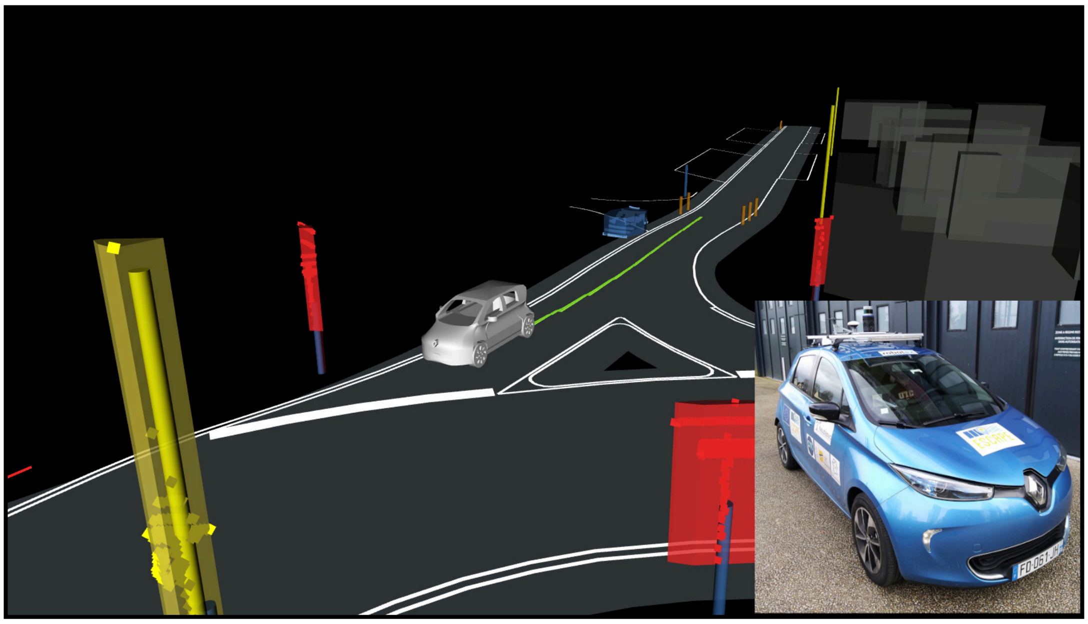
Spatio-temporal data fusion for intelligent vehicle localization

Anthony Welte

Par **Anthony WELTE**

Spatio-temporal data fusion for intelligent vehicle localization

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 11 décembre 2020
Spécialité : Automatique et Robotique : Unité de recherche Heudyasic (UMR-7253)

D2572

Spatio-temporal data fusion for intelligent vehicle localization

Anthony Welte

PhD thesis prepared in the
Heudiasyc Laboratory, UMR UTC/CNRS 7253

Defended on the Eleventh of December, 2020

Spécialité : Automatique et Robotique

Thesis Committee:

Reviewers:	Silvère Bonnabel	Université de la Nouvelle-Calédonie
	Hélène Piet Lahanier	ONERA
Examiners:	Véronique Berge Cherfaoui	Univ. de Technologie de Compiègne
	Luc Jaulin	ENSTA Bretagne
	Lyudmila Mihaylova	The University of Sheffield
Industrial Advisor	Clément Zinoune	Renault S.A.
Supervisors:	Philippe Bonnifait	Univ. de Technologie de Compiègne
	Philippe Xu	Univ. de Technologie de Compiègne

The doctoral grant relative to the project Fustinav is part-funded by the European Regional Development Fund (ERDF).

Acknowledgements

This thesis would not have been possible without the help and support from several people.

First, I would like to thank my thesis supervisors Philippe Xu and Philippe Bonnifait for their experience and guidance during the last three years. Our talks have greatly contributed to this thesis.

I thank Clément Zinoune for sharing his experience, in particular concerning HD maps. His advice and guidance enabled this work to address important problems in the industry.

I am also grateful for the thesis committee members: Silvère Bonnabel, Hélène Piet-Lahanier, Véronique Berge Cherfaoui, Luc Jaulin and Lyudmila Mihaylova, who have shown interest in my work. I am especially thankful to the two reviewers Silvère and Hélène whose comments and suggestions have greatly contributed to the finalization of this manuscript.

I am also grateful for the other members of the Heudiasyc laboratory with whom I have worked and shared ideas: Antoine, Corentin, Edouard, Elwan, Federico, Joelle, and Stefano. They have all contributed to making the last three years enjoyable.

I am also grateful for Stephane and Thierry whose help has been essential to record the datasets used in this work.

Finally, I am also thankful for Jonathan and Gabriel with whom I have been confined during the first phase of lockdown that France went through in 2020. They were essential to keep my sanity intact during the strange times that we have lived.

Abstract

Developing autonomous vehicles raises many challenging issues. Localization is an essential basic capability for vehicles to be able to navigate autonomously on the road. It is critical for safety reasons that the localization system provides an accurate and reliable localization. This can be achieved through already available sensors (e.g. yaw rate gyro, wheel encoders, GNSS receivers) used for ADAS and basic navigation, and new technologies (e.g. lidars, smart cameras) that provide valuable information on the vehicle surroundings. These sensors combined with highly accurate maps result in greater accuracy for localization. Localization consists of estimating the vehicle state based on previously obtained measurements either in a sequential manner or by batches. In this work, the benefits of storing and reusing information in memory (in data buffers) are explored.

Localization systems need to perform several tasks like high-frequency estimation, map matching, calibration and error detection. These tasks do not need to be performed at the same time and are not equally expensive to run. To address all issues, a generic framework composed of several processing layers is proposed and studied. A main filtering layer estimates the vehicle pose while other layers address the more complex problems. In particular, a layer running in parallel performs map matching of detected features and a post-processing layer does calibration and map error detection.

High-frequency state estimation relies on proprioceptive measurements combined with GNSS observations. These measurements can not only be delayed or arrive out of sequence, but also be affected by biases due to the limitations of the sensors. Calibration is therefore essential to obtain an accurate pose. By keeping state estimates and observations in a buffer, the observation models of these sensors can be calibrated. This is achieved using smoothed estimates in place of a ground truth as it would not be available in commercial vehicles.

Dead-reckoning and GNSS can provide a high frequency estimation, but are not accurate enough for autonomous driving applications. Lidars and smart cameras provide measurements of features of the environment that can be used for localization when combined with prior knowledge (stored in a map) of their position. Using such observations raises matching issues as correct matches with the map features have to be obtained. Matching can be difficult due to state estimation errors and the limited number of measurements available in real time. Moreover, it needs to be robust to missing features and sensors miss detection. In this work, the matching problem is addressed on a spatio-temporal window. Thus, matching is done with a buffer of observations, resulting in a more detailed picture of the environment. To limit the effect of state errors on matching, the state buffer is adjusted using the observations and all possible matches.

Although using mapped features for localization enables to reach greater accuracy, this is only true if the map can be trusted. Road networks do not appear to change often, but a small change can have disastrous effects on localization. Roads repainted slightly off and road signs moving by a few decimeters either involuntarily (e.g. accidents) or on purpose (e.g. modification of an intersection) are very common. An approach using the post smoothing residuals has been studied to detect changes and either mitigate or reject the affected features.

The thesis relies heavily on experimental data acquired with vehicles equipped with lidar sensors and smart cameras. High Definition (HD) maps have also been used in the framework of the SIVALab joint laboratory between Renault and Heudiasyc CNRS-UTC.

Résumé

Le développement des véhicules autonomes soulève de nombreux problèmes. La localisation constitue une brique essentielle permettant aux véhicules de naviguer de manière autonome sur la route. Pour des raisons évidentes de sureté, le système de localisation doit fournir une information précise et fiable. Cela peut être atteint à travers les capteurs déjà existants (par exemple, gyros, roues codeuses, récepteurs GNSS) utilisés pour les systèmes de navigation et les ADAS; et de nouvelles technologies (par exemple, lidars, caméras intelligentes) qui fournissent des informations riches sur l'environnement avoisinant du véhicule. Ces capteurs combinés avec des cartes routières haute définition permettent d'atteindre un plus haut degré de précision. La localisation consiste à estimer l'état du véhicule en utilisant les mesures passées soit de manière séquentielle soit de manière groupée. Dans ce travail, l'intérêt d'enregistrer et réutiliser des informations sauvegardées en mémoire est exploré.

Les systèmes de localisation doivent réaliser plusieurs tâches comme une estimation à haute fréquence, des associations de données, de la calibration et de la détection d'erreurs. Ces tâches n'ont pas besoin d'être effectuées au même moment et n'ont pas le même coût en temps de calcul. Pour résoudre ces problèmes, une architecture générique composée de plusieurs couches de traitement est proposée et étudiée. Une couche principale de filtrage estime la pose du véhicule tandis que les autres abordent les problèmes plus complexes. En particulier, une couche réalise en parallèle la tâche d'association des éléments détectés avec les amers d'une carte haute définition et une étape de post-traitement traite la calibration du système et la détection des erreurs de carte.

L'estimation d'état haute fréquence repose sur des mesures proprioceptives combinées à des observations GNSS. Ces mesures peuvent non seulement avoir du délai ou être arrivées dans le désordre, mais aussi être affectées par des biais de mesure dus aux limitations des capteurs. La calibration du système est ainsi essentielle afin d'obtenir une pose précise. En gardant les états estimés et les observations en mémoire, les modèles d'observation des capteurs peuvent être calibrés. Pour cela, des estimations lissées sont utilisées à la place des vérités terrain, qui ne peuvent pas être disponibles sur les véhicules du commerce.

La navigation à l'estime et les systèmes GNSS permettent d'avoir une estimation à haute fréquence, mais ne sont pas suffisamment précis pour les applications de conduite autonome. Les lidars et les caméras intelligentes fournissent des mesures d'éléments de l'environnement qui peuvent être utilisés pour la localisation lorsqu'ils sont combinés avec des informations a priori (contenues dans une carte) sur leur position. Utiliser de telles observations nécessite de devoir les associer à la carte. Cette association peut être difficile à cause des erreurs d'estimation et du nombre limité de mesures disponibles en temps réel. Dans cette thèse, le problème d'association est abordé à travers une fenêtre spatio-temporelle. Ainsi, l'association est faite avec un *buffer* d'observations, amenant une image plus détaillée de l'environnement. Pour limiter l'effet des erreurs d'état sur l'association, le *buffer* d'états est ajusté avec les observations et toutes les associations possibles.

Bien que l'utilisation d'amers cartographiés permette d'améliorer la localisation, cela n'est possible que si la carte est fiable. Le réseau routier ne change pas souvent, mais de petites modifications peuvent avoir des effets désastreux sur la localisation. Des routes repeintes de manière légèrement décalée et des panneaux routiers qui ont été bougés de quelques dizaines de centimètres soit involontairement (par accident) ou volontairement (par exemple, modification d'une intersection) sont relativement fréquentes. Une approche utilisant des résidus lissés a posteriori est étudiée pour détecter ces changements de carte et affaiblir et rejeter ces éléments pour les passages futurs.

Cette thèse repose grandement sur des données expérimentales acquises avec des véhicules équipés de capteurs lidars et de caméras intelligentes. Des cartes haute définition ont également été utilisées dans le cadre du laboratoire commun SIVALab entre Renault et le laboratoire Heudiasyc CNRS-UTC.

Contents

List of Figures

List of Tables

Acronyms

ABS Anti-lock Braking System. 4, 33, 41

ADAS Advanced Driver-Assistant Systems. 1, 4

BFGS Broyden–Fletcher–Goldfarb–Shanno. 97, 108

CAN Controller Area Network. 46, 138

CCDA Combined Constraint Data Association. 132

DR Dead Reckoning. 1, 7, 8, 33, 41, 54, 129, 131

ESC Electronic Stability Control. 4, 33, 41, 47

GLONASS Global Navigation Satellite System. 52

GNSS Global Navigation Satellite System. 2, 3, 5

GPS Global Positioning System. 1, 2, 52

HD map High Definition map. xv, 67, 140, 139, 140, 143

ICP Iterative Closest Point. 3

IMU Inertial Measurement Unit. 4, 10, 39, 59, 137

PCA Principal Component Analysis. 79

ROS Robot Operating System. 99, 107

RTK Real-Time Kinematic. 2, 39, 52, 99, 137

SLAM Simultaneous Localization And Mapping. 3, 17, 21, 64, 112

Symbols

k Identifier of an epoch (time).

j Identifier of an observation (e.g. $\boldsymbol{z}_k^j$ is the j observation at time k).

i Identifier of a map feature (e.g. $\boldsymbol{m}_i$).

K Time of the end of a buffer.

$\boldsymbol{x}_k$ State of the vehicle at epoch k.

$\hat{\boldsymbol{x}}_{k|K}$ Estimation of the vehicle state $\boldsymbol{x}_k$ using observations from time 1 to K.

$\check{\boldsymbol{x}}_k$ Ground Truth state of the vehicle at time k.

$f(\cdot)$ The evolution model.

$\boldsymbol{F}_k$ The Jacobian of the evolution model linearized at time k.

$\boldsymbol{Q}_k$ The covariance matrix of the evolution model noise.

$h(\cdot)$ An observation model.

$\boldsymbol{H}_k$ The Jacobian of the observation model linearized at time k.

$\boldsymbol{R}_k$ The covariance matrix of the observation model noise.

$\boldsymbol{m}_i$ A map feature, can be a lane marking or a road sign.

$\mathcal{R}$ A frame of reference, e.g. $\mathcal{R}_M$ is the vehicle mobile frame.

${}^{M}\emptyset$ Indicate that $\emptyset$ is expressed in the frame $\mathcal{R}_M$.

1. General Introduction

Contents

1.1. Localization for Intelligent Autonomous Vehicles

Performance improvements and size reduction of electronics makes possible embedding complex functions in vehicles. Passenger cars can now help the driver by providing advice such as route guidance or executing tasks such as keeping lane, speed or suddenly break in case of immediate hazard. Those functions are known as Advanced Driver-Assistant Systems (ADAS). Effort is also made to make passenger and public transport vehicles able to achieve complete driving with no human intervention (i.e. autonomous vehicles). To reach this goal, vehicles need to understand where it is in the environment in order to take the convenient actions at any time.

