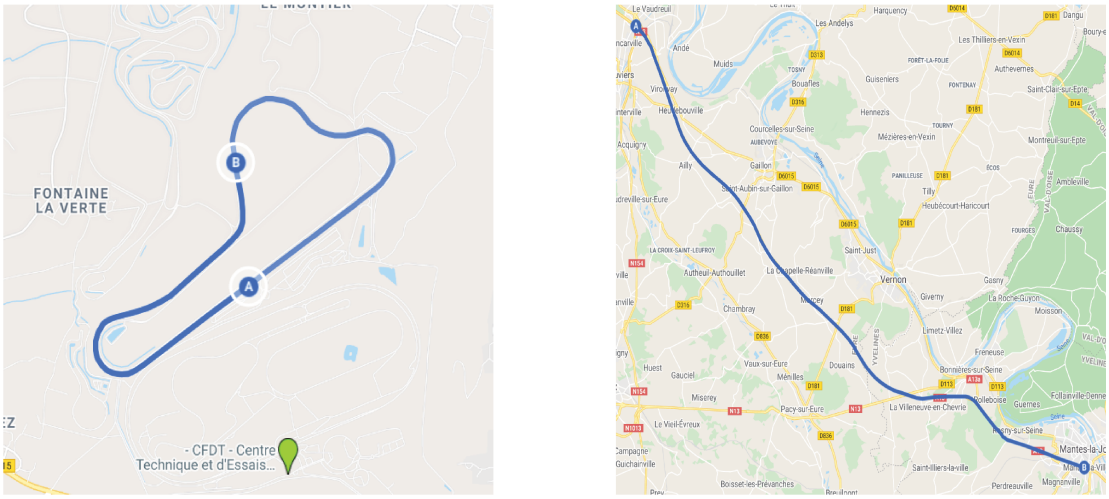
Résumé en français

Dans ce chapitre, nous proposons une description des expériences que nous avons menées afin de valider notre système de localisation sur carte. Nous avons testé notre approche sur deux pistes différentes. La première est une piste de voie rapide localisé au centre technique d'Aubevouye de Renault. Le deuxième essai a été effectué dans des conditions réelles de conduite sur l'autoroute A13. Nous avons testé notre système en variant la vitesse du véhicule (de 30Km/h à 90 km/h) pour étudier l'impact de la vitesse sur le précision de l'estimation de la position du véhicule. Enfin, le test sur l'autoroute A13 est un essai représentatif des conditions réelles de conduite dont l'existence d'obstacles.

4.1 Introduction

The aim of this chapter is to present the conducted experiments to evaluate our proposed localization approach and the achieved results. We tested our proposed solution in different driving scenarios and on two different highway roads. The first is an experimental test track (CTA2) of 5 km long located at CTA, Renault's Aubevoye's Technical Center (Figure 4.1a). This track is designed to exactly replicate a two-lane highway environment. The second is a section of the A13 highway of about 100 kilometers long, running from Paris and ending at Aubevoye (Figure 4.1b).

We have driven our prototype vehicle "MELO" at different speeds from 30 kph to 90 kph in CTA2 test track and up to 130 kph on A13 highway, and in different driving scenarios involving lane-keeping and lane-change maneuvers. On one hand, our test track provides a well-controlled environment, where tests can be performed in ideal and safe conditions, especially those that are not possible on real highway roads (e.g. driving at low speed). On the other hand, driving on A13 highway allowed us to experiment real traffic scenarios, where real world related problems are to be expected, such as missed detections, obstruction, dynamic objects etc. As for the maps, the A13 highway and the Renault test track were mapped by the same provider. Nevertheless, guardrail reflectors are not yet provided in the A13 highway map.



(a) Renault test track

(b) A13 Highway road

Figure 4.1: Highway roads used for evaluation of the localization system

4.2 Evaluation methodology

The evaluation methodology of the localization system consists in testing our approach under different use cases and to evaluate the impact of the road perception system on the accuracy of the localization outputs. Let us recall that the detected objects from the perception system are: lane markings, traffic signs, rail reflectors and the median barrier. Let us also recall that we have two different particle filtering implementations: the *constrained update strategy* and the *unconstrained (standard) update strategy*. The main idea of the adopted methodology is to compare the filters outputs by removing in each test one of the detected road objects. In addition, in order to simulate the GNSS signal loss we also evaluated the approach by disabling the GNSS inputs. However, the GNSS signal input is still indispensable for the initialization. The tested use cases are listed in Table 4.1 and are summarized below:

- **All update**: in this case, all the objects detected by the road perception module are utilized. However, it is important to note that this evaluation was only performed at CTA2, because the guardrail reflector layer is not yet available in the A13 highway map.
- **Traffic signs**: in this case, guardrail reflectors are not used. The goal is to evaluate the contribution of traffic signs to the localization accuracy. This has been tested for both implementations of the particle filter.
- **Traffic signs without GNSS**: similarly to the previous case but GNSS data is not used as measure for the update steps of the filter. The goal is to evaluate potential GNSS signal losses.

- **Lane markings only**: in this case, the median barrier is not considered and the update step is performed with lane markings only. The goal is to evaluate the multi-hypothesis problem.
- **Median barrier Only**: only the median barrier is used in the update step. The goal is to evaluate the contribution of median barrier to the localization accuracy.
- **Lane markings and median barrier**: in this case, lane markings and median barrier are used in the update step of the filter. The goal is to evaluate accuracy improvements obtained by fusing both measurements.

Configuration	Traffic signs	Rail reflectors	Lane markigns	Median barrier	GNSS data
All update	Yes	Yes	Yes	Yes	Yes
Traffic signs	Yes	No	Yes	Yes	Yes
Traffic signs without GNSS	Yes	No	Yes	Yes	No
Lane markings and median barrier	No	No	Yes	Yes	Yes
Lane markings only	No	No	Yes	No	Yes
Median barrier only	No	No	No	Yes	Yes

Table 4.1: Summary of the use cases and configurations for the evaluations processes

For all the experiments, the number of particles is a constant parameter and is properly defined to ensure a total length of 20 meters for the generated curve $\mathcal{C}$ in *the constrained update strategy* (10 meters in front of the vehicle and 10 meters behind). We draw particles along the curve $\mathcal{C}$ with a step size set to $0.1m$ thus requiring a total number of particles equals to $200 = \frac{20}{0.1}$. Moreover, to activate the *constrained update strategy*, the threshold value in Equation 3.59 is fixed to $w_{s,th} = 0.2$. The integration of GNSS inputs requires to set the value of ρ_{max}. In our case, we have chosen 15 meters because the goal is to develop a localization system that works with a low cost, degraded GNSS data.

4.2.1 Evaluation metrics

The evaluation of the localization system should be defined according to specific metrics to qualify its accuracy. Choosing a suitable evaluation metric is also important for the design of the overall autonomous navigation system. A typical metric is the computation of the euclidean distance between the localization position outputs and reference

positions obtained from a highly accurate ground truth. The euclidean distance is generally calculated with respect to a fixed Cartesian frame. Other important metrics for autonomous vehicle navigation are the cross-track and along-track distances that are the curvilinear abscissa defined with respect to the map.

A Absolute error

The calculation of the absolute errors is straightforward. Let $x_{1:T}$ be the estimated localization positions and $x^*_{1:T}$ the corresponding ground truth positions. The absolute error at time k is calculated as follows:

$$error_{abs,k} = \|x_k - x^*_k\| \tag{4.1}$$

B Along-track and cross-track errors

The along-track (AT) and cross-track (CT) errors are signed and calculated with respect to the curvilinear coordinates attached to the map. Contrary to the absolute error, they implicitly incorporate the map error. In other words, the absolute error is the result of the localization system error and the ground truth position error, whereas both along-track and cross-track errors are calculated by passing through the map which in turns has errors. Nevertheless, both metrics are expected to yield similar results with highly-accurate maps.

The calculation of cross-track and along-track errors depends on the representation of lanes in the map. A good comparison of map representations and their impact on the calculation on the errors is given in [Héry et al., 2018]. In our case, the lanes are modeled by polylines, thus, we approximate the errors by using the map tracker points. Map tracker points are the projection points of one vehicle position into map polylines (cf. Parag.3.3.2.D). Let x_k, x^*_k be the filter estimate and the associated ground truth position at time k. Let $\{p_{1,k}, p_{2,k}, ..., p_{n,k}\}$ and $\{p^*_{1,k}, p^*_{2,k}, ..., p^*_{n,k}\}$ be their corresponding map tracker points. Hence, the associated errors are computed as:

$$CT_{error} = \frac{1}{n}\sum_{j=1}^{n}(\|p_{j,k} - x_k\| - \|p^*_{j,k} - x^*_k\|) \tag{4.2}$$

$$AT_{error} = \frac{1}{n}\sum_{j=1}^{n}\|p_{j,k} - p^*_{j,k}\| \times sign(p_{j,k}, p^*_{j,k}) \tag{4.3}$$

Considering λ^*_k as the heading of the ground truth at time k, the $sign$ function is given by:

$$sign(p_{j,k}, p^*_{j,k}) = \begin{cases} 1\ , \text{ if } u.v > 0 \\ -1\ , \text{ if } u.v \leq 0 \end{cases} \tag{4.4}$$

Where $u = [\cos(\lambda^*_k), \sin(\lambda^*_k)]^T$, $v = \overrightarrow{p_{j,k}p^*_{j,k}}$ and $(.)$ is the inner product operator.

4.3 Experimental results

4.3.1 CTA2 test track results

Our mule car MELO has been driven on the CTA2 test track at different speeds, in order to assess the impact of the ego speed on the localization accuracy. At first, all detected objects from our road perception module have been used. Then, we disabled the guard rail reflector input in order to evaluate its impact on the performance. In both cases, the two particle filter implementation are tested and compared.