Localization has always been a critical issue for intelligent transportation systems. Even when the system is driven manually, the driver needs to know where the vehicle is to reach the destination. This is not a problem for small distances but as road networks develop, localization systems become an essential part of vehicles. Nowadays, drivers are most familiar with the Global Positioning System (GPS) accessible for navigation from any smartphone and available in most vehicles. But automobile manufacturers have not waited for satellite-based positioning to develop localization systems.

Already in 1966, General Motors imagined a localization system relying on magnets embedded in the road, the Driver-Aid Information and Routing (DAIR) system. The magnets would be placed at regular intervals on highways and main axis enabling vehicles to know which road section they were traveling on. Although the system did not have a map, the driver could be guided using prerecorded routes saved on punch cards. The cost associated with outfitting roads with the systems made it impractical for automobiles. However, similar systems are now used to track trains on rail networks. A system closer to what we now use for navigation came in 1981, with Japanese car manufacturers being able to use dead-reckoning to track the movement of the vehicle thanks to new automotive grade gyrometers. Such systems were able to track the vehicle movement from its starting position by integrating the vehicle displacements. The vehicle position could then be displayed on maps printed on a transparent film. With this system the driver could see the road network and the vehicle position and take decision accordingly to reach a destination. As with all Dead Reckoning (DR) systems, the estimation drifts away from the true position with time and distance. It is therefore not applicable for long travels. The system was later improved by Etak which replaced the transparent physical maps by numeric maps saved on cassette tapes. The improved system also performed map matching to prevent drift for dead-reckoning only localization. The maps and vehicle position were displayed using a cathode-ray tube screen. The Etak system was the earliest example of map-aided localization for commercial automobiles. Through a series of acquisition it ended up part of the now ubiquitous vehicle navigation company TomTom.

Automotive localization took a major leap forward with the completion of the GPS constellation in 1995. Although initially intended for military use, the system had been opened to civil applications a few years earlier. The improvement in accuracy from the removal of *Selective Availability* in 2000, made GPS the ideal localization system for automotive applications. To this day GPS, and more generally Global Navigation Satellite Systems (GNSSs), are the main localization systems for consumer applications. Combined with dead-reckoning and map matching to handle tunnels and areas where GNSSs are inaccurate, it forms the main localization technology used for commercial applications.

Solutions used for navigation to this day do not provide sufficient accuracy to safely drive autonomously. At this point, the driver has always been in control of the vehicle and errors in localization could not result in significant danger. To achieve fully autonomous vehicles, the localization system needs to be accurate and reliable enough to enable the vehicle to drive safely.

The early prototypes of autonomous vehicles used accurate GNSSs. Although conventional GNSSs are not accurate enough, using Real-Time Kinematic (RTK) methods that compensate errors using a static receiver with known position, the accuracy can be greatly improved. Additionally, using expensive fiber optic gyrometers, early autonomous vehicle prototypes were able to accurately position themselves. To complete one of the earliest autonomous driving challenges, the 2005 DARPA Grand Challenge, all successful teams used high accuracy GPS positioning as their main localization solution. Additionally some teams used lidars to build maps for local navigation in satellite denied environments. Because of their cost, such systems are not planned to be available for commercial vehicles. However, they are still a vital component of experimental vehicles as they provide the best localization available. They are used as *ground truth* to evaluate cheaper localization solutions.

To develop localization systems not relying on expensive high accuracy GNSS solutions, the use of maps has shown some success [Tsuchiya et al., 2019, Wang et al., 2020, Vivacqua et al., 2018]. As creating maps can be itself costly, localization with maps has often been studied in the context of Simultaneous Localization And Mapping (SLAM). SLAM strategies have enabled robots with perception sensors to localize themselves without relying on global positioning systems. These systems build a local map of the observed environment around the robot and use this map to localize it. Cameras and lidars are, in this approach, the localization sensors. SLAM was initially introduced thanks to the seminal work by [Smith and Cheeseman, 1986] and [Durrant-Whyte, 1988] that set the foundation of the modeling of landmark uncertainties. Initially, these techniques built feature maps of specific features detected using perception sensors which were then used to estimate the vehicle state. The problem was initially addressed using Kalman filtering techniques by considering landmarks as part of the state of the system to be estimated. Later FastSLAM [Montemerlo, 2003] was developed using a particle filtering strategy which better modeled non-Gaussian error distributions and increased the robustness of SLAM to wrong data associations. FastSLAM has been used with both feature maps and dense maps (e.g. occupancy grid maps, evidential maps). More recently SLAM strategies using multi-layer lidar sensors have been developed. Scan matching techniques are used to estimate the relative displacement between two scans. The maps are built by accumulating the scan into a single point cloud. In such maps the vehicle location is found by fitting observed points to a map point cloud using techniques such as Iterative Closest Point (ICP). Using point cloud maps is computer intensive because of the amount of data to go through. Methods approximating the map by a series of Gaussian distribution have been proposed [Magnusson, 2013, Wolcott and Eustice, 2015]. Because of the high level of detail contained in point cloud maps, the localization accuracy that can be reached is suitable for autonomous driving applications. However, using such maps at large scales create problems as those maps are heavy. Feature-based SLAM methods usually consist of a feature detection system (front-end) and an estimation system using the detection to build the map and localize the robot (back-end). Methods to solve the SLAM problem initially relied on Kalman filter estimation as a back-end [Leonard and Durrant-Whyte, 1991]. More recently, global optimization strategies have become more widely used thanks to new efficient graph representation of the estimation [Kaess et al., 2012].

Recently, new types of maps have started being used for autonomous driving applications: High Definition maps. While traditional maps built using SLAM can be very detailed and are locally accurate, they can be affected by drifts if no outside localization system is used. High Definition maps exist in different formats but share characteristics such as global accuracy and contain much more information than traditional maps. Building such maps either relies on SLAM with added anchoring points used to guarantee the global accuracy of the map, or on high accuracy localization systems. Two main types of HD maps are distinguished: dense maps, containing point clouds or raster images, and features maps, containing specific interesting objects. Even when the final map is composed of features, the building process usually involves building a dense map and extracting features from it. Although at this stage HD maps are usually built directly by the user, third party providers are starting to offer such maps. Historic map providers such as TomTom and Here already offer commercial HD map services, mainly for highways. New providers are also appearing. The Nvidia Drive platform enables users to use third-party maps while also enabling HD maps to be built and updated by the user.

To achieve accurate localization, autonomous vehicles are equipped with several types of sensors. All vehicles sold are already equipped with sensors providing information about the vehicle kinematics since they are required for mandatory systems. Inertial Measurement Units (IMUs) equip vehicles to provide a source of information unaffected by skidding, while wheel encoders provide the actual wheel rotation. ADAS use these sensors along with steering wheel angle measurements and accelerometers to detect and correct when the vehicle is not performing the driver's intent. Electronic Stability Control (ESC) prevents the vehicle from under and over steering. Anti-lock Braking System (ABS) Systems prevent wheels to lock when the driver brakes hard. Increasingly vehicles are also being equipped with a camera. Using this sensor, measurements of distances to lane-markings can be obtained, enabling car manufacturers to provide Lane Departure Warning (LDW) system and more recently lane keeping systems. Lidars have not yet reached prices acceptable to be installed on commercial vehicles (although some high-end models are equipped) but have been used extensively in research applications. Lidars came down in price significantly in the last few years. Therefore, it is probable that the first autonomous vehicles would be equipped with lidars.

1.2. Motivation & Problem Statement

Localization for intelligent vehicles is challenging because it needs to address several problems that can affect the quality of the estimation. Moreover, solution to these problems are not necessarily mutually compatible.

Localization relies on input measurements to produce a localization solution. After the raw measurement is obtained by the sensor, it usually requires some processing and can then be transmitted to the localization process. Each step takes time and leads to delays that need to be accounted for before producing an estimate. The localization process itself can also introduce processing delays. Measurement delays will result in measurements received after the corresponding state has been estimated or will affect the computation of expected observations.

Measurement delays also result in additional problems. As measurement delays can vary from one sensor to another, the sensor observations are not guaranteed to be received or processed in the correct order.

Once measurements are received and the delays are managed, the measurements need to be linked to the vehicle state. This step is not trivial as even simple measurements (e.g. gyro measurement) are affected by errors (biases, scaling factors, etc.). The localization system therefore needs to be able to calibrate the observation models to limit these errors. This is difficult as commercial vehicles do not have an independent high accuracy localization system (ground truth) to perform calibration.

While the aforementioned constraints are not specific to intelligent vehicles, this type of robot leads to additional problems. Maps are a natural tool to improve vehicle localization, however matching measurements to mapped features is challenging in environments where the number of features can be high (cities, intersections, etc.).

Moreover the localization system has to be able to detect and reject errors in the map. The road network changes gradually, thus constant monitoring is required to avoid localization error.

Intelligent vehicles are subject to similar issues as robotic exploration but also differ in a major way. While robotic exploration visits the same location multiple times, vehicle are only expected to go from A to B without looping. Hence, intelligent vehicles will not encounter loop closures and therefore cannot use loop closure techniques to improve localization.

1.3. Objectives

While solving the aforementioned problems, the localization system also needs to satisfy several requirements.

Localization is critical to several components of intelligent vehicles such as control. Therefore, it needs to satisfy temporal constraints. To properly control the vehicle, a high frequency ($\geq$ 50 Hz) estimation is needed. However, high frequency is not enough. An estimate is only useful if it is accurate at the time it is received. Hence, estimation needs to be done with little delay. Estimation delays can have significant effect. In 10 ms, the vehicle travels several decimeters and can rotate by about 0.3 degrees.

Intelligent vehicles require accurate pose estimates. Conventional GNSS receivers might be able to localize a vehicle anywhere on Earth but do not reach the accuracy level required for safe autonomous travel. Although the accuracy requirement depends on driving task, errors need to be less than one meter in general and down to 10 cm can be required. A vehicle needs among others to be able to identify on which lane it is driving which requires good enough accuracy (lane-level accuracy). Defining accuracy requirements for intelligent vehicle is not straight forward as it highly depends on the driving situation and the task to perform. Driving typically requires lateral accuracy to be able to maintain a trajectory within a lane. Longitudinal accuracy requirements are not as strict. Except near intersections or crosswalks, a one meter error in the longitudinal direction is far less risky than the same error in the lateral direction. Also some situations require high longitudinal accuracy (e.g. around crosswalks, to park, etc.), in most cases the vehicle is driving on roads without any obstacle thus the accuracy requirement should not be as strict. This separation of localization requirements in longitudinal and lateral dimensions is in itself flawed as roads are curved and lateral errors can turn into longitudinal errors and vice versa.

The localization system needs not only to be accurate but also reliable. The system consistency is essential as an overconfident system will lead to more risks which cannot be tolerated is safety-critical situations. The system therefore is allowed to only exceed the set consistency bound in extremely rare cases. For instance, a wrong match used in the estimation process can cause the system to become incorrectly localized but confident in the localization (because new observation can only reduce the uncertainty of the estimate). Such occurrences have to be contained to a minimum.

Moreover, the localization system needs to be able to deal with new problems. Often part of the localization problem, e.g. matching or calibration, involve adding computational cost to the state estimation itself. To be able to cope with the high-frequency estimation requirement while still performing all necessary tasks of the localization, the architecture has to enable matching, calibration and other processes to be run without charging the state estimation. It must be generic in the sense that it is not structurally made to solve one problem but rather can be easily expanded to solve any new problem that may arise. This thesis studies in particular those last three problems.

The architecture also needs to be modular. Intelligent vehicles being still largely in development, the sensors and algorithms used are not settled. The localization system needs to be agnostic to specific input to be easily expanded as research within Renault and SIVALab brings new developments.

While most localization systems process input data and then discard it, this work aims at discovering additional uses that data can have by storing it instead of directly processing it. Storing measurements in buffers has obvious uses such as dealing with delays but it can also help in other tasks such as calibration, map matching and error detection.

1.4. Contributions

The main contributions of this work are summarized as follows.

- A generic framework for localization using Kalman filtering is proposed. The framework enables localization while being agnostic to the type of inputs. It is expandable with parallel processes to preform additional tasks required by specific measurements, e.g. matching for map feature measurements. This framework enables high-frequency (50 Hz) localization while accounting for known delays to provide the best possible estimate to the rest of the vehicle systems.
- An odometric model using commercial vehicle standard sensor suite is proposed as well as a strategy to calibrate parameters of the model [Welte et al., 2019a]. The proposed calibration method relying on Kalman smoothing enables parameter tuning with ground truth. With this calibration strategy the odometric model drift is reduced enabling localization to rely more on odometry.
- A matching technique is proposed to match measurements globally over a measurement window [Welte et al., 2019b, Welte et al., 2020]. Using Kalman smoothing and an optimization step, the system can match more measurements while simultaneously reducing the risk of mismatches. This results in greater localization accuracy.
- An error detection and mitigation method using post-smoothing residuals. The observation residuals are either used to compute a confidence factor used to weight observations [Welte et al., 2019b] or fused across multiple trajectories to decide whether or not to eliminate a feature.
- Practical evaluation of the proposed methods. Methods are all tested on real data recorded using an experimental vehicle on public roads. Sensor sets involved in methods' implementation are close to those used in intelligent transportation systems.

1.5. Manuscript Organization

This manuscript starts in Chapter 2 with a presentation of the estimation framework developed in this work. It introduces the general architecture and the interaction between the different layers. The three layers of the current architecture: filtering, matching and post-processing form a sensor agnostic architecture. Those three layers will be then detailed in each following chapter.

First the estimation with a high-frequency filtering is introduced. In particular, Chapter 3 presents the dead-reckoning system and its calibration. DR is essential as it is the only system available in any situation. An accurate DR not only enables localization for emergency situations but also enables to take more time to handle other measurements. Calibration of such systems is required to limit drifts. This is also addressed thanks to a smoothing strategy and a least square estimation.