Table 4.2: Localization accuracy results on CTA2 test track. All detected objects from our road perception module are used: traffic signs, guard rail reflectors, lane markings and barriers.

	30 kph	50 kph	70 kph	90 kph
Absolute error	μ : 0.52 std: 0.33	μ : 0.67 std: 0.35	μ : 0.8 std: 0.56	μ : 1.08 std: 0.75
Along-track error	μ : -0.21 std: 0.54	μ : -0.43 std: 0.57	μ : -0.53 std: 0.78	μ : -0.83 std: 0.93
Cross-track error	μ : -0.05 std: 0.15	μ : -0.08 std: 0.13	μ : -0.07 std: 0.14	μ : -0.07 std: 0.16

(a) Particle filter implementation with constrained update

	30 kph	50 kph	70 kph	90 kph
Absolute error	μ : 0.52 std: 0.33	μ : 0.67 std: 0.35	μ : 0.8 std: 0.56	μ : 1.08 std: 0.75
Along-track error	μ : -0.21 std: 0.54	μ : -0.43 std: 0.57	μ : -0.53 std: 0.78	μ : -0.83 std: 0.93
Cross-track error	μ : -0.05 std: 0.15	μ : -0.08 std: 0.13	μ : -0.07 std: 0.14	μ : -0.07 std: 0.16

(b) Particle filter implementation without constrained update

Tables 4.2a and 4.2b summarize mean and standard deviation values (μ and std) for the absolute, along-track and cross-track errors using all detected objects from our road perception module. The error graphs of the measurement records are illustrated in Figure A.1.

As we can see in Table 4.2, the localization accuracy is affected by the vehicle speed. Indeed, the mean absolute error increases from $0.52m$ at 30 kph to $1.08m$ at 90 kph. Regarding the cross-track error, the vehicle speed has no significant effect. Indeed, the y-component (i.e. lateral component) of the ego velocity vector is most often very small in all driving maneuvers (lane-change and lane-keeping). The standard update strategy and the constrained update strategy give similar results for all vehicle speeds. This is expected because the constrained update strategy is basically designed to tackle the particle deprivation problem that occurs when the filter hypothesis start the drift

away from the most likely area around the true position. Since guard rail reflectors are repetitive features, the deprivation problem is very unlikely to occur.

Table 4.3: Localization accuracy results on CTA2 test track without the guard rail reflector input

	30 kph	50 kph	70 kph	90 kph
Absolute error	μ : 1.79 std: 1.84	μ : 1.28 std: 0.70	μ : 3.62 std: 2.38	μ : 4.62 std: 2.46
Along-track error	μ : -0.33 std: 2.54	μ : -1.19 std: 0.80	μ : -3.59 std: 2.47	μ : -0.37 std: 5.14
Cross-track error	μ : -0.05 std: 0.14	μ : -0.08 std: 0.14	μ : -0.06 std: 0.15	μ : -0.08 std: 0.14

(a) Particle filter implementation with constrained update

	30 kph	50 kph	70 kph	90 kph
Absolute error	μ : 3.51 std: 1.82	μ : 1.28 std: 0.70	μ : 6.02 std: 2.27	μ : 5.45 std: 2.45
Along-track error	μ : 0.79 std: 3.95	μ : -1.19 std: 0.80	μ : -6.11 std: 2.27	μ : 5.45 std: 2.32
Cross-track error	μ : -0.05 std: 0.16	μ : -0.08 std: 0.14	μ : -0.05 std: 0.17	μ : -0.05 std: 0.32

(b) Particle filter implementation without constrained update

The same scenarios have been replayed while disabling the guard rail reflector input. As we can see in Table 4.3, the localization errors significantly increase in comparison with Table 4.2. Indeed, guard rail reflectors are repetitive and play an important role in reducing the along-track errors. This is not the case for traffic signs that can be absent for hundred of meters. The importance of the proposed constrained update strategy can be observed in Figure A.2. At different vehicle speeds (30 kph, 70 kph and 90 kph), the particles are initialized and their relative positions are updated using inertial data for a period of time without encountering any traffic sign. This makes the longitudinal drift increases; therefore getting the particles far from the true position of the vehicle. When a traffic sign is detected, the standard update strategy is not capable of covering back to the true position since all the particles have drifted away, hence, the calculated weights are very low (close to zero). However, with the proposed approach, the filter rapidly converges back around the true position of the vehicle. This is manifested by the instantaneous decrease of the absolute and along-track errors illustrated in Figure A.2. At 50 kph, the constrained update strategy is not activated (i.e $w_{s,max}$ remains greater than $w_{s,th}$) because the filter is initialized near a traffic sign location, which allowed the particles to stay close to the true position of the vehicle.

4.3.2 A13 highway results

Following the test track evaluations, we have conducted real world experimentations on the A13 highway to assess the performance of our localization approach. Guard rail reflectors are not used in these experiments because they are not yet available in the map. The results discussed hereafter are computed from a section of approximately 6 kilometers long. The conditions on the A13 highway are degraded in comparison with the Renault test track. At first, LiDAR reflectivity returns are sometimes too low that the lane perception module does not always detect the lines. Second, moving objects in the environment (e.g. other vehicles) cause occlusions that seriously impact the detection of traffic signs and lane markings. Finally, since the A13 has three lanes, the multi-lane hypothesis problem occurs. The main goals of the experiments are:

1. To evaluate our localization system when both lane markings and median barrier inputs are available and assess the contribution of each object type.
2. To evaluate our localization system without the GNSS inputs as signal losses occur in real-world scenarios (e.g. tunnels).
3. To evaluate our localization system using all the detectable features by our road perception module (except for guard rail reflectors)
4. To evaluate our localization system performance in a lane change scenario.

Figure 4.2 illustrates the separate contributions of each of lane markings and median barrier to the cross-track localization error, as well as their joint contribution when simultaneously available. When only the lane markings are used, the cross-track error is in the order of 4m and remains at that value during a time interval $t_1 = [0 - 900]$ frames. Then, from $t = 900$ frames onwards, the error is shifted near zero. Since the observed shift is in the order of one lane width, this can be explained by the fact that the filter converged to the adjacent (wrong) lane in the time interval t_1. In $t_2 = [900 - 1700]$ frames, the filter converged back to the correct lane. The ambiguity in t_1 is mainly due to multi-lane hypothesis. This problem occurs when the detected lane markings are not sufficient to decide which lane the ego vehicle is in. A typical example is when the lane marking detection module only detects the lines of the ego lane in a multi-lane roads resulting thus to equal likelihood of occurences in all road lanes.

	lanesOnly	barrierOnly	lanesAndBarrier
Cross-track error	μ : 1.74 std: 1.76	μ : 0.43 std: 0.65	μ : 0.0 std: 0.13

Table 4.4: Cross-track localization error using separately lane markings, median barrier and the fusion of both

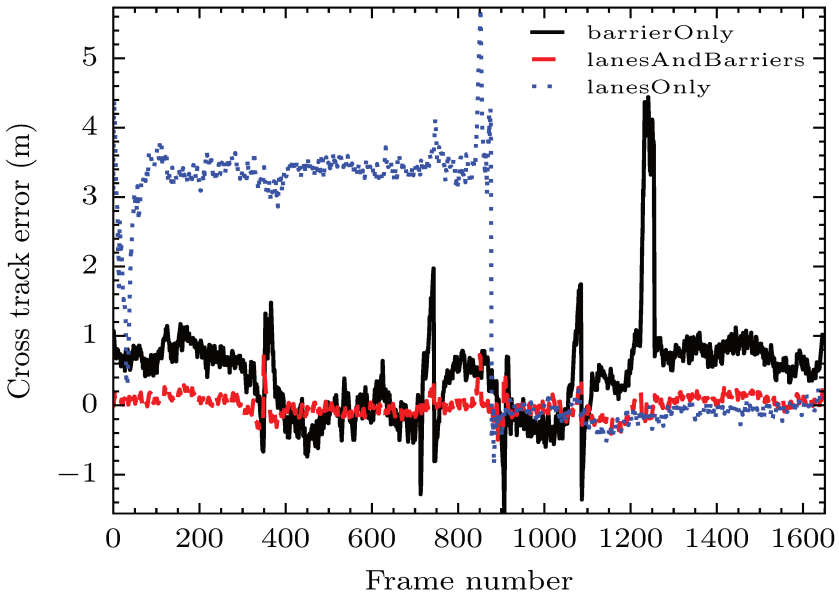


Figure 4.2: Cross-track localization error using separately lane markings, median barrier and the fusion of both on A13 highway section.

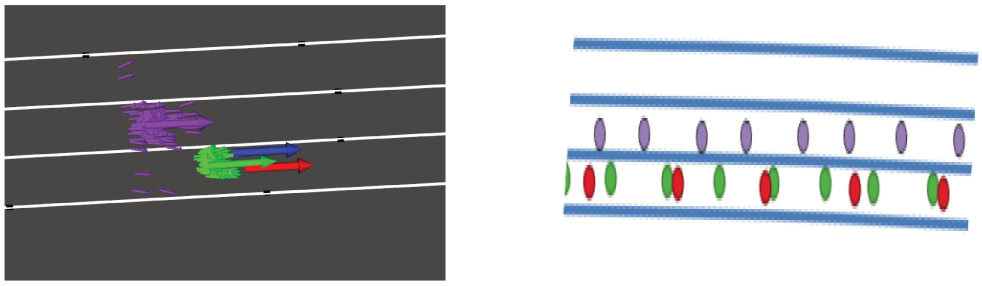
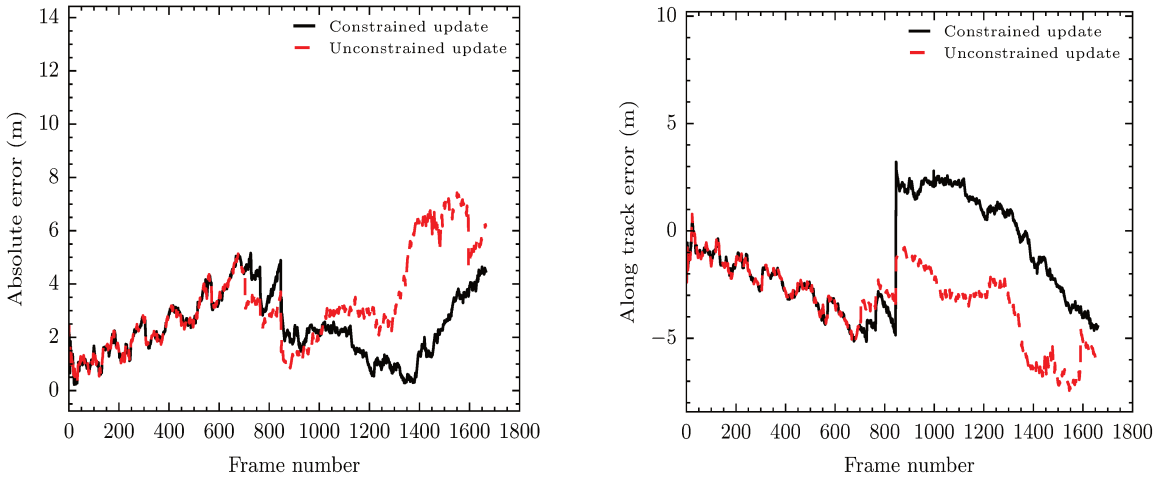


Figure 4.3: Multi-lane hypothesis problem. (Left) two different filter implementations. Purple: lane markings only, green: lane markings and median barrier, blue arrow: ublox GNSS, red arrow: GNSS/RTK (ground truth). Small arrows are filter particles and big arrows are the mean estimate (i.e mean of particles), red arrow is GNSS/RTK position and blue arrow is an automotive grade GNSS position. (Right) Trajectory path of particle position's mean estimates. Purple: trajectory path of lane markings only, green: trajectory path of lane markings and median barrier and red trajectory path of GNSS/RTK positions.

Figure 4.2 also illustrates the cross-track error obtained by median barrier only. The filter estimate is more stable within the correct lane, but is not well positioned in it (mean error is 0.43 m). Hence, using only the median is capable of achieving WHICH LANE localization level. The accuracy of the cross-track localization can be further improved by jointly using lane markings and the median barrier (mean error 0 and standard deviation 0.13 m), therefore achieving the WHERE IN LANE level. Table 4.4 summarizes mean and standard deviation values of the cross-track localization errors for each configuration. The multi-lane hypothesis problem is illustrated in Figure 4.3.

Figure 4.4 illustrates the along-track and the absolute localization errors obtained by using all perception objects: lane markings, median barrier and traffic signs. The aim is to assess their impact on the along-track (i.e. longitudinal) error in real driving conditions and for both update implementations (constrained and unconstrained). The absolute error is always positive, whereas the along-track error is signed. The sign of the along-track error indicates the position of the filter estimate with respect to the GNSS/RTK position. Positive value indicates that the estimate is in front of the RTK position and negative value if the filter estimate is behind the RTK position.



(a) Absolute error

(b) Along-track error

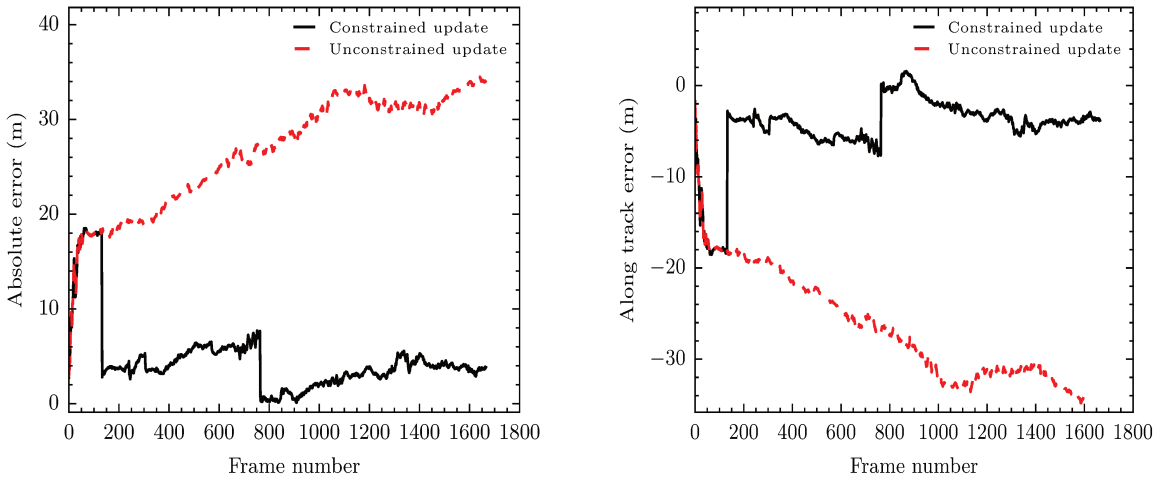
Figure 4.4: Along-track and absolute error graphs using all perception objects: lane markings, median barrier and traffic signs evaluated for both update strategy (constrained and unconstrained).

Table 4.5 shows the mean μ and the standard deviation std values for absolute, along-track and cross-track localization errors. Results demonstrate that the constrained update implementation outperforms the standard implementation as for the absolute and the along-track errors, while the cross-track error is nearly the same for both implementations which is expected since traffic signs are mainly used to target the longitudinal error. However, the obtained accuracy is far from being comparable to

	all perception objects (pf constrained)	all perception objects (pf unconstrained)
Absolute error (m)	μ : 2.36 *std*: 1.19	μ : 3.22 *std*: 1.74
Along-track error (m)	μ : -1.34 *std*: 2.26	μ : -3.21 *std*: 1.76
Cross-track error (m)	μ : 0.0 *std*: 0.15	μ : 0.0 *std*: 0.16

Table 4.5: Absolute, along-track and cross-track localization mean μ and standard deviation *std* error values using all perception objects: lane markings, median barrier and traffic signs for both update strategy (constrained and unconstrained).

the obtained accuracy using guard rail reflectors. Indeed, traffic signs are not repetitive objects in highway roads and may be absent for hundred of meters. In the temporary absence of traffic signs, the localization system only relies on GNSS data to bound the IMU integration drifts.



(a) Absolute error

(b) Along-track error

Figure 4.5: Absolute and along-track error graphs using all perception objects except GNSS data for the update steps.

To assess the impact of a GNSS signal loss (e.g. tunnels), Figure 4.5 illustrates the error graphs for the along-track and the absolute localization errors using all objects from the perception module without integrating GNSS data. The obtained results for both filter implementation are completely different. Table 4.6 depicts the mean and standard deviation of along-track, absolute and cross-track errors. The unconstrained update filter implementation significantly drifts from the ground truth and reaches a

	all perception objects without GNSS (Pf constrained)	all perception objects without GNSS (Pf unconstrained)
Absolute error (m)	μ : 4.68 std: 3.79	μ : 26.52 std: 5.85
Along-track error (m)	μ : -4.59 std: 3.9	μ : -26.42 std: 5.90
Cross-track error (m)	μ : 0.0 std: 0.15	μ : 0.0 std: 0.16

Table 4.6: Absolute, along-track and cross-track mean μ and standard deviation std error values using all perception objects except GNSS data for the update steps.

mean value of -26 m (i.e. 26 meters behind the RTK position). However, the constrained update filter implementation is capable of converging back near the true position.

Figure 4.6 illustrates a lane change scenario. In this scenario, traffic signs, lane markings and median barrier are all used as inputs and the results are compared to GNSS/RTK ground truth. As we can see, our localization system is capable of accurately performing in a lane-change maneuver with a score rate of 100%. This score rate is derived by comparing the lane assignment of each filter estimate with the GNSS/RTK ground truth.

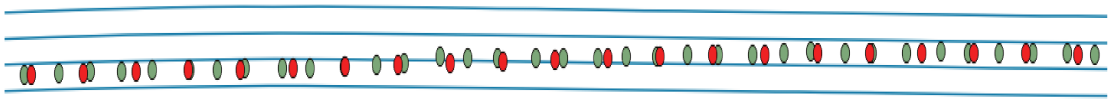


Figure 4.6: Lane Change scenario. Red circle: trajectory of GNSS/RTK ground truth. Green circles: trajectory of the filter estimate using traffic signs, lane markings and median barrier. Blue lines are lane marking lines stored in the map

4.4 Concluding remarks

Experimental tests have been conducted in two highway environments: the Renault test track (CTA2) in Aubevoye and a section of the A13 highway. At CTA2, we conducted experiments for different vehicle velocities in order to evaluate the impact of speed variations on the localization accuracy. Unsurprisingly, the impact of velocity is mainly observed in the longitudinal direction (along-track error) rather than the lateral direction (cross-track error) as the direction of the velocity vector is dominated by the longitudinal component (v_x component). We also tested the contribution of each detected object provided by the LiDAR perception module. On one hand, experimental results show that lane markings and the median barrier are necessary to localize the

vehicle in the lateral (cross-track) direction and have minor contributions to the longitudinal (along-track) direction. On the other hand, guardrail reflectors and traffic signs are necessary to localize the vehicle in the longitudinal direction.