While DR is essential for short term localization, it is not sufficient for the accuracy requirements of intelligent vehicles. Chapter 4 addresses this issue by introducing map-aided localization with the detection of geo-referenced map features. Using such features require matching which has to be accurate to ensure the localization integrity. A global matching over a measurement window is therefore presented. It uses Kalman smoothing and a trajectory adjustment step to improve matching, leading to a better localization.

The final chapter, Chapter 5, addresses the main issue with using maps: map errors. The environment changing regularly, maps are not always entirely accurate. Detecting, managing and correcting such errors are therefore essential for the map to be useful for localization.

Experimental evaluation of the proposed methods are provided in each chapter.

2. Buffered Estimation Framework

Contents

2.1. Introduction

The localization problem can be formalized as a problem of estimating a state $\boldsymbol{x}_k$ representing the system at the current time k. The states always contain the position of the system and can additionally contain its orientations, velocities, etc. To estimate the state, observations are used. Observations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k$ depend on the state and can thus be used to estimate it. The localization task consists in estimating the state $\boldsymbol{x}_k$ knowing observations $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k$.

This is done by finding the most likely state knowing the observations (by solving Equation 2.1).

$$\hat{\boldsymbol{x}}_k = \arg\max_{\boldsymbol{x}_k} p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\right) \tag{2.1}$$

The method chosen to perform the localization task needs to be tailored to the requirements of the considered system. In the context of autonomous vehicles, several criteria need to be fulfilled for the system to be acceptable. Other subsystems (control, trajectory planning) rely on this estimate to work properly. The localization has to provide a state estimate at high frequencies with minimal delay. The localization has to be accurate enough to enable safe autonomous driving. This is required to drive in environments where obstacles can be close to the road and where the traffic is dense (urban area). Beyond the system accuracy, it also needs to be reliable. The state estimate cannot be believed to be more accurate than it is as this would lead other components to take decision they should not. An estimate on the wrong lane might lead the vehicle control to behave incoherently. If the system is confident to be on a specific lane, it cannot be on an adjacent one. Hence, beyond simply being accurate, the integrity of the system needs to be verified. The system needs to reliably predict its accuracy and fail only in extremely rare instances.

State estimation at high frequency rarely uses significant memory. The observations and states are discarded as they are not used anymore. However, many of the problems mentioned in Section 1.2 can be addressed using some memory. Keeping both states and measurements in buffers has classically been used to address delays but can also benefit other aspects of localization. Matching measurements to map features works best the more information is available. Strategies of post processing could also be applied using buffers to tune parameters and evaluate the map. Hence, in this chapter, an architecture relying on buffers is developed to address the challenges of localization for intelligent vehicles.

2.2. Buffer State of the Art

Localization using buffers of data is not new. Simultaneous Localization and Mapping problems requiring maps to be built use some memory to record states. Buffers have also long been used in filtering strategies to deal with delays.

State estimation using buffers can be separated into two main families. Some estimation frameworks rely on buffers entirely and aim at estimating buffers of states using buffers of measurements. Such global approaches optimize the state estimates iteratively. Other methods extend well known filtering strategies to take advantage of buffers. The estimation is then done sequentially rather than as a batch.

2.2.1. Filtering and Smoothing Estimation Schemes

Sequential estimation schemes are often based on Kalman filtering. Most commercial GNSS receivers rely on it to estimate the vehicle position and velocity from the pseudoranges and doppler measurements. It is also often used in combination with sensors such as Inertial Measurement Unit (IMU) to produce state estimations at high frequency.

Kalman filtering is able to produce a state estimation at relatively low computational cost while simultaneously estimating the uncertainty of the state estimate. This makes Kalman filtering ideal for applications that require high frequency estimation with low delay.

Kalman filtering (and other similar estimation strategies) uses two types of relation linking the unknown states and the observations. First the states evolve in a predictable manner. Using an evolution model $f\left(.\right)$, the state $\boldsymbol{x}_k$ can be predicted using the previous state $\boldsymbol{x}_{k-1}$ as such

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k\right) + \boldsymbol{w}_k \tag{2.2}$$

where $\boldsymbol{w}_k$ is an unknown error often modeled as a Gaussian noise and $\boldsymbol{u}_k$ is the input of the system coming from the control system. In this work the input will be ignored, the localization system is studied on its own without a control system. In the rest of this thesis, $f(\cdot)$ will be referred as only a function of the state.

A state is related to an observation using an observation model $h(.)$. The observation model enables to predict expected observations given the state

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{2.3}$$

Observations are rarely perfect, either the model is somewhat unknown or the measurement itself is noisy. An unknown error $\boldsymbol{v}_k$ therefore exists.

Kalman filtering relies on several assumptions. The evolution and the observation models need to be linear for the estimation to be optimal. Several variations have been proposed to deal with non-linear systems. The Extented Kalman Filter (EKF) solves the problem by linearizing the models. The Unscented Kalman Filter propagates sigma point using the non-linear model instead of propagating the covariance matrix of the state estimate. Another assumption made by Kalman filtering is that the noises of the system can be represented by zero-mean uncorrelated Gaussian white noises. This assumption can be challenging as sensors often have internal filters making their outputs temporally correlated. For instance, automotive grade GNSS receivers highly filter pose estimates such that the final estimate already includes information of past measurements. This is useful when the receiver returns the final pose that will be used by the vehicle navigation system for instance. However, it is a problem for localization systems that need to integrate other sensors to the estimation. The pre-filtered GNSS estimations result in time-correlated observations making them non-trivial to use for localization.

Using Equation 2.2 and Equation 2.3, and by modeling each model error by uncorrelated Gaussian White noises with respective covariance matrices $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$, Kalman filtering enables the estimation of the state at each time. This estimation process also assumes that an initial guess $\hat{\boldsymbol{x}}_{0|0}$ (no matter how uncertain) can be obtained with its corresponding covariance matrix $P_{0|0}$.

With these parameters, each state can be estimated sequentially by first predicting the state at time k ($\hat{\boldsymbol{x}}_{k|k-1}$) based on the previously filtered state at $k-1$ ($\hat{\boldsymbol{x}}_{k-1|k-1}$). This is done using the evolution model as described by the following equations,

$$\hat{\boldsymbol{x}}_{k|k-1} = f\left(\hat{\boldsymbol{x}}_{k-1|k-1}\right) \tag{2.4}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k \tag{2.5}$$

where $\boldsymbol{F}_k$ is the Jacobian of the evolution model $f(\cdot)$ linearized around $\hat{\boldsymbol{x}}_{k|k-1}$.

The state estimate $\hat{\boldsymbol{x}}_{k|k-1}$ is the best estimate obtainable without using the observations from time k. If at time k observations are available, they are used to update the estimate using the observation model.

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h_k\left(\hat{\boldsymbol{x}}_{k|k-1}\right) \tag{2.6}$$
$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k \tag{2.7}$$
$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1} \tag{2.8}$$
$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k \tag{2.9}$$
$$\boldsymbol{P}_{k|k} = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k\right) \boldsymbol{P}_{k|k-1} \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k\right)^\top + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^\top \tag{2.10}$$

where $\boldsymbol{H}_k$ is the Jacobian of the observation model $h(\cdot)$ linearized around $\hat{\boldsymbol{x}}_{k|k-1}$.

Different formulations for the Kalman update exist. Here the Joseph form is used as it is more numerically stable than the standard Kalman update formation.

Using this process the states can be estimated iteratively in real time. The covariance matrices $P_{i|j}$ associated with the states $\hat{\boldsymbol{x}}_{i|j}$ are estimated simultaneously.

2.2.1.1. Augmented State

Kalman filtering as stated previously does not, in its standard form, use buffers. It estimates states using available observations and then forget past state estimates and observations.

Kalman filtering has been extended to include a memory of past states [Anderson and Moore, 1979]. Fixed lag smoothing consists of a Kalman filter with the difference being that it uses an augmented state vector containing not only the most recent state information but also the S previous states. Hence in fixed lag smoothing, the estimated state is

$$\hat{\boldsymbol{\mathcal{X}}}_{k|k} = \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k} & \dots & \hat{\boldsymbol{x}}_{k-S|k}\end{bmatrix}^\top \tag{2.11}$$

The evolution model of such an augmented state is trivial, the most recent state evolves using the robot evolution model $f(\cdot)$ and the other states, having their future values already stored in the vector can simply take this value. The augmented evolution model is therefore described by the evolution model Jacobian $\boldsymbol{\mathcal{F}}_k$ and the model noise covariance matrix $\boldsymbol{\mathcal{Q}}_k$ as defined in Equation 2.12 and Equation 2.13.

$$\boldsymbol{\mathcal{F}}_k = \begin{bmatrix} \boldsymbol{F}_k & 0 & \cdots & & 0 \\ \boldsymbol{I} & 0 & \cdots & & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & 0 \\ 0 & \cdots & 0 & \boldsymbol{I} & 0 \end{bmatrix} \tag{2.12}$$

$$\boldsymbol{\mathcal{Q}}_k = \begin{bmatrix} \boldsymbol{Q}_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.13}$$

This way of performing filtering has two consequences, when an observation is added the entire buffer of states stored in the vector are improved (thus the name fixed-lag smoothing). Also observations do not need to be related to the most recent state. Indeed, because previous states are also contained in the estimated vector, an observation can be related to any of them. This enables fixed lag smoothing to easily deal with delayed and out of sequence measurements [Challa et al., 2002] and loop closures [Garcia et al., 2002]. Because all states become correlated with each other during the filtering, an observation on a past state also improves the most recent state. A delayed observation can therefore be used to update the estimation along with other non-delayed observations using the Jacobian of the non-delayed observation model $\boldsymbol{H}_k$ and delayed observation model $\boldsymbol{H}_k^\star$ combined to form the augmented observation matrix in Equation 2.14.

$$\boldsymbol{\mathcal{H}}_k = \begin{bmatrix} \boldsymbol{H}_k & 0 & \cdots & 0 & \boldsymbol{H}_k^\star & 0 & \cdots & 0 \end{bmatrix} \tag{2.14}$$

The main drawback of this approach is its use of resources. The more states are contained in the estimated vector, the larger the size of the involved matrices. All states have to be updated at each iteration even if no observation directly affects the state. Even though Kalman filtering does not invert the state space matrices, the size of the matrix can make the computation costly.

2.2.1.2. Estimation using a Predicted Observation Buffer

Another approach to deal with delayed measurements is to store the predicted observation $h^\star\left(\hat{\boldsymbol{x}}_{k|k-1}\right)$ of the delayed observation in the state vector. Instead of storing the states thus enabling the Kalman update to compute the innovation and correct the appropriate state, this method store the expected observation for a measurement. The state contains the current state $\hat{\boldsymbol{x}}_{k|k}$ and the predicted observation of potentially delayed measurements for each S previous time [Gopalakrishnan et al., 2011].

$$\boldsymbol{\mathcal{X}}_k = \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k} & h^\star\left(\hat{\boldsymbol{x}}_{k|k-1}\right) & \ldots & h^\star\left(\hat{\boldsymbol{x}}_{k-S|k-S-1}\right) \end{bmatrix}^\top \tag{2.15}$$

At each iteration, the current state is used to predict the next one and the most recent expected observation is computed as $h^\star\left(\hat{\boldsymbol{x}}_{k|k-1}\right)$. Equation 2.16 and Equation 2.17 are used to compute the predicted state and covariance matrix using the standard Kalman filtering equations.

$$\boldsymbol{\mathcal{F}}_k = \begin{bmatrix} \boldsymbol{F}_k & 0 & \cdots & 0 \\ \boldsymbol{H}^\star \boldsymbol{F}_k & 0 & \cdots & 0 \\ 0 & \boldsymbol{I} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 \quad \boldsymbol{I} & 0 \end{bmatrix} \tag{2.16}$$

$$\boldsymbol{\mathcal{Q}}_k = \begin{bmatrix} \boldsymbol{Q}_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.17}$$

Observations received on time can be used to update the estimation directly. The delayed observations do not relate to the current state but will relate to an expected observation contained in the augmented state. The observation model for such observation is simply the identity matrix as the predicted observation is already stored in the state. The observation matrix can thus be written as in Equation 2.18, where the first row of the matrix is used to update the state based on observations received without delays, whereas in the second row the identity matrix is placed appropriately depending on the delay of the observation.

$$\boldsymbol{\mathcal{H}}_k = \begin{bmatrix} \boldsymbol{H}_k & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \boldsymbol{I} & 0 & \cdots & 0 \end{bmatrix} \tag{2.18}$$

By comparison with fixed-lag smoothing, this method has the advantage of resulting in a smaller state if the observation dimension is smaller than the state and a single measurement is affected by delays. If multiple measurements are affected by delays, each expected measurement will have to be included in the state in order to be used.

2.2.1.3. Kalman Smoothing

Kalman filtering enables to estimate each state based on the current observations and the previous state (estimated using past observations). Therefore a state at time k has been estimated with the knowledge of all observations from time 1 to k. Kalman filtering only tries to estimate the most recent state. Therefore, if new observations become available, they will be used to estimate future states and not the state at time k. In some cases it can be useful to obtain a complete trajectory $\left\{\hat{\boldsymbol{x}}_{k|N}\right\}_{k\in[\![0,N]\!]}$ from the initialization to the last time N estimated with all the observations $\{\boldsymbol{z}_k\}_{k\in[\![1,N]\!]}$ available. This is a smoothing problem. Extensions of the Kalman filter have been proposed to solve it. Rauch-Tung-Striebel (RTS) smoothing (or Kalman Smoothing) [Rauch et al., 1965] builds upon Kalman filtering to address this problem. While Kalman filtering only performs a forward sequential estimation, Kalman smoothing performs an additional backward sequential smoothing. Once the filtering from epoch 1 to N has been done, the states are smoothed starting from $\hat{\boldsymbol{x}}_{k|k}$ down to

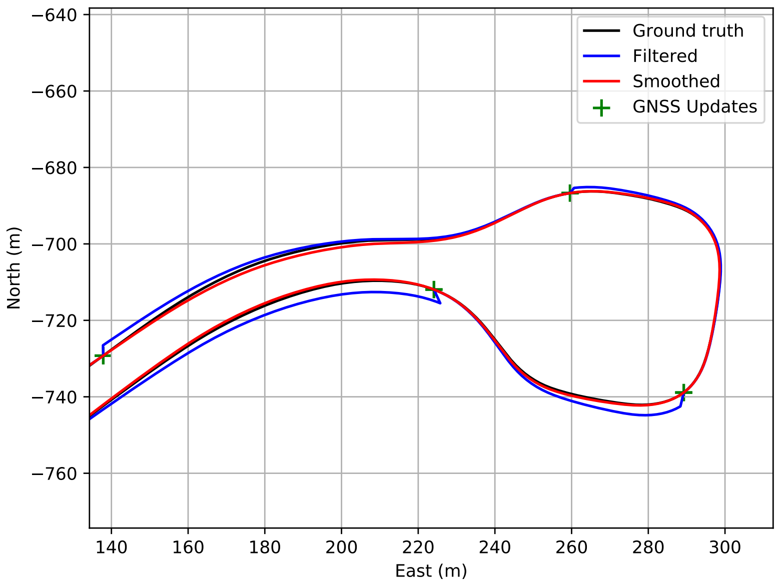


Figure 2.1.: Comparison of the filtered (blue) and smoothed (red) estimates. While the filtered trajectory is corrected at each GNSS updates and then drifts, the smooth trajectory is fully corrected.

state $\hat{\boldsymbol{x}}_{0|0}$. As shown in Figure 2.1, filtering only corrects the real time estimates and does not improve past estimates. Using smoothing, the improved accuracy obtained from future observations is propagated backward leading to better estimates. Since the last state estimate $\hat{\boldsymbol{x}}_{N|N}$ has already been estimated using all observations, it is not improved. However, using Kalman smoothing, all previous state estimates benefit from observations received after their estimation.

To smooth the states, Kalman smoothing uses not only the filtered state estimates $\left\{\hat{\boldsymbol{x}}_{k|k}\right\}_{k\in[\![0,N]\!]}$ but also the predicted states $\left\{\hat{\boldsymbol{x}}_{k|k-1}\right\}_{k\in[\![1,N]\!]}$. After the filtering, when smoothing is performed, these states are used to compute the smoothed states $\left\{\hat{\boldsymbol{x}}_{k|N}\right\}_{k\in[\![0,N]\!]}$ in a backward manner using Equation 2.19 and Equation 2.20. The smoothing process starts with the estimate $\hat{\boldsymbol{x}}_{N|N}$ (the last state obtained with filtering), and improves on the estimation of state $\boldsymbol{x}_{N-1}$ using the predicted state $\hat{\boldsymbol{x}}_{N|N-1}$ and the last smoothed state $\hat{\boldsymbol{x}}_{N|N}$. The process repeats from back to front. At each step the estimation of the state $\boldsymbol{x}_k$ is improved on using the predicted state $\hat{\boldsymbol{x}}_{k+1|k}$ and the smoothed state from the previous step $\hat{\boldsymbol{x}}_{k+1|N}$.

$$\hat{\boldsymbol{x}}_{k|N} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{J}_k \left(\hat{\boldsymbol{x}}_{k+1|N} - \hat{\boldsymbol{x}}_{k+1|k}\right) \tag{2.19}$$

$$\boldsymbol{P}_{k|N} = \boldsymbol{P}_{k|k} + \boldsymbol{J}_k \left(\boldsymbol{P}_{k+1|N} - \boldsymbol{P}_{k+1|k}\right) \boldsymbol{J}_k^\top \tag{2.20}$$

with

Table 2.1.: Equivalences between variables used for Kalman updates and Kalman smoothing

	Kalman update	Kalman smoothing
Updated State	$\hat{\boldsymbol{x}}_{k\|k}$	$\hat{\boldsymbol{x}}_{k\|N}$
Gain	$\boldsymbol{K}_k$	$\boldsymbol{J}_k$
Observation	$\boldsymbol{z}_k$	$\hat{\boldsymbol{x}}_{k+1\|N}$
Observation Model	$\boldsymbol{H}_k$	$\boldsymbol{F}_k$
Observation Covariance	$\boldsymbol{R}_k$	$\boldsymbol{Q}_k$

$$\boldsymbol{J}_k = \boldsymbol{P}_{k|k}\boldsymbol{F}_{k+1}^{\top}\boldsymbol{P}_{k+1|k}^{-1} \tag{2.21}$$

As it can be seen from the previous equation, the smoothing step is similar to the filter itself. In the filtering, the state is corrected based on the difference between observations and predicted observations $\tilde{\boldsymbol{y}}_k$ weighted by the Kalman gain ($\boldsymbol{K}_k$). Here the state is corrected based on the difference between the next state estimate $\hat{\boldsymbol{x}}_{k+1|N}$ and our prediction of that state $\hat{\boldsymbol{x}}_{k+1|k}$ weighted by a gain $\boldsymbol{J}_k$. The state $\hat{\boldsymbol{x}}_{k+1|N}$ at $k+1$ can be interpreted as an observation of the state at k with the observation model being the evolution model. Equivalence between the variables used for Kalman update and for Kalman smoothing is shown in Table 2.1. The main difference is that in the filtering case, the measurement uncertainty is often uncorrelated with the state estimate uncertainty whereas in smoothing the next estimated state $\hat{\boldsymbol{x}}_{k+1|N}$ uncertainty is correlated with the predicted state $\hat{\boldsymbol{x}}_{k+1|k}$ uncertainty. The detailed derivation of the Kalman smoothing equations is provided in Appendix B.

These equations show that to perform smoothing, the evolution model does not need to be inverted. However, the matrix $\boldsymbol{P}_{k+1|k}$ has to be inverted which can be computationally expensive. Despite this step, our testing shows that, on the problem considered in this thesis, the smoothing step is much faster than the filtering which requires to invert a matrix the size of the dimension of the measurement space.

It is important to note that since the smoothing starts from the last estimate and update previous estimates, the last estimate at epoch N is not improved in any way. This makes sense since the Kalman filter computes the optimal state estimate $\hat{\boldsymbol{x}}_{N|N}$ using the measurements $\{\boldsymbol{z}_k\}_{k\in[\![1:N]\!]}$. It is therefore already computed using all available information.

The aforementioned equations are applicable for a linear system with Gaussian centered noises. However, Kalman filtering has many variations (extended / unscented [Gong et al., 2013] / cubature [Arasaratnam and Haykin, 2011] / etc.) that can account for non-linearity and non-Gaussian noises. The same is true for Kalman smoothing, for instance the Extended Kalman smoother can be achieved by replacing the matrix $\boldsymbol{F}_{k+1}$ of a linear evolution model by the Jacobian of the non-linear model (as it is done to obtain the Extended Kalman filter).

2.2.2. Global Optimization Schemes

While filtering techniques are widely used for real-time state estimation, they assume that a state only depends on the previous state and on the current measurements. Also, filtering schemes are typically reliant on a linear description of the problem. When the system is not linear, it is linearized around a point. This does not result in optimal estimates. Optimization schemes process measurements as a batch to estimate every state simultaneously. These methods iteratively minimize a cost function. The function can be re-linearized at each iteration resulting in a better handling of non-linearity. Because all states are estimated using all measurements simultaneously, measurements do not need to depend on a single state. This enables such methods to introduce loop closure strategies, thus improving localization in areas traversed multiple times.

For these reasons, optimization schemes are used for mapping and, when high-frequency localization is not required. In particular, Simultaneous Localization And Mapping (SLAM) problems are often addressed using optimization techniques.

Their main drawback is the processing time which can vary widely depending on the type of measurements received and if loop-closure constraints are introduced.

2.2.2.1. iSAM

One family of such techniques is the iterative smoothing and mapping (iSAM) techniques [Kaess et al., 2008]. The SAM problem can be formulated by a minimization problem with the residuals of the evolution and observation models. iSAM aims at finding the states $\{\boldsymbol{x}_k\}_{k\in[\![0:N]\!]}$ and landmark positions $\{\boldsymbol{m}_i\}_{i\in[\![1:M]\!]}$ that are the most likely knowing the observations $\{\boldsymbol{z}_j\}_{j\in[\![1:J]\!]}$. The problem can be written as

$$\hat{\boldsymbol{x}}_{0|N}\dots\hat{\boldsymbol{x}}_{N|N},\hat{\boldsymbol{m}}_1\dots\hat{\boldsymbol{m}}_M = \underset{\boldsymbol{x}_0\dots\boldsymbol{x}_N,\boldsymbol{m}_1\dots\boldsymbol{m}_M}{\arg\max}\ p\left(\boldsymbol{x}_0\dots\boldsymbol{x}_N,\boldsymbol{m}_1\dots\boldsymbol{m}_M,\boldsymbol{z}_1,\dots,\boldsymbol{z}_J\right) \tag{2.22}$$

The probability density function p can be decomposed using the evolution and observation models (defined as for Kalman filtering)

$$p\left(\boldsymbol{x}_0\dots\boldsymbol{x}_N \mid \boldsymbol{z}_1,\dots,\boldsymbol{z}_N\right) = p\left(\boldsymbol{x}_0\right)\prod_{k=1}^{N} p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}\right)\prod_{j=1}^{J} p\left(\boldsymbol{z}_j \mid \boldsymbol{x}_{k_j},\boldsymbol{m}_{i_j}\right) \tag{2.23}$$

where $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}\right)$ is defined by the evolution model and $p\left(\boldsymbol{z}_j \mid \boldsymbol{x}_{k_j},\boldsymbol{m}_{i_j}\right)$ is defined by the observation model assuming the observation $\boldsymbol{z}_j$ observed at time k_j is associated to the map feature $\boldsymbol{m}_{i_j}$.

Assuming the models are affected by Gaussian noises, solving Equation 2.22 is equivalent to solving

$$\hat{\boldsymbol{x}}_{0|N} \dots \hat{\boldsymbol{x}}_{N|N}, \hat{\boldsymbol{m}}_1 \dots \hat{\boldsymbol{m}}_M = \underset{\boldsymbol{x}_0 \dots \boldsymbol{x}_N, \boldsymbol{m}_1 \dots \boldsymbol{m}_M}{\arg\min} \sum_{k=1}^{N} \left\| f\left(\boldsymbol{x}_{k-1}\right) - \boldsymbol{x}_k \right\|^2_{\boldsymbol{Q}_k} + \sum_{j=1}^{J} \left\| h\left(\boldsymbol{x}_{k_j}, \boldsymbol{m}_{i_k}\right) - \boldsymbol{z}_j \right\|^2_{\boldsymbol{R}_j} \tag{2.24}$$

where $\|\boldsymbol{e}\|^2_{\boldsymbol{E}} = \boldsymbol{e}^\top \boldsymbol{E}^{-1} \boldsymbol{e}$ is the squared Mahalanobis norm of $\boldsymbol{e}$, with covariance matrix E.

As $\|\boldsymbol{e}\|^2_{\boldsymbol{E}} = \left\| \boldsymbol{E}^{\top 1/2} \boldsymbol{e} \right\|^2$, the Global Optimization problem is in fact a Least Square Estimation problem. It can therefore be solved using traditional resolution strategies. The problem becomes finding $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{x}_0 & \dots & \boldsymbol{x}_N & \boldsymbol{m}_1 & \dots & \boldsymbol{m}_M \end{bmatrix}^\top$ as,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\arg\min} \left\| \boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b} \right\|^2 \tag{2.25}$$

where $\boldsymbol{A}$ and $\boldsymbol{b}$ are the result of linearizing Equation 2.24.

It is important to note that unlike in filtering schemes, iSAM does not treat the evolution model and the observation model any differently. Instead, both provide a factor that will influence the minimization. The two models only differ in the type of underlying variables of the state they link. The evolution model produces equations that link components of the state vector concerning the robot actual state with other components also concerning the robot state (but at a different time). On the other hand, the observation model produces equations that link components of the state vector concerning the robot state with components concerning features of the environment being mapped. The minimization does not treat one type of equation differently than the other.

Equation 2.25 can be solved by classic least squares resolution techniques. However, as new observations are received, the entire resolution has to be redone entirely. This is fine for small state dimensions but becomes quickly cumbersome when the dimension increases. To remedy to this effect, iSAM introduces a new way to add a new observation to an already solved system.

iSAM uses a $\boldsymbol{QR}$ decomposition of the matrix $\boldsymbol{A}$, where $\boldsymbol{Q}$ is an orthogonal matrix and $\boldsymbol{R}$ is an upper triangular matrix. $\boldsymbol{Q}$ being orthogonal, Equation 2.25 can be rewritten using the equivalence

$$\left\| \boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b} \right\|^2 = \left\| \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ 0 \end{bmatrix} \boldsymbol{\theta} - \boldsymbol{b} \right\|^2 = \left\| \begin{bmatrix} \boldsymbol{R} \\ 0 \end{bmatrix} \boldsymbol{\theta} - \boldsymbol{Q}^\top \boldsymbol{b} \right\|^2 \tag{2.26}$$

the system can them be solved by back-substitution since $\boldsymbol{R}$ is upper triangular. The last variable can be directly estimated as it only depends on a single element. Knowing the last variable, the second to last can be estimated as it depends only on the last two variables, and so on.

In a typical resolution scheme, the decomposition would have to be recomputed entirely every time a new state or observation is added. iSAM proposes to add the new information not to the original matrix but to the already decomposed system. The new measurements might depend on some variables in the middle of the state vector, thus introducing non-zero terms in the lower part of the triangular matrix $\boldsymbol{R}$. The decomposition can then be done only on the square block of $\boldsymbol{R}$ affected by the new information. It is done using the known rotation strategy that iteratively zero-out elements of $\boldsymbol{R}$ until it is triangular. The rotations needed to make the matrix triangular are added to $\boldsymbol{Q}$ ($\boldsymbol{Q}$ being built by aggregating rotations, it is a rotation matrix in this $\boldsymbol{QR}$ decomposition). The $\boldsymbol{QR}$ decomposition is therefore achieved not by recomputing it from scratch but leveraging the work previously done. This step alone enables to iteratively decompose the problem as new observations arrive. One significant problem left is that the resulting matrix $\boldsymbol{R}$ will not be sparse, which is preferred for large systems. To improve the matrix sparsity, iSAM uses a reordering step which moves variables that are related (by a measurement) closer to each other. Thus, the portion of the matrix that will be affected by measurements linking variables located in the original states at different locations will be much smaller because these variables have been moved closer to each other in the new state.