For lateral direction, the best cross-track accuracies are obtained by jointly using lane markings and median barrier. The obtained accuracy is below 20cm for all tested vehicle speeds. However, if one of these detected objects is not taken into account, the cross-track accuracy can be deteriorated. Indeed, the only use of lane markings is not sufficient and may lead to the multi-lane hypothesis problem. This problem occurs when some lane markings are not detected and is characterized by a convergence of the filter to the wrong ego lane, leading therefore, to a significant cross-track error (shifted by $\sim$ lane width). On the other side, the use of median barrier allows to robustly estimate the correct ego lane index but is not capable to infer the position of the vehicle in it (i.e *WHICH LANE* localization level). This is a conceptual choice in our approach, the median barrier are used to mitigate the multi-hypothesis problem due to missed lane marking detections.

For longitudinal direction, we studied the impact of three different inputs: traffic signs, guardrail reflectors and GNSS data. In contrast to the cross-track error, the along-track error is around 3 meters when using traffic signs and GNSS inputs (e.g. A13 results) and is around 1 meter when adding guardrail reflectors (e.g. CTA2 results). This is because guardrail reflectors are repetitive features (e.g. spatial periodicity of 40 meters at CTA2), whereas traffic signs may be absent for hundred of meters on highways. In the absence of traffic signs and guardrail reflectors, GNSS data is used to limit the cumulative drift obtained from odometry integrations by enclosing the particles within an ellipsoid error (3σ-ellipsoid) of GNSS data.

One of the contributions of this study is the proposal of an improved version of particle filter, named *contrained-update particle filter*, that tackles the *particle deprivation problem*. Concretely, in our case, the deprivation problem occurs when guardrail reflectors are not used and, consequently, the localization system may be subject to cumulative drift between two detected traffic signs. The cumulative drift can be significantly high (i.e. because two consecutive traffic signs can be very distant on highways) so that the particles tend to gradually vanish from the spatial region around the true position. Different approaches have been addressed in the literature to alleviate this problem by increasing the number of particles to populate more regions in the space. However, increasing the number of particles increases the computational complexity of the filter. In our approach, the deprivation problem is solved by efficiently re-distributing the set of particles using the geometry of the road extracted from the map. Experimental results showed that the proposed improvement gives better along-track accuracies than the standard particle filter implementation.

Chapter 5

Conclusion and perspectives

Résumé en français

Le présent chapitre résume les travaux présentés dans ce manuscrit. Nous résumons d'abord le problème qu'on chercher à résoudre et les contributions scientifiques apportées. Ensuite, nous détaillons les résultats expérimentaux pour la validation de notre système.

This chapter concludes the research work presented in this manuscript. At first, we summarize the problem and our proposed solution to address it. Then, we focus on main results obtained from the experimental evaluation. Finally, we discuss the open questions and perspectives for future research.

5.1 Proposed approach

In this thesis, we addressed the development of a highly accurate localization system to enable highly automated driving functions on highway roads. Based on the state-of-art review of mapping and localization techniques, we proposed a global localization system architecture with position tracking, feature detection (perception module) and map matching (localization module). We have chosen LIDAR as the main sensor for environment perception, thanks to its range measurement accuracy and its robustness against lighting conditions. Then, we developed a LIDAR-based perception module and a localization module based on particle filtering and a very accurate third party map. The perception module detects road primitive features that are lane markings, traffic signs, median barriers and guardrail reflectors. The detection processes mainly rely on the shape and reflectance values of these features. Before detecting lane markings, we developed a road segmentation method to eliminate the impact of high reflective isolated points in the environment that may lead to false lane marking detections. The road segmentation approach uses a geometric analysis of layer points on the road

surface by locally approximating the geometry with a circular arc model. Once the road surface points are filtered and kept, a 2D reflectivity grid map is constructed and a Hough transform is applied in order to extract straight lines. The segmentation of the road surface allows to define a low reflectivity threshold since the asphalt points have low reflectivity values. The straight line model for lane markings is a good approximation on highway roads since they are designed to have low curvature values. In addition, the perception module is mainly designed to feed a localization algorithm, thus, the detection range can be set in the vicinity of the vehicle (10-20 meters for lane markings). The detection of the median barrier is very similar to the method used to detect lane markings except that we extracted it from a height grid map instead of a reflectivity grid map. Finally, to cope with possible occlusions, a standard tracking algorithm based on Kalman Filter is implemented.

Detections of traffic signs and guardrail reflectors are performed on a front reflectivity grid map that is obtained by projecting LiDAR point cloud onto a 2D front view. On one hand, traffic signs are considered to be high reflective plane surfaces whose normal vectors are parallel to the driving direction. On the other hand, guardrail reflectors are considered to be a cluster of high reflective points in planar surfaces characterized by a normal vector that is perpendicular to the driving direction. For both features, the detection process goes through three major steps. First, a detection of Regions Of Interests (ROIs) is executed on the corresponding front reflectivity grid maps. For traffic signs, a closed contour operator is applied to a 2D image, that is constructed from the reflectivity values of the front grid map, in order to obtain bounding boxes indicating potential candidates of traffic signs. Since LiDAR points are sparse, we applied a dilation to the 2D image in order to fill the gaps within the closed contours. Regarding guardrail reflectors, we applied a template matching algorithm where the template is a selected sample from our dataset. The template matching is implemented using a Normalized Cross-Correlation (NCC) technique. The second step after detecting ROIs is geometric-based filtering approach. For the retained candidates, we applied a RANSAC plane estimation in order to keep only plane surfaces that satisfy certain conditions (mainly the directions of the normal vectors and the number of inlier points in the RANSAC process).

The localization approach is based on an implementation of the particle filtering algorithm and takes as inputs the perception data from LiDAR sensors and a highly accurate third party digital map. Particle filter searches for the best associations between perception data and map attributes through manipulating multiple vehicle position hypothesis called "Particles". The best particles are those for which the projected map attributes to particle positions are very close (metrically speaking) to the perceived objects. A key contribution of the thesis is the proposal of an improved version of particle filtering, namely *constrained-update* particle filtering, which tackles the *particle deprivation problem* without modifying the number of particles.

Experimental tests have been conducted on two different highway roads. The first is a Renault highway-like test track (CTA2) situated at Aubevoye, France. The second

is a section the A13 French highway. At CTA2, we conducted experiments for different vehicle velocities in order to evaluate the impact of the speed. We also tested the contribution of guardrail reflectors on the accuracy of the localization system. In the A13 section, the vehicle is moving at high speeds (up to 130 kph), our approach is tested only with the traffic signs as guardrail reflectors are not yet mapped for this road. Error evaluations have been studied with two different metrics: the absolute error from one side, the cross-track (lateral) and along track (longitudinal) errors from the other side. The main comments that can be given are as follows:

1. Lane markings and the median barrier are necessary to localize the vehicle in the lateral (cross-track) direction. The overall cross-track accuracy is below $20cm$ for all tested velocities. Indeed, velocity impacts are rather to be expected in the longitudinal direction than the lateral direction. However, this accuracy is conditioned to the joint use of lane markings and median barrier. On one hand, lane markings alone are not sufficient and may lead to the multi-lane hypothesis problem. On the other hand, the median barrier alone is only capable to achieve the *WHICH LANE* localization level. When fusing both features, the system is capable to achieve the *WHERE IN LANE* localization level.

2. In the longitudinal direction, we studied the impact of three different inputs: traffic signs, guardrail reflectors and GNSS data. In contrast to the cross-track error, the along-track error is highly sensitive to speed variations (e.g. $1.79m$ for 30 Kph .Vs. $4.69m$ for 90 kph at CTA2). In general, the along-track error is around 3 meters when using traffic signs and GNSS inputs (e.g. A13 results) and is around 1 meter when adding guardrail reflectors (e.g. $0.52m$ at 30 kph .Vs. $1.08m$ at 90 kph at CTA2). This is because guardrail reflectors are repetitive features (e.g. spatial periodicity of 40 meters at CTA2), whereas traffic signs may be absent for hundred of meters on highways. This allows to validate the concept of using guardrail reflectors as promising features for localization. Finally, in the absence of traffic signs and guardrail reflectors, GNSS data is used to limit the cumulative drift obtained from odometry integrations by enclosing the particles within an ellipsoid error (3σ-ellipsoid) of GNSS data.

3. The *constrained-update* particle filter implementation gives similar accuracies in the cross-track direction and better accuracies in the along-track direction compared to the standard implementation, except for the case where guardrail reflectors are used. Indeed, the constrained-update implementation is only triggered when the particle weights tend to be very low, which is very unlikely using guardrail reflectors. Nevertheless, when guardrail reflectors are not integrated, the difference can be significantly high. For example, on A13 highway, the absolute error is reduced from $3.22m$ to $2.36m$ when using traffic signs and GNSS data, and from $26.52m$ to $4.68m$ when only traffic signs are available (e.g. GNSS signal loss scenario).

5.2 Future research

This section introduces several improvements that can be integrated in the current system.

A Extension of the A13 map

The map of the A13 highway section did not include guardrail reflectors layer. An investigation can be done to evaluate the localization system with the use of guardrail reflectors in the particle filter update step. At this moment, we used the CTA2 test track to validate the concept.

B Integration of a vision system

One major improvement would be to replace the inertial data of the internal IMU sensor with oVisual odometry (VO). VO systems are more accurate and give better results in terms of displacement calculation. The impact of the integration of the IMU data can be seen when traffic signs are not detected and when no guardrail reflectors exist. In this case, the particle filter solely rely on integrating inertial data to predict the position in the longitudinal direction. We believe that with a visual odometry system, the cumulative drift will be reduced.