2.2.2.2. iSAM2

iSAM has been improved to better deal with variable reordering, partial estimation and re-linearization. Variable reordering can be a burden when the matrix is stored in its sparse form as the index references cannot simply be switched. iSAM2 [Kaess et al., 2012] introduces a representation of the matrix as a Bayes tree where each node corresponds to a block of the matrix and each leaf of this block corresponds to other blocks depending on the root. In a tree structure, the order in which the leaves are referenced is irrelevant, thus reordering can be done at almost no cost. Also iSAM2 enables partial estimation as inference is done from the root to the leaves, inference can be stopped before it reaches some leaves. If the evolution of a part of the tree is small enough, the leaves of this section do not need to be updated.

2.2.2.3. Concurrent Filtering and Smoothing

iSAM2 still has the issue that the estimation time is not constant and therefore real-time applications are not suited to this method. An evolution of the method has been proposed to deal with this issue [Williams et al., 2014]. A solution is to force the root of the tree to correspond to the variables needed to estimate future variables. Thus future variables only depend on this root and are thus part of an isolated leaves series rooted at the main root of the tree. Inference on this isolated branch can be done very quickly. Periodically, the branch is included in the main tree where loop closure information can be added and a new root is chosen with which the process can repeat. Hence the isolated branch can be estimated quickly while in parallel the rest of the tree can be updated and take the time it needs.

Although this smoothing method enables to guarantee real-time estimation, the accuracy of real-time estimation is not improved over the Kalman estimation [Lange et al., 2013]. It would, however, be interesting if the studied problem had to integrate loop closures.

2.2.3. Bounded Error Schemes

The methods presented to this point, use modeling of the uncertainties through probability distributions, both Kalman filtering and iSAM assume that the errors can be modeled by known distributions. The localization solution is therefore a point with associated distribution. Knowing the distribution of the solution enables the users to evaluate the confidence intervals for different risks. Depending on the risk tolerance of the system, different intervals can be computed to check if the vehicle can safely operate. One of the main drawbacks of the aforementioned methods is their difficulty with dealing with non-linear systems. Non-linearity is addressed using local linearizations of the system. These cannot accurately describe how error distributions are affected.

Another type of method to solve the localization problem is set-membership state estimation. Here, the errors are not described by distributions but rather by sets (intervals). The error is known to be within a set with probability 1. Starting from a large set of possible values, each observation adds a constraint to the solution set. As more observations are obtained the solution set becomes smaller and smaller. This method is well suited for problems with non-linear observation models (e.g. range only localization [Jaulin, 2011]). The solution sets can be any part of the considered space. They do not even need to be connected sets making set-membership state estimation naturally able to deal with multi-hypothesis solutions.

Another advantage of set-membership is that the true value can be guaranteed to be in the solution set as long as the bounded error assumptions (with known bounds) are valid. While probabilistic methods can be used to compute confidence intervals for small risks, there is never a guarantee that the solution falls within that interval. For probability distributions with non-bounded support, e.g. Gaussian distributions, a non-zero risk always remains. Assuming the sets used to model observation errors are accurate and correct, interval resolution methods can estimate the solution set for the vehicle state in a guaranteed fashion. This feature of interval analysis makes such methods well suited to safety-critical applications where the solution needs to be guaranteed. In cases where some constraints can be broken, q-relaxation methods have been developed to allow q constraints not to be satisfied. This enables these methods to deal with some potentially unreliable observations.

Such methods have been used for autonomous vehicle applications. [Wang and Lambert, 2018] have solved the localization problem using low-cost sensors combined with a map of the environment. The map is an interval map in which features are referenced directly as sets. One of the main advantages of interval analysis and constraint programming is the ability to add highly non-linear constraints. [Drevelle and Bonnifait, 2013] used this feature of interval methods to solve the localization problem in urban canyons. Using a map of the drivable space, the authors added constraints to limit the solution set to points on the road. Using q-relaxation, they were also able to deal with spurious observations caused by multipath and non line-of-sight measurements.

Localization and SLAM do not require past estimates to be conserved. The current interval states are simply updated. Some applications (e.g. surveying) require past states to also be estimated. The interval representation of space can be extended to time to create tubes, interval functions over time. Constraints over the time dimension can then be introduced to perform tasks such as loop closures [Aubry et al., 2013, Rohou et al., 2018].

Interval analysis has several drawbacks. If the sets used to model the observation are not accurate, no solution can be found. This problem is addressed by q-relaxation techniques but this increases computational cost significantly if a lot of observations have to be considered. Hence, the methods are best suited for use with accurate sensors for which observation set can be defined accurately. Also unlike with probabilistic approaches, interval methods do not benefit from receiving the same observation multiple times. With probabilistic methods, receiving the same information twice result in an increased confidence in the solution. Because interval methods are guaranteed methods, receiving the same set twice does not bring more information.

Probabilistic and interval methods are not mutually exclusive. Both Kalman filtering [Tran et al., 2017] and particle filtering [Abdallah et al., 2008, Merlinge et al., 2016] have been extended to use intervals. The state vectors (and covariance matrices for Kalman) are then modeled using intervals rather than single values.

2.3. Proposed Architecture

2.3.1. Motivation And General Outline

Autonomous vehicles require the localization system to perform several tasks in order to provide a good enough solution. Indeed, the localization system is subject to many potential error sources that need to be considered in the estimation to produce an accurate solution. It is therefore required to develop a generic architecture able to cope with all the error sources while maintaining the strict requirements of an autonomous vehicle localization system.

Real world measurements are not perfect measurements that can be used naively to estimate the vehicle state. Measurements suffer from delays that can be caused by transmission delays or by the computational time required to produce the measurement. Moreover these delays being different for every sensor (and not necessarily constant) the measurements can be received by the localization system in a different order that they were produced.

Measurements can also be ambiguous. Measurements of geo-referenced features are particularly prone to ambiguities as matching detection to map features is hard. Dealing with such ambiguities is detrimental to the accuracy and integrity of the localization system.

Even if accurate matches can be found, the observations can have errors due to inaccurate prior information. Indeed, maps are static snapshots of the environment that are only valid at the time the mapping is done. It is therefore essential for the localization system to include or enable the inclusion of mechanisms to either mitigate or eliminate such faults.

Localization systems are also affected by longer-term problems. As the vehicle ages, parameters that might have been assumed static are found to change slightly. Tires wear out, and sensor biases slowly change over time.

While all the aforementioned problems need to be considered, the localization system needs to succeed in its main goal, to provide an accurate and high frequency real-time localization solution.

To address these problems while satisfying the requirements of an autonomous vehicle localization system, a layered architecture is presented. Each layer runs at a different rate to perform different tasks aimed at improving localization. The layers are depicted on Figure 2.2 with the functions they perform and each function is detailed in the following subsections. This framework relies mostly on the *Filtering Layer* which is in charge of computing the current localization solution in real-time. The other layers are there to assist the main layer for tasks that are not achievable in real time. Two supporting layers are included in this framework to perform tasks aimed at either helping the real time localization (matching layer) or future drives

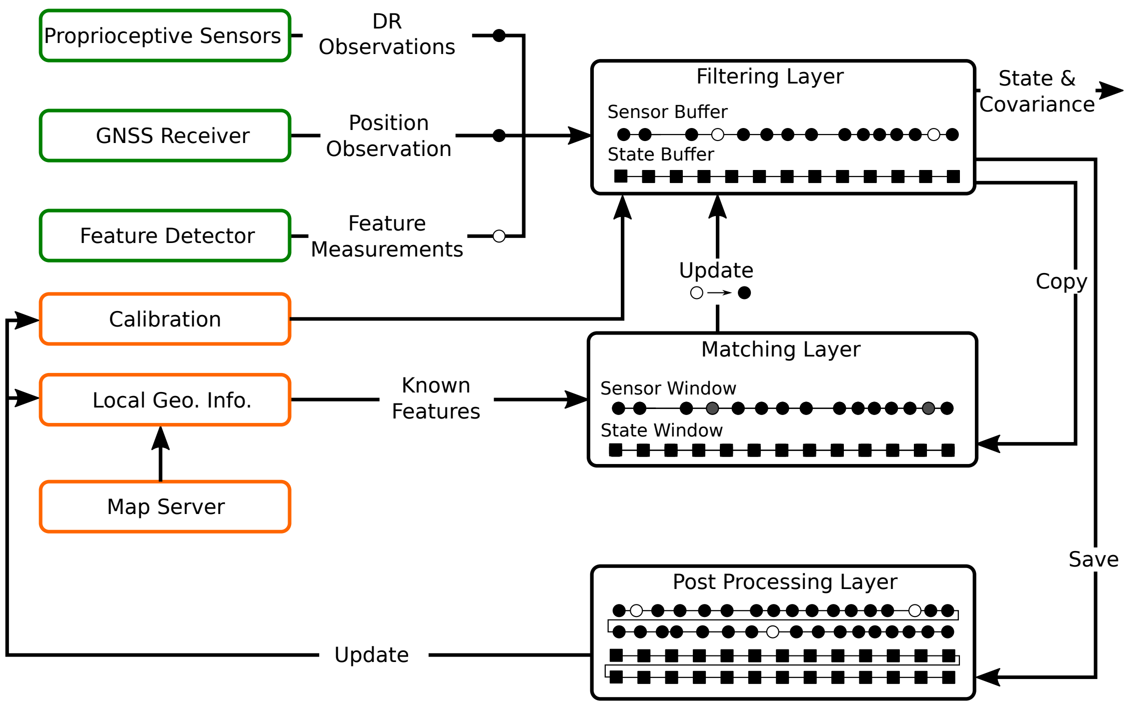


Figure 2.2.: Buffered Localization Architecture

(post processing layer). The main layer is independent of the supporting ones and is only aware of the observations it has available. The supporting layers help the localization either by affecting the observation available for localization or by updating parameters used by the main localization layer. The following section explains in more details this structure and the sharing of information between layers.

2.3.2. Observation vs Measurements

The localization system uses several types of sensors to estimate the vehicle state. In this work a distinction is made between measurements and observations.

Some sensor outputs are ready to be used for state estimation. They do not require further processing. These sensors include proprioceptive sensors, pose from GNSS receivers, etc. These particular sensor outputs will be referred to as observations. Observations are measurements for which the observations vector $\boldsymbol{z}$, the observation model $\boldsymbol{x} \longmapsto h(\boldsymbol{x})$ (with its Jacobian $\boldsymbol{H}$) and the covariance matrix $\boldsymbol{R}$ (such that $\boldsymbol{z} - h(\boldsymbol{x}) \sim \mathcal{N}(0, \boldsymbol{R})$) are known. Observations are therefore ready to be used in a filtering scheme.

Some sensor outputs received by the localization system are not considered as observations. These include lane markings and road signs. The measurement (distance to lane marking or road sign position) might be available but the observation model of such measurements cannot be directly known. The measurements need to be matched to mapped features in order to be able to predict the expected observation. Once the matches of the measurements have been found, the measurements are considered to be observations as they can be used for state estimation.

In Figure 2.2, measurements are represented by empty circles. The observations are filled circles. The system has several buffers containing these measurements. All buffers are sensor buffers that can contain both observations and simple measurements. Thus the buffers contain measurements that are not yet usable in the estimation process. The main estimation process does not try to turn measurements into observations, this is left to the parallel *Matching Layer* to do.

Measurements do not necessarily become observations in the localization system. Some measurements might not have corresponding map features. These measurements will thus never be used to estimate the vehicle state. However, these measurements are not useless as they can be used in post-processing strategies to improve the map.

It is important to note that some measurements of map features can be directly observations. For instance, this is the case for map-aided detection systems. When the detection system actively searches for a specific map feature, there is no matching needed and the outcome of the detection system is directly an observation that can be used. In Appendix E a map-aided crosswalk detection system is presented. The system searches for crosswalks based on areas of interest stored in the map. If a crosswalk is detected no association is needed as the detected crosswalk is assumed to be the crosswalk referenced in the map in the area of interest. This detector therefore directly outputs an observation.

2.3.3. Real-Time Filtering Layer

The main layer is aimed at providing the actual localization solution. It performs the fusion of observations to estimate the vehicle state in real-time. This layer is agnostic to the type of observations available, it simply estimates the state based on the given observations. Although it is affected by tuning parameters (biases, etc.), the variables that most affect it are the observations.

This layer has to provide high frequency estimates in real-time, thus it does not perform computationally expensive tasks. Matching measurements to features and parameter tuning are delegated to the two other layers of the localization system.

2.3.3.1. Handling delays

This layer does solve some of the problems when performing state estimation for real systems. In particular observation delays can be dealt with efficiency in real-time.

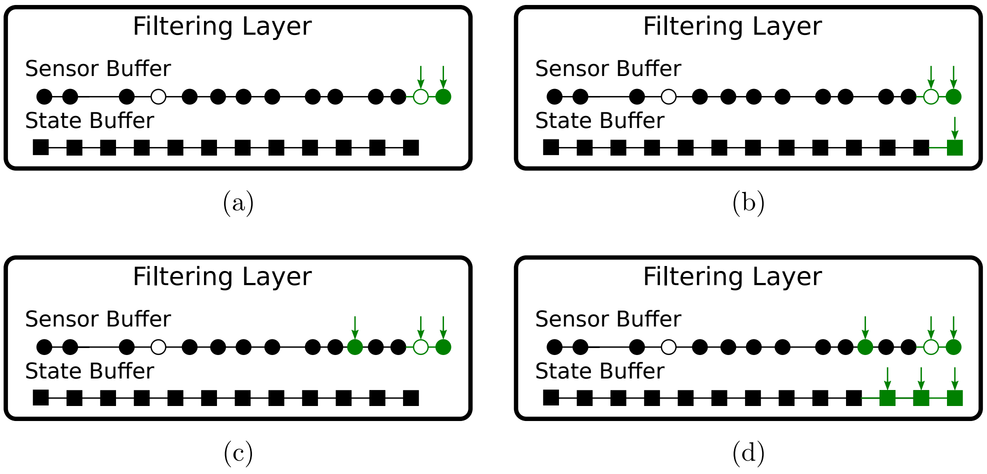


Figure 2.3.: Comparison of buffer updates with and without observation delays. When observations have no delays (a), the new state estimates are obtained with a single update (b). However, when observations are delayed (c) or updated, previous states have to be recomputed (d).

Instead of using augmented states, this layer deals with delays by recomputing the states. To do this, the observations are not used directly as they are received. Instead, the observations are stored in an observation buffer. The observations are added to the buffer at the appropriate place such that they are correctly ordered by measurement times instead of arrival times. The observation buffer therefore describes the available observations as if the system had no delays (of course the observations that have not been received yet cannot be known in advance). Additionally, the state estimates are also stored in a buffer such that the estimation can be restarted from an earlier time and to provide the other processes with the states that have been estimated with observations from the sensor buffer. In the general case, new observations are received without delays and added to the end of the measurement buffer (as shown in Figure 2.3 (a)). These observations are used to estimate the next state (Figure 2.3 (b)). However, if observations are received with delay, the last state estimates have to be recomputed to include the delayed observation at the appropriate time in the estimation (Figure 2.3 (c, d)).

Storing measurements in a buffer in the correct order compensates the effect of reception delays. However, it does nothing to solve for the difference between measurement timestamps and estimation timestamps. Indeed, the states are estimated at a regular pace regardless of the times of the received measurements. The processes are asynchronous. Therefore the measurement timestamp will never be exactly the same as the estimation timestamp. Some measurements will not be influenced much by this difference. Speed, yaw rates or distances to lane markings do not vary significantly in an estimation period T_e. Thus these observations can be used without further consideration. Some measurements can be very sensitive to this. In par-

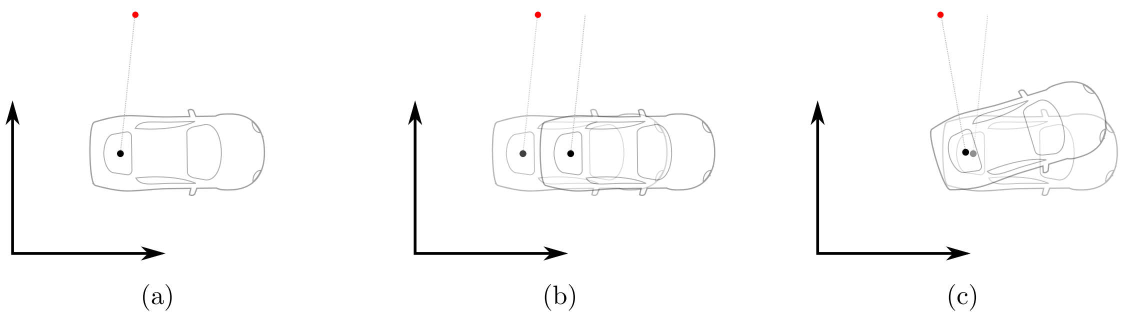


Figure 2.4.: Effect of unhandled delays on state estimation. (a) shows the true situation at measurement time, (b) shows the visible situation at estimation time, (c) shows the result of estimation without accounting for delays

ticular, road signs measurements need to be carefully used. Indeed, as road sign measurements can influence the vehicle estimated orientation. At common vehicle speeds, the predicted observation can vary in the order of decimeters. As Figure 2.4 illustrates, this can result in the apparition of biases in the estimation of the vehicle yaw that have disastrous influence on future state estimation. At 10 m/s not accounting for the difference between observation times and estimation times can result in errors of 20 cm. Such errors on a road sign observed 10 m away from the vehicle results in errors of over one degree. To prevent this problem, the observation needs to be fused at the proper time. To handle such small delays, the observation model $h(\cdot)$ is slightly changed. Since the observation is not related to the state at time k but to a state at time $k - \epsilon$ (where $\epsilon < 1$), the observation model has to be modified to relate the delayed observation at time $k - \epsilon$ to the computed state at time k. This is done using the standard evolution model $f(\cdot)$ applied on a negative time step $-\epsilon$ (noted $f_{-\epsilon}$) as shown in Equation 2.27.

$$\boldsymbol{z}_{k-\epsilon} = h\left(f_{-\epsilon}\left(\boldsymbol{x}_k\right)\right) \tag{2.27}$$

The linearized version of this observation model is

$$\boldsymbol{H}_k = \boldsymbol{H}_{k-\epsilon} \cdot \boldsymbol{F}_k \tag{2.28}$$

where $\boldsymbol{H}_{k-\epsilon}$ is the Jacobian of $h(\cdot)$ linearized around $f_{-\epsilon}\left(\hat{\boldsymbol{x}}_{k|k-1}\right)$ and $\boldsymbol{F}_k$ is the Jacobian of $f_{-\epsilon}(\cdot)$ linearized around $\hat{\boldsymbol{x}}_{k|k-1}$.

With this approximation, the difference between the measurement timestamp and the estimation timestamp can be compensated without having to compute an intermediate state separately for the delayed measurement.

2.3.3.2. Gating

Another task performed by this layer is a simple gating before the fusion is performed. Some observations provided to the localization systems might be noticeably wrong. To remove such observations, a gating step is used. The gating step checks the likelihood of the innovations $\boldsymbol{y}_k$ given the predicted state $\hat{\boldsymbol{x}}_{k|k-1}$. If obtaining the innovation is too unlikely, the system rejects the observation and does not use it. The observations that do not satisfy Equation 2.29

$$\boldsymbol{y}_k^\top \boldsymbol{S}_k^{-1} \boldsymbol{y}_k \leq G \tag{2.29}$$

where the innovation $\tilde{\boldsymbol{y}}$ is defined as in Equation 2.6.

The threshold G is chosen using a χ^2 distribution as shown in Equation 2.30. It depends on the dimension of the observation space $d_{\boldsymbol{z}_k}$ and of the risk α of rejecting good observations.

$$G = F_{\chi^2}^{-1}\left(1-\alpha, d_{\boldsymbol{z}_k}\right) \tag{2.30}$$

where F_{χ^2} is the cumulative distribution function of a χ^2 distribution.

2.3.3.3. Interaction with Other Layers

Measurements are added to the sensor buffer whatever source they come from. However not all measurements are used for state estimation. Some measurements require additional processing (e.g. matching) to be used. These measurements stay in the buffer and will be used when these processings have been performed (for instance by the low frequency layer).

This process continuously estimates the vehicle state. At each new time k, a new state is estimated. To estimate the states, Kalman filtering is used as it enables high-frequency estimation and provides uncertainty estimates. The filter can be run from time $k-1$ to time k to estimate the new state based on the new observations. In this framework, the filter is not simply run for one time step. New observations are not necessarily added at the head of the measurement buffer. Therefore, the filter is run from the most recent unaffected state. For instance, if a new observation (either a new observation or an existing measurement that has been matched) has been added to the buffer at time $k-5$, the filtering is run starting from the state estimate $\hat{\boldsymbol{x}}_{k-6|k-6}$ all the way forward to time k (thus 6 filtering steps). This process is done to ensure that the state estimates buffer always accurately depicts the observation buffer available at the time.

This process does not directly communicate with the low frequency and post-processing layers. It is only affected by the sensor buffer and the tuning parameters. The state

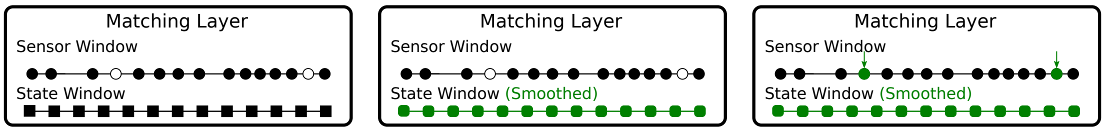


Figure 2.5.: Buffers updates performed by the matching layer. The state and sensor windows (a) obtained from the Filtering Layer are used to find matches. The states are first smoothed (b) to be used to find matches. Once matches have been found some measurements become observations (c).

estimates are not corrected directly by the other processes. Instead the measurement and state buffers are regularly extracted by the other layers, used to produce new observations or tune parameters, the observation buffer and parameters are then updated to reflect the new information. An update in an existing observation in the sensor buffer will be treated as an entirely new observation and the filter runs from a previous time to accurately account for the change.

Because of the limitation imposed to this estimation layer, few observations can be used directly when they are added to the buffer. Only GNSS pose observations and observations used for dead-reckoning do not need matches and can be used in the estimation without intervention from the other processes (although tuning the observation models of DR sensors is also essential).

2.3.4. Parallel Matching Layer over a Window of Data

The matching layer is here to assist the high frequency estimator. Since the high frequency estimator needs to provide real-time localization, it cannot perform computer intensive tasks. The matching layer runs in parallel at low frequency to perform those tasks and provides its result to the main estimator.

One of such tasks is the matching of detections to map features. While there are fast matching methods that could be run at high frequency, the accuracy and reliability needed for autonomous driving applications require the use of more advanced matching strategies. These more computationally expensive methods are run at a lower frequency to turn detected features into observations that can be used for state estimation. The sensor buffer is updated to include the new observations. These observations will be used in the next filter iteration to estimate the states.

This process is not a state estimator running in parallel with the high-frequency estimator. It uses the state and sensor buffers saved by the main layer. Periodically the top layer buffers are copied to the low frequency layer. The main estimator can thus keep updating its buffer while the other process performs its tasks. Using the state estimate buffers and the sensor buffer, the matches are found using the method presented in Chapter 4. Some of the more dynamic parameters (the yaw rate bias

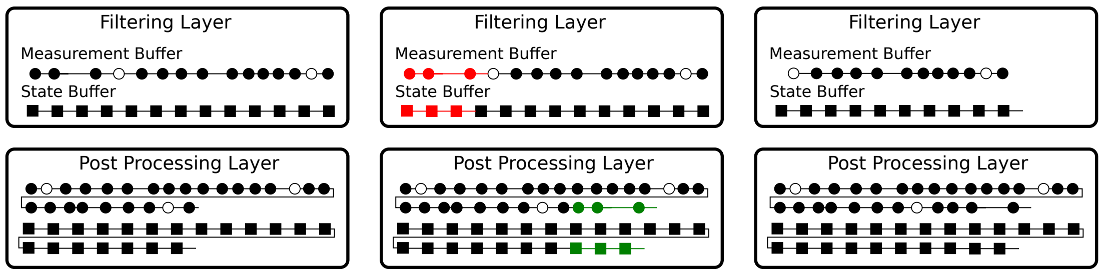


Figure 2.6.: Recording of states and observations for post processing. The oldest states and observations from the filtering layer (red) are not discarded but rather moved to the end of buffers saved in the post-processing layer (greed).

for instance) of the odometric model can also be tuned by the method described in Chapter 3.

This layer only affects the main estimation through the matches and parameters it provides. It does not directly update the states of the main layer. Once the buffers are finished to be processed, they are discarded. The buffers from the main layer are again copied and the process repeats.

Because the chosen estimation scheme is a Kalman filtering scheme, only the most recent state $\boldsymbol{x}_k$ is estimated using all available observations. The older states are estimated using the observations available at their estimation time. Hence each state in the state buffer is not estimated with the same information. Some processes require an accurate estimate of all states in the buffer. To achieve this result with a Kalman filtering scheme, a smoothing step is used. The first step of most algorithms run by this layer is therefore to apply the Kalman smoothing step to improve the states of the entire buffer as shown in Figure 2.5.

2.3.5. Post Processing over All Recorded Data

The final layer of the localization system is the post-processing layer. Although once the end of the drive is reached, there is no point in improving the localization itself, there are still processes that can be run to improve future drives. To perform these tasks, all states and observations of the entire drive are recorded. When the states and observations are purged (and therefore will not be updated) from the filtering layer buffers (Figure 2.6 (b)), they are saved in long term buffers. These buffers contain every filtered and predicted states as well as every observation used to generate these states. The measurements that were not successfully matched are also stored. The post-processing buffers will therefore grow indefinitely until the post-processing tasks are run. This might happen at the end of a drive if the available memory is sufficient to store all this data or happen periodically on long sequences. In any case, the filtering layer does not consider these states and can keep estimating the real-time state while only keeping a limited length buffer.

2.3.5.1. Calibration

Calibration is essential for the system to handle parameters that can vary. Odometric models involve several parameters that need to be regularly calibrated in order to limit the natural drift of the estimate. Some parameters vary somewhat rapidly, e.g. yaw rate bias, and therefore would need to be calibrated during the drive but others vary much more slowly and can thus be calibrated using one drive for the next. In particular, this is the case of wheel circumferences that will decrease as the tires age and get used up. Slowly varying parameters can thus be estimated using the information of an entire trajectory without requiring highly accurate ground truth systems.

2.3.5.2. Map Errors

Map-aided localization is reliant on the accuracy of the map to provide accurate localization. Maps are known to change and therefore the system needs a way to detect errors in mapped features such that it can avoid using such features and hopefully correct the error. The map matching will prevent obvious errors detectable in real-time but post-processing offers the possibility of detecting more subtle errors and to modify the map accordingly.

2.4. Conclusion

Several approaches aim at solving the localization problem using buffer of states and/or observations. Some methods try to estimate the poses as a batch while other uses sequential processing to achieve the same result. While global optimization strategies should generally produce better estimates, they are hard to use for real-time state estimation. Sequential schemes are very efficient and are already used for time-sensitive applications but can have difficulties to cope with erroneous data and non-linear systems.

The proposed framework, relies on a sequential strategy (Kalman Filter) to ensure the real-time constraint of localization for autonomous driving are met. To cope with the drawback of such methods, state and sensor buffers are used. Using a multi-layer architecture, the buffers can be processed, at high-frequency to provide the state estimate, at lower frequency for matching and parameter tuning, and in post processing to detect map errors and further tune vehicle parameters. This structure enables the main estimation layer to benefit from more advance processing while always being able to achieve its mission of providing high frequency real-time state estimates. With this framework the problems of delayed, out-of sequence, ambiguous measurements can be addressed without penalizing the time performance of the estimation.