C An evaluation protocol of the perception module

In this work, our proposed approach was evaluated based on the outputs of the global localization system. The results of the localization implicitly contain the accuracy of perception. However, perception can be also used for other purposes such as mapping, obstacle detection and avoidance, etc. Therefore, an evaluation protocol of perception using a highly accurate map as ground truth could be investigated.

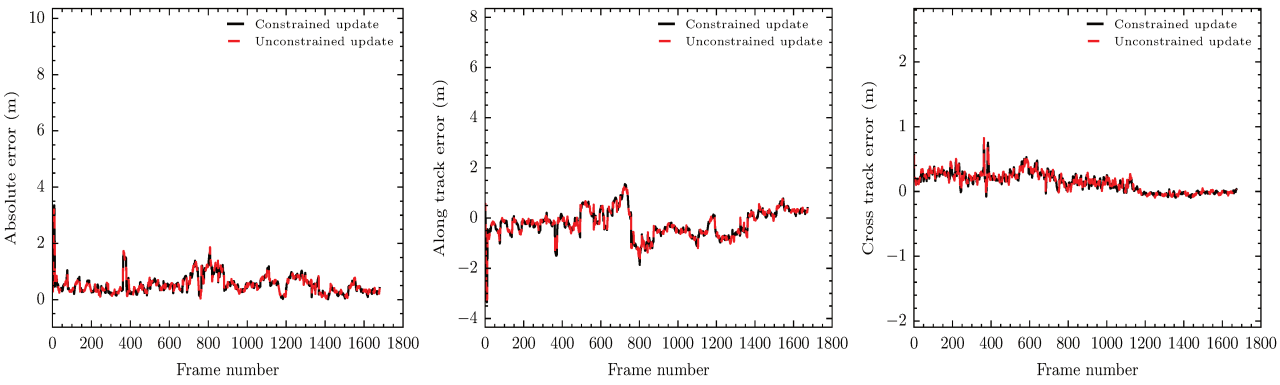
D Vision-based Artificial intelligence (AI) algorithms to improve LiDAR perception

Vision-based AI algorithms can be integrate to improve the robustness of the LiDAR perception system. Indeed, by labeling specific road features (lane markings, traffic signs, rail reflectors, etc), we can train AI algorithms to detect these features and to fuse the results with the LiDAR perception algorithms.

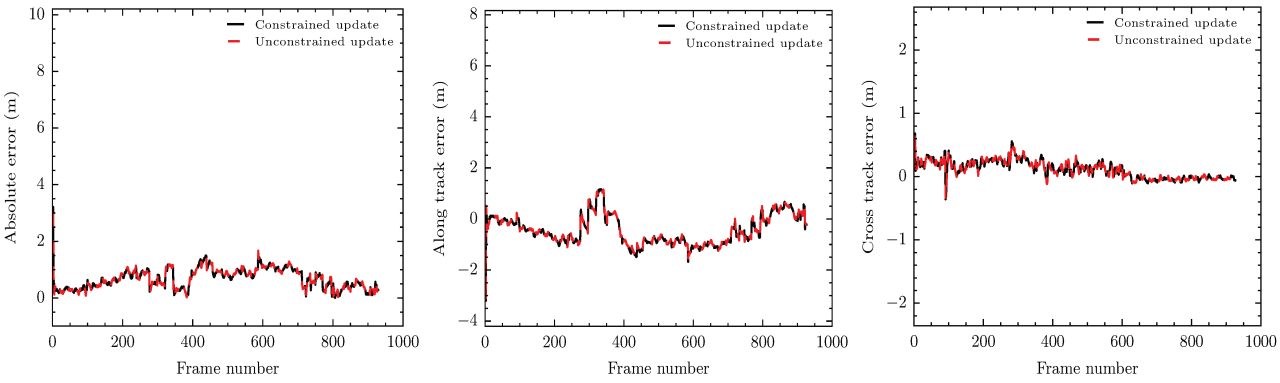
Appendices

Appendix A

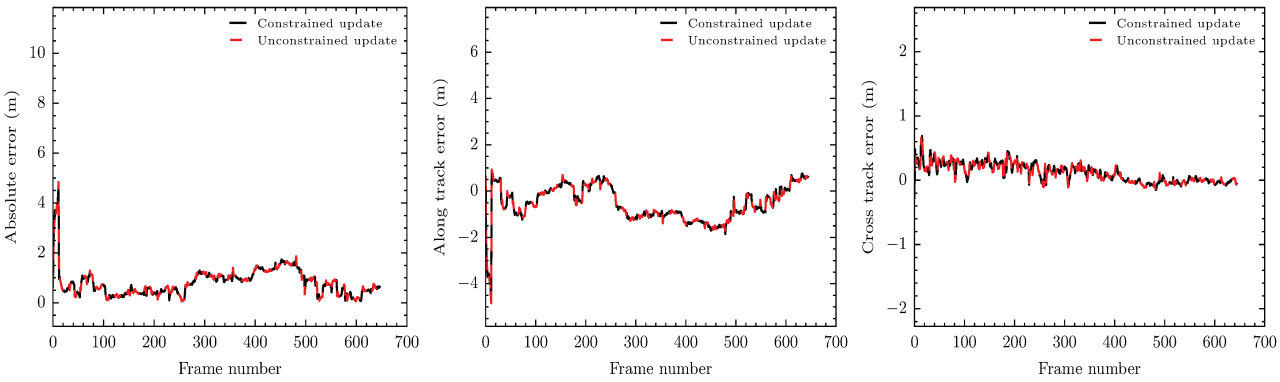
Figures: results at CTA2



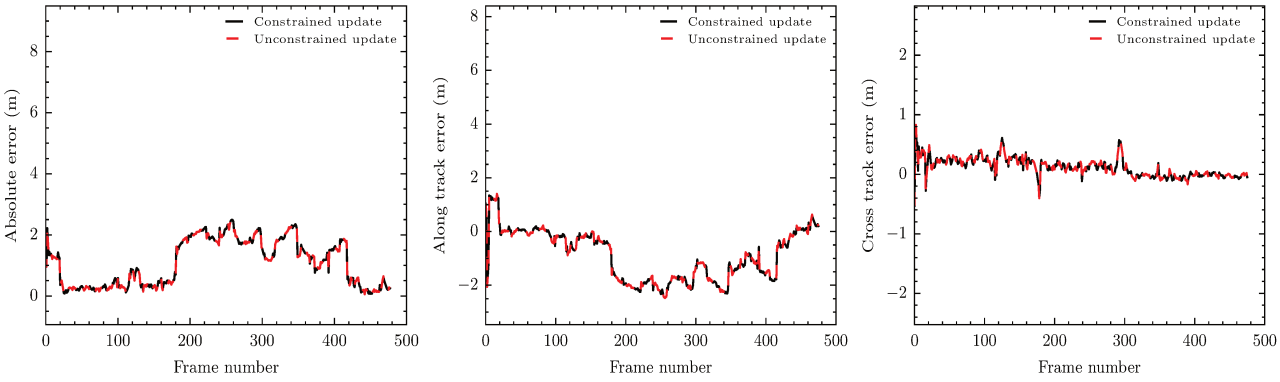
(a) Absolute error graph: 30 Kph **(b)** Along-track error graph: 30 Kph **(c)** Cross-track error graph: 30 Kph



(d) Absolute error graph: 50 Kph **(e)** Along-track error graph: 50 Kph **(f)** Cross-track error graph: 50 Kph

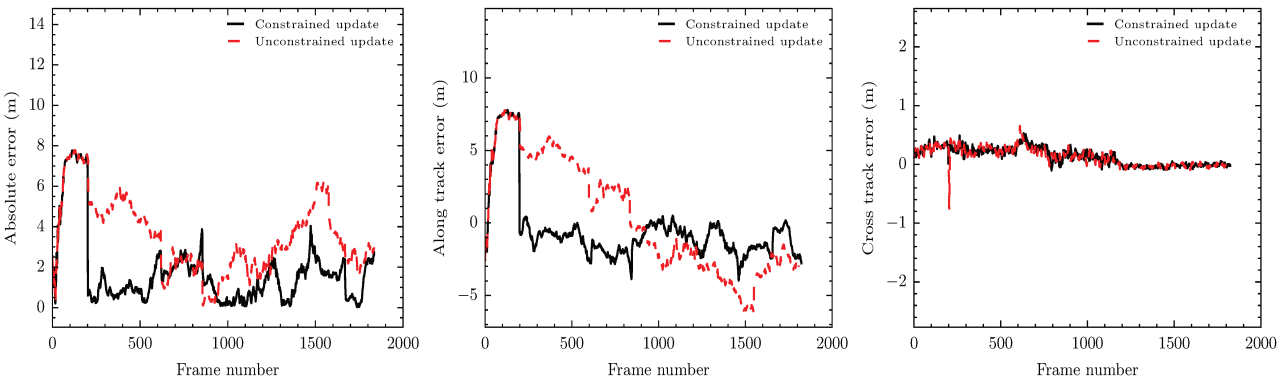


(g) Absolute error graph: 70 Kph **(h)** Along-track error graph: 70 Kph **(i)** Cross-track error graph: 70 Kph