The next chapters will cover aspects of each layer starting with Chapter 3 on the dead-reckoning and its calibration which is essential to the *Filtering Layer*. Chapter 4 details the *Matching Layer* and Chapter 5 covers the map error detection performed by the *Post Processing Layer*.

3. Dead-Reckoning Model Calibration using Smoothing

Contents

3.1. Introduction

A strong Dead Reckoning (DR) system is essential for autonomous driving. As such systems rely only on proprioceptive sensors, they are a system that works in most situations as it is minimally affected by environmental factors. Dead-reckoning is essential for safety as the vehicle always needs to be able to drive to safety areas (emergency lane) in case of critical failures of the rest of the system.

The accuracy of dead reckoning system depends widely on the application. Ranging from small robot odometry that drifts quickly to orbital launchers that rely heavily on dead-reckoning and inertial measurements to navigate. Localization using DR systems is subject to an unavoidable drift. This drift is firstly caused by the random-walk phenomenon that emerges from the integration of proprioceptive measurements over time and, secondly, by inaccuracies in the modeling of both the displacement of the mobile frame and the observation models of the sensors. The proprioceptive measurement errors are all the more significant that DR sensors are affected by systematic errors (biases/scaling factors), which increase the drift of the localization system. These parameters have to be tuned before the DR system can be used.

Commercial vehicles are already equipped with sensors enabling dead-reckoning estimation. These sensors are used for Anti-lock Braking System (ABS) and Electronic

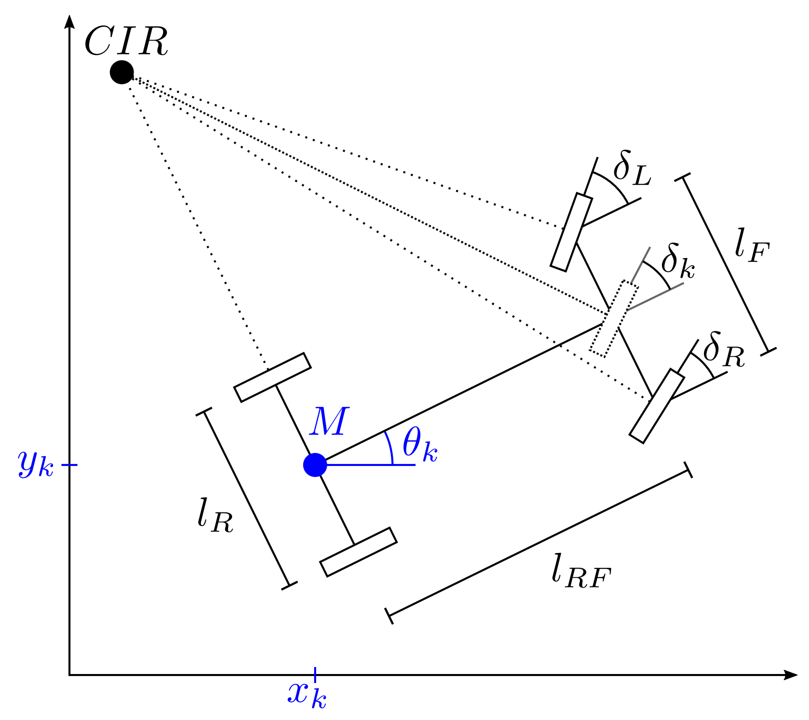


Figure 3.1.: Graphic of a vehicle following the Ackermann steering geometry

Stability Control (ESC). Using such sensors and assuming that the vehicle follows the Ackermann steering geometry (see Figure 3.1), the relative movement of the vehicle can be inferred. Regular calibration is required as automotive grade sensors have to be affordable [Harr and Schaefer, 2018] (and therefore prone to errors). Vehicles have wheel speed sensors from which the vehicle speed and yaw rate can be inferred. This can only be done by accurately knowing the wheel circumferences ρ_{XX}[1]. These parameters change slowly as the wheels are used. Vehicles also directly provide an estimation of the vehicle longitudinal speed. This observation is affected by a scaling factor a_v. The steering wheel angle is also measured by the ESC system to know the driver's intent. A yaw rate sensor is also used to provide measurements of the vehicle rotation unaffected by environmental factors but subject to a bias b_ω.

In this work two vehicles (shown in Figure 3.2) were used to test the method on experimental data. The two vehicles are both Renault ZOEs but differ slightly. Because one vehicle is the first generation of Renault ZOE and was modified to enable autonomous control and the other is a standard production vehicle some of the available information differ. The old Renault ZOE provides observations of the steering wheel angle, the wheel encoder ticks, the yaw rate from the gyrometer and the vehicle speed. The newest model does not provide wheel encoder ticks but rather wheel rotation speeds. The steering wheel angle is also not available. The

[1]where XX corresponds to RL: rear left, RR: rear right, FL: front left, FR: front right

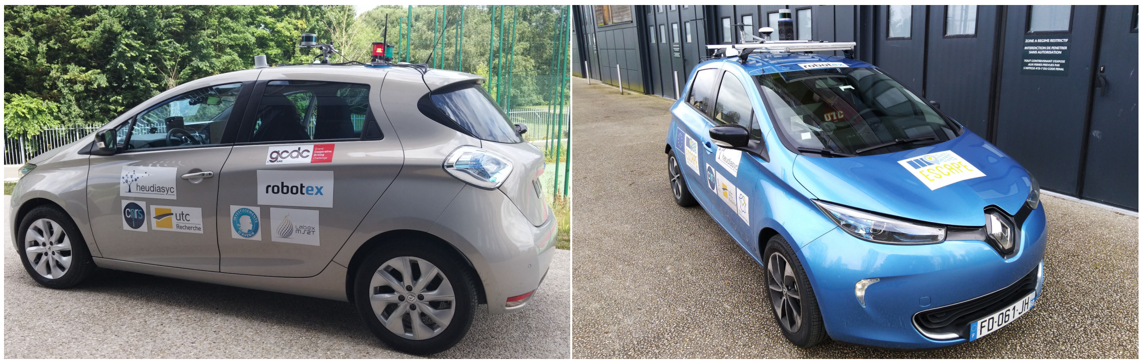


Figure 3.2.: The two vehicles used to obtain experimental data. The gray vehicle (left) is a first generation Renault ZOE ZE which was modified to enable autonomous operation, the blue vehicle is a more recent Renault ZOE ZE which has not been modified for autonomous control.

speed measurement is available but is not used for this vehicle as explained later in Subsection 3.4.2. Since the available observations differ between the two vehicles, the calibration parameters that will be estimated also differ. The wheel circumferences ρ_{XX} and the gyrometer bias are needed for both vehicles but the scaling factor affecting the speed measurement is not needed for the most recent vehicles. Hence, six parameters, listed in Table 3.1, are estimated for the older vehicle while only five parameters are estimated for the most recent Renault ZOE.

The next Section presents methods classically used for the calibration of such systems. The choice of the evolution model used is then detailed, followed by an overview of the considered observations. Finally, the proposed calibration method using smoothed state estimates is presented and an experimental evaluation of the method is shown.

3.2. State of the Art

3.2.1. Zero velocity UPdaTe (ZUPT)

The simplest way to identify the gyro bias is to use the moment when the vehicle is static. Indeed since the front wheels of vehicles cannot turn at 90 degrees, the vehicle has to move to turn. Therefore, when the vehicle does not move the true yaw rate should be zero. In practice, the measured yaw rate is not null and might vary. For instance, on a Renault Zoe, the typical bias is in the same order of magnitude as the yaw rate discretization step. The returned value when the vehicle is stopped alternates between one and two multiplied by the discretization step. ZUPT estimates the bias by averaging the returned values when the vehicle is stopped

Table 3.1.: Dead reckoning sensors and parameters affecting their observation models. On newer vehicles, the speed measurement is not used. Hence, a_v is only needed for older generations.

Renault ZOE	First generation	Second generation
Dead-reckoning data		
Steering wheel angle Δ_k	✓	–
Wheel encoder ticks Δ_k^{XX}	✓	–
Wheel rotation speed ω_k^{XX}	–	✓
Yaw rate ω_k	✓	✓
Vehicle speed v_k^{CAN}	✓	✓
Calibration parameters		
Wheel circumference Rear-Left: ρ_{RL}	✓	✓
Wheel circumference Rear-Right: ρ_{RR}	✓	✓
Wheel circumference Front-Left: ρ_{RL}	✓	✓
Wheel circumference Front-Right: ρ_{RR}	✓	✓
Gyrometer bias: b_ω	✓	✓
Speed scaling factor: a_v	✓	–

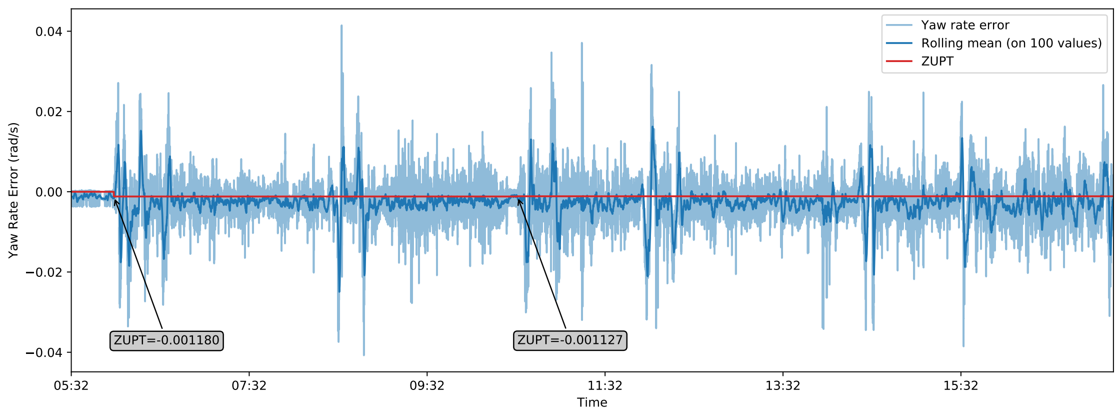


Figure 3.3.: Yaw rate error (blue) and estimated bias (red) using ZUPT.

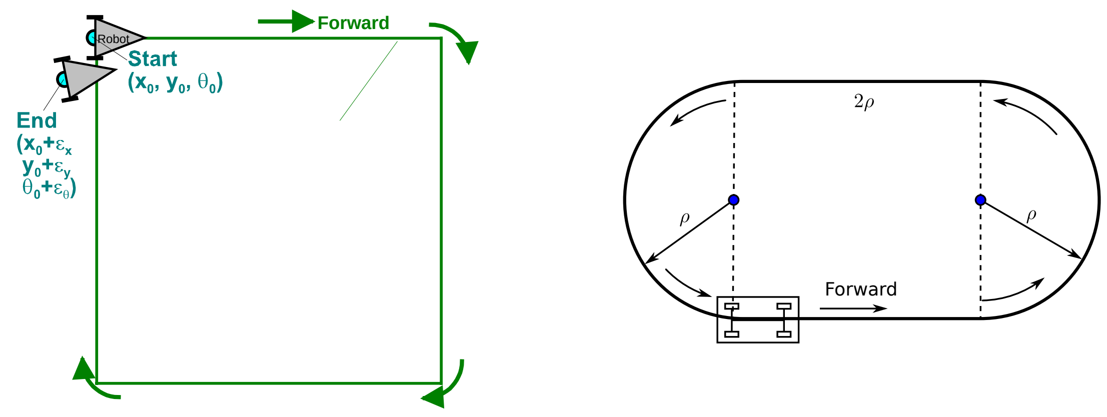


Figure 3.4.: Trajectories used for calibration. (a) Square calibration trajectory used in [Borenstein and Liqiang Feng, 1995]. (b) Modified trajectory used in [Lee and Woojin Chung, 2008] to enable car-like robot calibration.

[Basnayake, 2009]. The wheel encoders can be used to detect when the vehicle is stopped, at which point the bias b_ω is computed,

$$b_\omega = \frac{1}{|\{k : v_k = 0\}|} \sum_{k:v_k=0} \omega_k \tag{3.1}$$

where v_k is the vehicle longitudinal speed estimated using the wheel encoders.

Although ZUPT would appear to be able to estimate the true yaw rate bias, it appears on real trajectories that the bias cannot be well estimated simply by averaging the static yaw rates. With Figure 3.3, it is clear that the real error (blue) is different than the one estimated using ZUPT (red). Even though the ZUPT estimation is correct when the vehicle is stopped, the bias changes when the vehicle is moving, making the ZUPT estimate not entirely accurate.

3.2.2. Calibration using objective paths

Odometric systems not properly calibrated will result in large errors accumulated over time. For a properly calibrated system, the noise affecting the estimate should be centered. Hence, such systems would result in different estimates even if they go through the same trajectory, each estimate being random. If the odometry is not properly calibrated, the resulting estimates will not be completely random. The estimates will have systematic errors, e.g. the estimates systematically drift left or right of the expected trajectory.

[Borenstein and Liqiang Feng, 1995] proposed a method to calibrate a tank-like robot using a square trajectory. By having the robot perform the trajectory several times, the authors are able to discern how the odometric system should be calibrated.

[Lee and Woojin Chung, 2008] have extended the approach to car-like robot by replacing the square trajectory by an oblong shape.

In both cases the trajectory is chosen in order to make the interesting parameters (wheel base, wheel circumference) easily observable using simply the final state estimate.

This kind of method is conceivable for manufacturers. Vehicles could be tested on specific circuits to calibrate vehicles before the sale. Although some parameters are not expected to change throughout the vehicle operation (wheelbase), others can change slowly (wheel circumferences) and some can even change within a few minutes (gyro biases). Hence, a generic calibration method that does not rely on specific trajectories is still required.

3.2.3. Online Estimation

Calibration can sometimes be done directly as part of the state estimation. Indeed, calibration parameters can be seen as additional dimensions of the vehicle state. Given proper measurements, the parameters become observable and can therefore be estimated through Kalman filtering or other estimation schemes [Xiao et al., 2019].

[Brunker et al., 2017, Zhao et al., 2016] have calibrated gyro bias and wheel circumferences using this approach. This strategy is also used with other estimation schemes. For this application, the augmented state can be

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & v_k & \omega_k & b_\omega \end{bmatrix}^\top \tag{3.2}$$

where b_ω is the gyro bias.