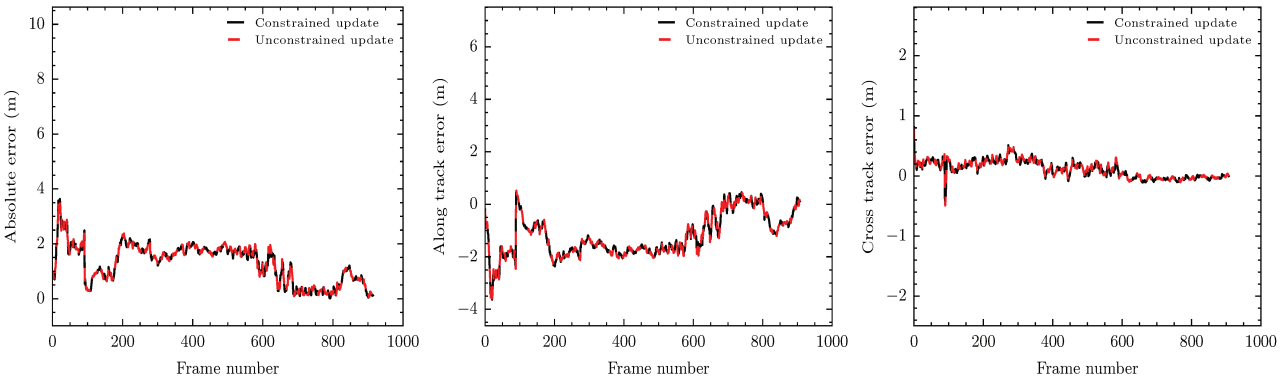


(j) Absolute error graph: 90 Kph **(k)** Along-track error graph: 90 Kph **(l)** Cross-track error graph: 90 Kph

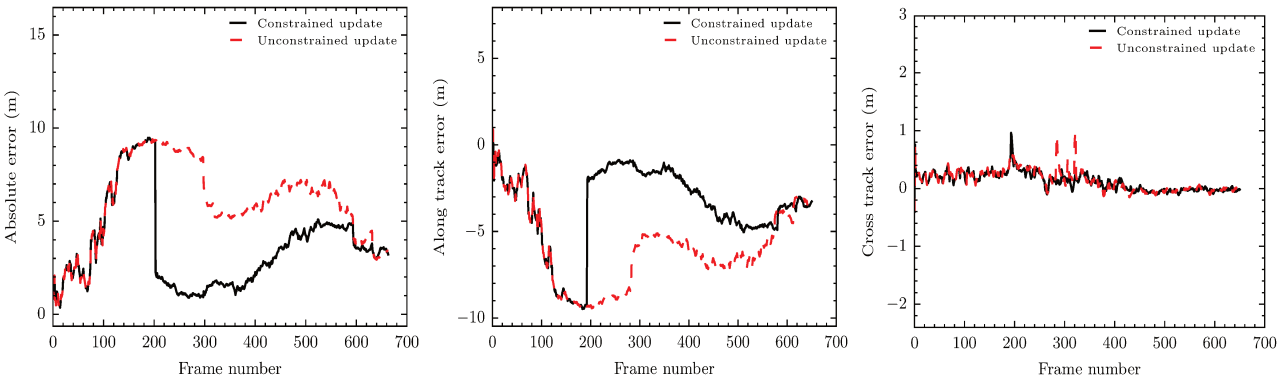
Figure A.1: Localization error graphs of experimental results on CTA2 test track. All detected objects from our road perception module are used: traffic signs, guard rail reflectors, lane markings and barriers.



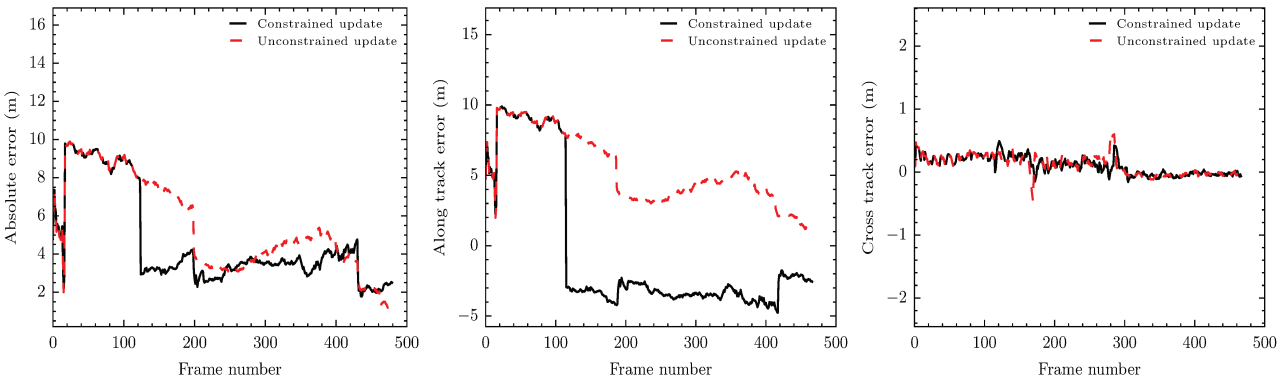
(a) Absolute error graph: 30 Kph **(b)** Along-track error graph: 30 Kph **(c)** Cross-track error graph: 30 Kph



(d) Absolute error graph: 50 Kph **(e)** Absolute error graph: 50 Kph **(f)** Absolute error graph: 50 Kph



(g) Absolute error graph: 70 Kph **(h)** Absolute error graph: 70 Kph **(i)** Absolute error graph: 70 Kph



(j) Absolute error graph: 90 Kph **(k)** Absolute error graph: 90 Kph **(l)** Absolute error graph: 90 Kph

Figure A.2: Localization error graphs of experimental results on CTA2 test track, without the guard rail reflector input.

Bibliography

[Her, 2019] (2019). Here hd live map - on the road towards autonomous driving. https://www.here.com/products/automotive/hd-maps. Accessed: 2019-05-09.

[Tom, 2019] (2019). Tomtom hd map with roaddna. https://www.tomtom.com/automotive/automotive-solutions/automated-driving/hd-map-roaddna. Accessed: 2019-05-09.

[SAE, 2020] (2020). Sae international releases updated visual chart for its "levels of driving automation" standard for self-driving vehicles. https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles. Accessed: 2020-04-14.

[Álvarez et al., 2007] Álvarez, J. M., López, A. M., and Baldrich, R. (2007). Shadow Resistant Road Segmentation from a Mobile Monocular System. In *Pattern Recognition and Image Analysis*, pages 9–16. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Aly, 2008] Aly, M. (2008). Real time detection of lane markers in urban streets. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 7–12.

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

[Baró et al., 2009] Baró, X., Escalera, S., Vitrià, J., Pujol, O., and Radeva, P. (2009). Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):113–126.

[Bay et al.,] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, (3):346–359.

[Bender et al., 2014] Bender, P., Ziegler, J., and Stiller, C. (2014). Lanelets: Efficient map representation for autonomous driving. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 420–425. IEEE.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606.

[Betaille and Toledo-Moreo, 2010] Betaille, D. and Toledo-Moreo, R. (2010). Creating enhanced maps for lane-level vehicle navigation. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):786–798.

[Beylkin, 1987] Beylkin, G. (1987). Discrete Radon Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(2):162–172.

[Biber and Strasser, 2003] Biber, P. and Strasser, W. (2003). The normal distributions transform: a new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2743–2748. IEEE.

[Borghi and Brugali, 1995] Borghi, G. and Brugali, D. (1995). Autonomous Map Learning for a multi-sensor mobile robot using diktiometric representation and negotiation mechanism. *Proceedings International Conference on Advanced Robotics*, (September):1–8.

[Bosse and Zlot, 2009] Bosse, M. and Zlot, R. (2009). Continuous 3D scan-matching with a spinning 2D laser. *IEEE International Conference on Robotics and Automation*, pages 4312–4319.

[Bresson et al., 2017] Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):1–1.

[Cadena et al., 2016] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

[Canny, 1986] Canny, J. F. (1986). A Computational approach to edge detection. *IEEE Trans. PAMI*, 8(6):112–131.

[Caselitz et al., 2016] Caselitz, T., Steder, B., Ruhnke, M., and Burgard, W. (2016). Monocular Camera Localization in 3D LiDAR Maps. pages 1926–1931.

[Chai et al., 2002] Chai, L., Hoff, W. A., and Vincent, T. (2002). Three-dimensional motion and structure estimation using inertial sensors and computer vision for augmented reality. *Presence*, 11(5):474–492.

[Chausse et al., 2005] Chausse, F., Laneurit, J., and Chapuis, R. (2005). Vehicle localization on a digital map using particles filtering. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2005:243–248.

[Chen et al., 2009] Chen, X., Kohlmeyer, B., Stroila, M., Alwar, N., Wang, R., and Bach, J. (2009). Next generation map making. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*, (Ladybug 3):488.

[Chung et al., 2001] Chung, H., Ojeda, L., and Borenstein, J. (2001). Accurate mobile robot dead-rockoning with a precision-calibrated fiber-optic gyroscope. *IEEE Transactions on Robotics and Automation*, 17(1):80–84.

[Darpa, 2007] Darpa (2007). Urban Challenge Route Network Definition File (RNDF) and. *Network*.

[De Berg et al., 2008] De Berg, M., Cheong, O., Van Kreveld, M., and Overmars, M. (2008). *Computational geometry: Algorithms and applications*.

[Dellaert, 2012] Dellaert, F. (2012). Factor graphs and GTSAM: A hands-on introduction. (September):1–27.

[Deschaud, 2018] Deschaud, J.-E. (2018). IMLS-SLAM: scan-to-model matching based on 3D data. *Journal of Chemical Physics*, 148(13).

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–108.

[EDMap, 2004] EDMap (2004). Enhanced Digital Mapping Project Final Report.

[Engelson and McDermott, 1992] Engelson, S. P. and McDermott, D. V. (1992). Error correction in mobile robot map learning. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 3, pages 2555–2560. Publ by IEEE.

[Fischler and Bolles, 1981] Fischler, M. a. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Fouque et al., 2008] Fouque, C., Bonnifait, P., and Betaille, D. (2008). Enhancement of global vehicle localization using navigable road maps and dead-reckoning. *Record - IEEE PLANS, Position Location and Navigation Symposium*, pages 1286–1291.

[Fox et al., 1999] Fox, D., Burgard, W., and Thrun, S. (1999). Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427.

[Fu et al., 2016] Fu, H., Ye, L., Yu, R., and Wu, T. (2016). An efficient scan-to-map matching approach for autonomous driving. In *2016 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2016*, pages 1649–1654. IEEE.

[Fuentes-Pacheco et al.,] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, (1):55–81.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[Gil Jiménez et al., 2008] Gil Jiménez, P., Bascón, S. M., Moreno, H. G., Arroyo, S. L., and Ferreras, F. L. (2008). Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies. *Signal Processing*, 88(12):2943–2955.

[Gomes-Mota and Ribeiro, 2000] Gomes-Mota, J. and Ribeiro, M. I. (2000). Mobile robot localisation on reconstructed 3D models. *Robotics and Autonomous Systems*, 31(1):17–30.

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

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6.

[Harris and Pike, 1987] Harris, C. G. and Pike, J. M. (1987). 3D Positional Integration from Image Sequences. *Procedings of the Alvey Vision Conference 1987*, pages 32.1–32.4.

[Hata and Wolf, 2014] Hata, A. and Wolf, D. (2014). Road marking detection using LIDAR reflective intensity data and its application to vehicle localization. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 584–589.

[He et al., 2016] He, B., Ai, R., Yan, Y., and Lang, X. (2016). Lane marking detection based on convolution neural network from point clouds. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 2475–2480. IEEE.

[Héry et al., 2018] Héry, E., Masi, S., Xu, P., and Bonnifait, P. (2018). Map-based curvilinear coordinates for autonomous vehicles. *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, 2018-March:1–7.

[Hong et al., 2010] Hong, S., Ko, H., and Kim, J. (2010). VICP: Velocity updating iterative closest point algorithm. *Proceedings - IEEE International Conference on Robotics and Automation*, (Section 3):1893–1898.

[Houben, 2011] Houben, S. (2011). A single target voting scheme for traffic sign detection. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 124–129. IEEE.

[Huang and Stachniss, 2018] Huang, K. and Stachniss, C. (2018). Joint Ego-motion Estimation Using a Laser Scanner and a Monocular Camera Through Relative Orientation Estimation and 1-DoF ICP. *IEEE Int. Conf. Intell. Robot. Syst.*, pages 671–677.

[Huang et al., 2019] Huang, K., Xiao, J., and Stachniss, C. (2019). Accurate Direct Visual-Laser Odometry with Explicit Occlusion Handling and Plane Detection. In *2019 Int. Conf. Robot. Autom.*, pages 1295–1301. IEEE.

[IEEE Robotics and Automation Society, 2015] IEEE Robotics and Automation Society (2015). *1873-2015 IEEE Standard for Robot Map Data Representation for Navigation*.

[Javanmardi and Javanmardi, 2017] Javanmardi, E. and Javanmardi, M. (2017). Autonomous Vehicle Self-Localization Based on Probabilistic Planar Surface Map and Multi-channel LiDAR in Urban Area. pages 157–164.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. page 182.

[Kaess et al., 2008] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Trans. Robot.*, 24(6):1365–1378.

[Kalman, 1960] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(Series D):35–45.

[Kammel and Pitzer, 2008] Kammel, S. and Pitzer, B. (2008). Lidar-based lane marker detection and mapping. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 1137–1142.

[Konolige et al., 2007] Konolige, K., Agrawal, M., and Solà, J. (2007). Large Scale Visual Odometry for Rough Terrain. *Proceedings of the International Symposium on Robotics Research ({ISRR})*.

[Kuipers and Byun, 1991] Kuipers, B. and Byun, Y. T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47–63.

[Kummerle et al., 2011] Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G2o: A general framework for graph optimization. In *2011 IEEE Int. Conf. Robot. Autom.*, volume 30, pages 3607–3613. IEEE.

[Lacroix et al., 1999] Lacroix, S., Mallet, A., Chatila, R., Gallo, L., and B, E. S. C. S. V. R. (1999). ROVER SELF LOCALIZATION IN PLANETARY-LIKE ENVIRONMENTS A tentative classification of exte- roceptive localization techniques 3 Motion estimation using stereovi- sion and pixel tracking. (June).

[Lefaudeux and Nashashibi, 2012] Lefaudeux, B. and Nashashibi, F. (2012). Real-time visual perception : detection and localisation of static and moving objects from a moving stereo rig. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 522–527.

[Leutenegger et al., 2015] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.

[Levinson et al., 2008] Levinson, J., Montemerlo, M., and Thrun, S. (2008). Map-Based Precision Vehicle Localization in Urban Environments. *Robotics: Science and Systems III*, pages 121–128.

[Levinson and Thrun, 2010] Levinson, J. and Thrun, S. (2010). Robust vehicle localization in urban environments using probabilistic maps. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4372–4378.

[Li et al., 2010] Li, H., Nashashibi, F., and Toulminet, G. (2010). Localization for intelligent vehicle by fusing mono-camera, low-cost GPS and map data. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 1657–1662. IEEE.

[Lombardi et al., 2005] Lombardi, P., Zanin, M., and Messelodi, S. (2005). Unified stereovision for ground, road, and obstacle detection. In *IEEE Intelligent Vehicles Symposium, Proceedings*, volume 2005, pages 783–788. IEEE.

[Longuet, 1981] Longuet (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(September).

[Low, 2004] Low, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110.

[Lu and Song, 2015] Lu, Y. and Song, D. (2015). Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints. *IEEE Transactions on Robotics*, 31(3):736–749.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). Iterative Image Registration Technique With an Application To Stereo Vision. Technical report.

[Magnusson et al., 2007] Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827.

[Manjunath et al., 1992] Manjunath, B. S., Chellappa, R., and Von Der Malsburg, C. (1992). A feature based approach to face recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1992-June, pages 373–378. IEEE Comput. Soc. Press.

[Martin Ester, Hans-Peter Kriegel, Jörg Sander, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, X. X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise.

[Matthies and Shafer, 1987] Matthies, L. and Shafer, S. A. (1987). Error Modeling in Stereo Navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248.

[McCall and Trivedi, 2006] McCall, J. C. and Trivedi, M. M. (2006). Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):20–37.

[Møgelmose et al., 2012] Møgelmose, A., Trivedi, M. M., and Moeslund, T. B. (2012). Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497.

[Moosmann and Stiller, 2011a] Moosmann, F. and Stiller, C. (2011a). Velodyne SLAM. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 393–398. IEEE.

[Moosmann and Stiller, 2011b] Moosmann, F. and Stiller, C. (2011b). Velodyne SLAM. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 393–398. IEEE.

[Moravec, 1977] Moravec, H. (1977). Towards automatic visual obstacle avoidance. *International Joint Conference on Artificial Intelligence*, page 2.

[Moravec, 1980] Moravec, H. P. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover.

[Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. Institute of Electrical and Electronics Engineers.

[Mouragnon et al., 2006] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real Time Localization and 3D Reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 363–370. IEEE.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M., and Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[NHTSA, 2016] NHTSA (2016). TRAFFIC SAFETY FACTS 2016 Fatal Motor Vehicle Crashes: Overview. *Annu. Rev. Clin. Psychol.*, (October):1–9.

[NHTSA, 2018] NHTSA (2018). Overview of the 2018 Crash Investigation Sampling System. Technical report.

[Nist, 2003] Nist, D. (2003). Preemptive RANSAC for Live Structure and Motion Estimation Corresponding Author : Preemptive RANSAC for Live Structure and Motion Estimation. *Machine Vision and Applications*, 16(Iccv):1–29.

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, 1:652–659.

[Obradovic et al., 2007] Obradovic, D., Lenz, H., and Schupfner, M. (2007). Fusion of sensor data in Siemens car navigation system. *IEEE Transactions on Vehicular Technology*, 56(1):43–50.

[Otsu, 1979] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Pandey et al., 2011] Pandey, G., McBride, J. R., Savarese, S., and Eustice, R. M. (2011). Visually bootstrapped generalized ICP. *Proc. - IEEE Int. Conf. Robot. Autom.*, pages 2660–2667.

[Park et al., 1998] Park, K., Chung, H., and Lee, J. G. (1998). Dead Reckoning Navigation for Autonomous Mobile Robots. *IFAC Proceedings Volumes*, 31(3):219–224.

[Pink, 2008] Pink, O. (2008). Visual map matching and localization using a global feature map. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, pages 1–7.

[Polok et al., 2013] Polok, L., Solony, M., Smrz, P., Ila, V., and Zemcik, P. (2013). Incremental cholesky factorization for least squares problems in robotics? *IFAC Proc. Vol.*, 8(PART 1):172–178.

[Pradeep et al., 2008] Pradeep, V., Medioni, G., and Weiland, J. (2008). Piecewise Planar Modeling for Step Detection using Stereo Vision. *Workshop on Computer Vision Applications for The Visually Impaired*, pages 1–8.

[Prisacariu et al., 2010] Prisacariu, V. A., Timofte, R., Zimmermann, K., Reid, I., and Van Gool, L. (2010). Integrating object detection with 3D tracking towards a better driver assistance system. In *Proceedings - International Conference on Pattern Recognition*, pages 3344–3347. IEEE.

[Qin et al., 2018] Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

[R. Timofte, K. Zimmermann, 2009] R. Timofte, K. Zimmermann, L. V. G. (2009). Multi-view traffic sign detection, recognition, and 3D localisation. In *Machine Vision and Applications*, pages 1–8. IEEE.

[Ranganathan et al., 2013] Ranganathan, A., Ilstrup, D., and Wu, T. (2013). Lightweight localization for vehicles using road markings. In *IEEE International Conference on Intelligent Robots and Systems*, pages 921–927. IEEE.

[Rasmussen and Korah, 2006] Rasmussen, C. and Korah, T. (2006). On-Vehicle and Aerial Texture Analysis for Vision-Based Desert Road Following. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, volume 3, pages 66–66. IEEE.

[Ren et al., 2009] Ren, F. X., Huang, J., Jiang, R., and Klette, R. (2009). General traffic sign recognition by feature matching. In *2009 24th International Conference Image and Vision Computing New Zealand, IVCNZ 2009 - Conference Proceedings*, pages 409–414. IEEE.

[Ribas et al., 2008] Ribas, D., Ridao, P., Tardós, J. D., and Neira, J. (2008). Underwater SLAM in man-made structured environments. *Journal of Field Robotics*, 25(11-12):898–921.

[Richard and Andrew, 2004] Richard and Andrew (2004). *Multiple view geometry in computer vision*.

[Riveiro et al., 2016] Riveiro, B., Diaz-Vilarino, L., Conde-Carnero, B., Soilan, M., and Arias, P. (2016). Automatic Segmentation and Shape-Based Classification of Retro-Reflective Traffic Signs from Mobile LiDAR Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(1):295–303.

[Roumeliotis et al., 2002] Roumeliotis, S. I., Johnson, A. E., and Montgomery, J. F. (2002). Augmenting inertial navigation with image-based motion estimation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 4326–4333 vol.4.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571.

[Rusinkiewicz and Levoy, 2001] Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, 2001-Janua:145–152.

[Ruta et al., 2011] Ruta, A., Porikli, F., Watanabe, S., and Li, Y. (2011). In-vehicle camera traffic sign detection and recognition. *Machine Vision and Applications*, 22(2):359–375.

[Sarvaiya et al., 2009] Sarvaiya, J. N., Patnaik, S., and Bombaywala, S. (2009). Image registration by template matching using normalized cross-correlation. In *ACT 2009 - International Conference on Advances in Computing, Control and Telecommunication Technologies*, pages 819–822. IEEE.

[Scaramuzza and Fraundorfer, 2011] Scaramuzza, D. and Fraundorfer, F. (2011). Tutorial: Visual odometry. *IEEE Robotics and Automation Magazine*, 18(4):80–92.

[Scaramuzza et al., 2009] Scaramuzza, D., Fraundorfer, F., and Siegwart, R. (2009). Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *2009 IEEE International Conference on Robotics and Automation*, pages 4293–4299. IEEE.

[Schultz and Adams, 1998] Schultz, A. C. and Adams, W. (1998). Continuous localization using evidence grids. *Proceedings - IEEE International Conference on Robotics and Automation*, 4(May):2833–2839.

[Schuster et al., 2016] Schuster, F., Keller, C. G., Rapp, M., Haueis, M., and Curio, C. (2016). Landmark based radar SLAM using graph optimization. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 2559–2564.

[Se et al., 2005] Se, S., Lowe, D. G., and Little, J. J. (2005). Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375.

[Skog and Handel, 2009] Skog, I. and Handel, P. (2009). In-car positioning and navigation technologiesa survey. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21.

[Smith and Cheeseman, 1986] Smith, R. C. and Cheeseman, P. (1986). On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.

[S.Suzuki, 1985] S.Suzuki, K. (1985). Topological structural analysis of digital binary image by border following. *Cvgip*, pages 32–46.

[Strasdat et al., 2010] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Real-time monocular SLAM: Why filter? *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2657–2664.

[Suganuma and Uozumi, 2011a] Suganuma, N. and Uozumi, T. (2011a). Precise position estimation of autonomous vehicle based on map-matching. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):296–301.

[Suganuma and Uozumi, 2011b] Suganuma, N. and Uozumi, T. (2011b). Precise position estimation of autonomous vehicle based on map-matching. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):296–301.

[Taketomi et al., 2017] Taketomi, T., Uchiyama, H., and Ikeda, S. (2017). Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1).

[Tardif et al., 2008] Tardif, J.-P., Pavlidis, Y., and Daniilidis, K. (2008). Monocular visual odometry in urban environments using an omnidirectional camera. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–26.

[Tgri,] Tgri. Essentials of Satellite Navigation.

[Thrun, 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

[Thrun, 2002a] Thrun, S. (2002a). Probabilistic robotics. *Communications of the ACM*, 45(3):480.

[Thrun, 2002b] Thrun, S. (2002b). Robotic Mapping: A Survey. *Science*, 298(February):1–35.

[Toledo-Moreo et al., 2009] Toledo-Moreo, R., Bétaille, D., Peyret, F., and Laneurit, J. (2009). Fusing GNSS, dead-reckoning, and enhanced maps for road vehicle lane-level navigation. *IEEE Journal on Selected Topics in Signal Processing*, 3(5):798–809.

[Tong and Barfoot, 2013] Tong, C. H. and Barfoot, T. D. (2013). Gaussian Process Gauss-Newton for 3D laser-based Visual Odometry. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5204–5211.

[Triggs et al., 2000] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle adjustment – a modern synthesis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1883:298–372.

[Vázquez-Reina et al., 2005] Vázquez-Reina, A., Lafuente-Arroyo, S., Siegmann, P., Maldonado-Bascon, S., and Acevedo-Rodríguez, J. (2005). Traffic sign shape classification based on correlation techniques.

[Vitor et al., 2014] Vitor, G. B., Victorino, A. C., and Ferreira, J. V. (2014). Comprehensive performance analysis of road detection algorithms using the common urban Kitti-road benchmark. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 19–24. IEEE.

[Vu et al., 2013] Vu, A., Yang, Q., Farrell, J. A., and Barth, M. (2013). Traffic sign detection, state estimation, and identification using onboard sensors. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 875–880. IEEE.

[Wang et al., 2017] Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017). Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050.

[Wijk and Christensen, 2000] Wijk, O. and Christensen, H. I. (2000). Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data. *Robotics and Autonomous Systems*, 31(1):31–42.

[Wolcott and Eustice, 2016] Wolcott, R. W. and Eustice, R. M. (2016). Visual Localization within LIDAR Maps for Automated Urban Driving.

[Zhang and Singh, 2014a] Zhang, J. and Singh, S. (2014a). LOAM : Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems*.

[Zhang and Singh, 2014b] Zhang, J. and Singh, S. (2014b). LOAM : Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems*.

[Zhang and Singh, 2015] Zhang, J. and Singh, S. (2015). Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast. *IEEE International Conference on Robotics and Automation*, pages 2174–2181.

[Zhang and Singh, 2017] Zhang, J. and Singh, S. (2017). Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416.

[Zhou and Deng, 2014] Zhou, L. and Deng, Z. (2014). LIDAR and Vision-Based Real-Time Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicle. In *17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 578–583. IEEE.

RÉSUMÉ

Dans le cadre de cette thèse, un système de perception à base d'un capteur LIDAR et un système de localisation sur une carte numérique très précise ont été développés dans le contexte des développements des véhicules autonomes. Le système de perception proposé utilise les données 3D augmentées par la réflectivié du LiDAR afin de détecter les marquages au sol, les barrières, les panneaux de signalisation et les rétro-reflecteurs placés sur les barrières ou rails de sécurité dans un environnement autoroutier. Les objets détectés sont ensuite recalés par rapport à une carte numérique très précise. Cette dernière contient les lignes de marquage dans un format spécifique, les panneaux de signalisation ainsi que d'autres attributs sémantiques. Le recalage est assuré via une implémentation d'un filtre particulaire auquel nous avons effecuté des améliorations pour optimiser la distribution des particules sans pour autant en modifier le nombre. Cette méthode est appelée: la mise à jour contrainte du filtre. Pour évaluer la méthode proposée, nous avons utilisé un système de navigation satellitaire (GNSS) avec correction RTK comme une vérité terrain et nous avons adopté différentes métriques pour montrer la précision de notre système. Les expériences ont été menées sur deux autoroutes: une piste de test propre à Renault et un tronçon d'environ 50 kilomètres sur sur route ouverte. Les résultats sont prometteurs et montrent la faisabilité d'un système de localisation fondé sur des LiDARs seuls et avec une réprésentation éparse des données (sous forme d'amers plutôt que la totalité du nuage de points)

MOTS CLÉS

Véhicule autonome, Perception LiDAR, localisation précise sur carte

ABSTRACT

In this thesis, we address the problem of accurate localization of autonomous vehicles on highway roads using LiDAR sensors and a highly accurate third party map. The proposed approach is based on two core modules: perception and map-matching. The perception module uses the 3D data enhanced by the LiDAR reflectivity to detect road primitive features: lane markings, barriers, traffic signs and guardrail reflectors. The map-matching module incorporates these measurements and aligns them against a highly accurate third party map. The map-matching is performed using a particle filter, which we have improved in order to deal with the particle deprivation problem. The proposed improvement uses the road geometry in order to optimize the spatial distribution of particles while maintaining the number of particles constant. To evaluate the proposed method, we compared the localization outputs of our system to a Global Navigation Satellite System (GNSS) with RTK corrections (ground truth). Experiments have been conducted on two highway roads. The first is an experimental test track (CTA2) of 5 km long located at CTA, Renault's Aubevoye's Technical Center. This track is designed to exactly replicate a two-lane highway environment. The second is a section of the A13 highway, running from Paris and ending at Aubevoye. The results are promising and show the feasibility of a localization system based on LiDARs alone and with a sparse map data representation

KEYWORDS

Autonomous vehicles, LiDAR perception, Precise Map-based Localization.