The observation model associated to a gyrometrer observation would be

$$\omega = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \boldsymbol{x}_k \tag{3.3}$$

There would be no difference between the yaw rate and gyro bias for the observation. The estimation would only be different thanks to the evolution model which links the yaw rate to the vehicle yaw ($\theta_k = \theta_{k-1} + T_e \omega_{k-1}$) and does not link the gyro bias to anything.

The yaw rate bias is not directly observable. It only becomes observable thanks to the evolution model. This results in inaccurate estimation of the yaw rate and gyro bias at the start as there are no differences between the two. The gyro observation would contribute entirely to the yaw rate estimation when the bias is not included. While including the bias in the state results in the observation contributing to both the yaw rate and the bias. At the start, the yaw rate is underestimated while the bias is overestimated.

Adding calibration parameters to the state reduces the observability of the useful element of the state. This phenomenon only worsen as more parameters are added.

Online parameter estimation has also been done using Marginalized Particle Filtering. In fact [Lundquist et al., 2014] have shown faster convergence when estimating wheel circumferences using particle filtering rather than Kalman filtering.

3.2.4. Calibration with a Ground Truth

When it is available, a ground truth on the vehicle state can be used for calibration. Ground truth can be obtained using highly accurate (and expensive) Inertial Measurement Unit (IMU) and Real-Time Kinematic (RTK) GNSS systems. Since ground truth provides the true states of a dataset, it can be used to obtain the expected true observation using the observation models. The parameters can be estimated in order to make the model best predict the observations actually obtained.

Thus for a set of observations $\boldsymbol{z}^{(k)}$, with the observation model $h\left(\boldsymbol{x}_k, \boldsymbol{p}\right)$ depending on both the state and a set of parameters, the parameters can be estimated by solving,

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_k \left(\boldsymbol{z}^{(k)} - h\left(\check{\boldsymbol{x}}_k, \boldsymbol{p}\right)\right)^2 \tag{3.4}$$

where $\check{\boldsymbol{x}}_k$ are the ground truth states.

If the observations have different noise models, the problem could also be weighted by the observations covariance matrices $\boldsymbol{R}_k$,

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_k \left(\boldsymbol{z}^{(k)} - h\left(\check{\boldsymbol{x}}_k, \boldsymbol{p}\right)\right)^T \boldsymbol{R}_k^{-1} \left(\boldsymbol{z}^{(k)} - h\left(\check{\boldsymbol{x}}_k, \boldsymbol{p}\right)\right). \tag{3.5}$$

In most cases, for dead-reckoning sensors, the noise model is the same for all measurements. Therefore only Equation 3.4 needs to be solved in practice.

More recently, approaches using deep learning have been used. [Chong et al., 2016] have used deep learning to better model temperature fluctuations of the gyrometer (which affects the measurements). In [Brossard et al., 2020], the authors learned corrections to apply to the measurements to reduce noise. Using this approach results in lower drifts of the dead reckoning. [Chen et al., 2018] go even further by directly estimating displacement using raw measurements fed into a neural network.

3.3. Evolution Model

In order to predict the vehicle position based on internal measurements, an evolution model needs to be used. The evolution model describes how the vehicle moves relative to a previous pose. The choice of an evolution model affects the state vector that needs to be estimated. Several approaches can be found in the literature.

For vehicle control, a dynamic model is often preferred. Modeling the vehicle dynamic enables to account for effects such as tire deformation, shock absorbers, which can be relevant at regular vehicle speeds [Kong et al., 2015].

Localization generally uses, simpler, kinematic models. Although such models cannot model the various strengths affecting the system, they only require estimating the first derivatives of the state variables leading to a lighter estimation process. Kinematic models differ in the hypothesis they make to model the system.

A kinematic point model considers the state $\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & \dot{x}_k & \dot{y}_k & \dot{\theta}_k \end{bmatrix}^T$. The evolution model is trivial as the derivative of the main variables are already included in the state. With this model the vehicle movement is not constrained. For this reason this model is typical of unconstrained GNSS localization. A generic GNSS receiver is not constrained to move in a particular direction as such this model is appropriate. Moreover all variables can be observed by the receiver using the pseudoranges for x_k and y_k and the Doppler measurements for $\dot{x}_k$ and $\dot{y}_k$. Drones are also a type of robot for which this model is appropriate.

Unlike drones, cars are constrained in the way they can move. A car cannot simply move in any direction, it is constrained to move along its heading θ_k. As such, the two velocity components $\dot{x}_k$ and $\dot{y}_k$ are functions of the vehicle heading and its longitudinal speed v_k. This model, often referred as the *Tank model* (or unicycle), has the state vector $\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & v_k & \dot{\theta}_k \end{bmatrix}^T$. A discrete evolution model of this representation [Fouque et al., 2008] with a time period T_e small enough and a constant speed assumption is:

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix}}_{\boldsymbol{x}_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ \dot{\theta}_k \end{bmatrix}}_{\boldsymbol{x}_k} + T_e \begin{bmatrix} v_k \cos\left(\theta_k + \frac{T_e}{2}\dot{\theta}_k\right) \\ v_k \sin\left(\theta_k + \frac{T_e}{2}\dot{\theta}_k\right) \\ \dot{\theta}_k \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

This model does not include all the constraints of a car movement. Indeed, vehicles are not only constrained in their velocity but also in their rotation. Cars cannot turn while staying in place. They have to move (forward or backward) in order to rotate. This effect is considered in the *Bicycle model*. Instead of modeling the vehicle rotation directly using its yaw rate $\dot{\theta}_k$, it is modeled instead using its front wheel angle δ_k. As explained in the following section, the two front wheels of cars do not have the same angle. For this reason, the angle of a virtual front wheel is used instead, the virtual front wheel is placed centered between the actual front wheels of the vehicle. The change in heading of the vehicle can be expressed using its speed v_k, the virtual front wheel angle δ_k and its wheelbase (distance between the rear and front wheels) l_{RF}. An evolution model [Kong et al., 2015, Rajamani, 2006] of the state $\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \theta_k & v_k & \delta_k \end{bmatrix}^T$ is derived as

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ \delta_{k+1} \end{bmatrix}}_{\boldsymbol{x}_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ \delta_k \end{bmatrix}}_{\boldsymbol{x}_k} + T_e \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \\ \frac{v_k}{l_{RF}} \tan\delta_k \\ 0 \\ 0 \end{bmatrix} \tag{3.7}$$

Although this model describes the vehicle possible movement more accurately, it has not been chosen for the evolution model. Indeed, the predictions obtained using the evolution model are only as good as the accuracy of the state estimates. Therefore, the ease with which the variables of the state vector can be estimated needs to be considered when choosing an evolution model. The yaw rate can be derived from many observations (gyrometer, rear wheel speeds) with a linear model whereas the steering wheel angle is only available through a single measurement. Moreover, observations can rarely be modeled using the steering angle with a linear (or at least simple) model. For these reasons, the tank model is chosen to model the vehicle movement in this work.

3.4. Observations

Modern vehicles are equipped with several proprioceptive sensors that can be used for DR. In this section, the sensors and observations used in this work are presented. Additionally, the observation from a GNSS receiver is detailed as it is needed to avoid drifts of the estimation thus enabling a better calibration.

3.4.1. Wheel Speed Sensors

3.4.1.1. Rear Wheels

Wheel speed sensors are installed on all four wheels of most commercial vehicles. They are not only used to provide the driver with the vehicle speed but are also used by ABS and ESC to improve the vehicle braking or steering. These sensors

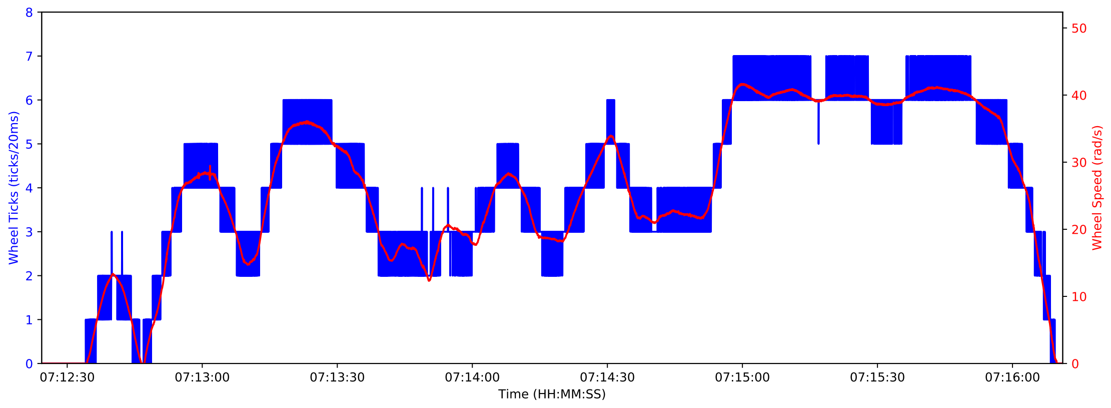


Figure 3.5.: Comparison of the rear wheel speed measurements (red) and wheel tick measurements (blue).

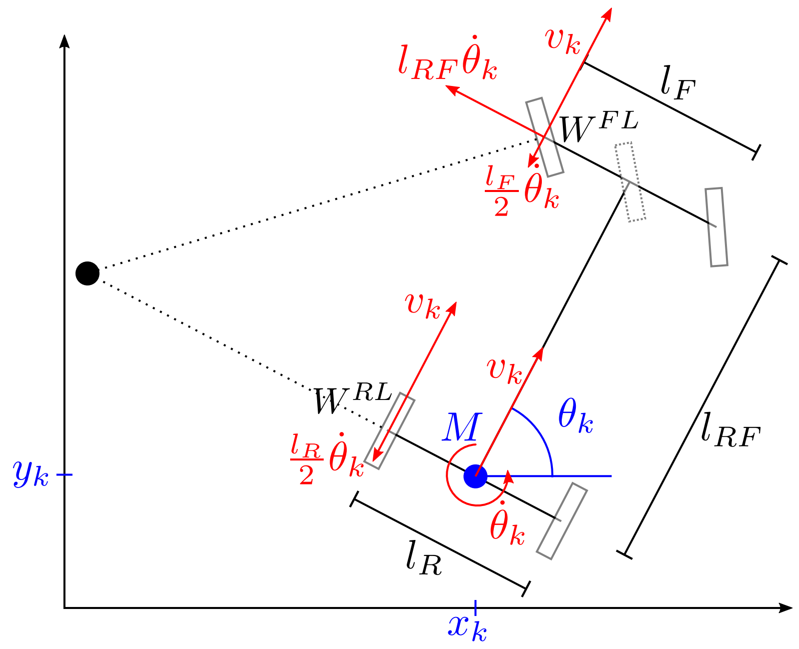


Figure 3.6.: Graphic of the wheel speeds on a four-wheel vehicle following the Ackermann steering geometry

are tachometers that consist in two parts: a toothed wheel and a detector. The detector senses when each tooth of the wheel passes in front of it (a tick). Using this signal, the speed of the vehicle can be deduced in two ways. By counting the number of ticks, the rotation of the wheel can be deduced. This measurement has the advantage of having a fixed measurement noise. Hence, the wheel rotation can be accurately tracked across a long time span. The only uncertainty of the system concerns the beginning and end of the measurement window. Since only an integer number of ticks can be measured, the position of the wheel at the beginning can only be known within a one tick interval. The same applies to the end of the measurement. Consequently, the error of a tick differential (equivalent to the wheel rotation) follows a triangular distribution within the interval $\left[-1;\ 1\right]$ tick. The other measurement type is obtained not by counting the ticks within a time interval but by timing the interval size between two ticks. The angle traveled by the wheel within a tick being constant, knowing the time it took to happen, the speed of the wheel is computed. This method provides a measurement of the wheel speed with some uncertainty which depends on the accuracy of the clock used to time the interval.

Sensors used on vehicles have few ticks per wheel turns (48 on a Renault ZOE) and provide measurements at high frequencies (100 Hz). With so few ticks and given typical vehicle speeds, the counted number of ticks does not change much from one measurement to the next which results in very noisy estimates of the wheel speeds. Timing the interval between ticks provides measurements much less noisy.

In both cases the observation model of each sensor can be derived from geometric considerations [Bonnifait et al., 2001]. From the Ackerman geometry, the speeds of each of the four wheels can be derived using the vehicle longitudinal speed and its yaw rate. The rear wheels have the simplest model as they do not turn and are aligned with the mobile frame center.

Under the assumption that the vehicle body is a rigid body, the speed of the body at the point of the rear left wheel can be expressed using the vehicle longitudinal speed v_k and its rotation rate $\dot{\theta}_k$. Assuming that the wheel is not subject to slipping, the speed of the wheel at the point of contact with the road expressed in the body frame (see Appendix A for a complete description of the vehicle frames) will be equal and opposite to the speed of the wheel center in the global frame. The sensors used in vehicles are not directly able to measure negative velocities. Therefore, the speed of the wheel at the contact point is computed simply as the speed of the point W^{RL} (see Figure 3.6). The speed of the rear left wheel is obtained using the vehicle longitudinal speed v_k and by adding the velocity caused by the lever arm between the estimation center and the wheel.

$$\frac{d\overrightarrow{OW^{RL}}}{dt} = \underbrace{\frac{d\overrightarrow{OM}}{dt}}_{\text{longitudinal velocity}} + \underbrace{\frac{d\overrightarrow{MW^{RL}}}{dt}}_{\text{velocity due to rotation}} \tag{3.8}$$

By expressing the vector coordinates in the global frame, the velocity of the wheel is obtained.

$$\begin{bmatrix}\dot{x}_k^{RL}\\ \dot{y}_k^{RL}\end{bmatrix} = \begin{bmatrix}v_k\cos\theta_k\\ v_k\sin\theta_k\end{bmatrix} + \frac{d}{dt}\begin{bmatrix}-\frac{l_R}{2}\sin\theta_k\\ \frac{l_R}{2}\cos\theta_k\end{bmatrix} \tag{3.9}$$

$$= \begin{bmatrix}v_k\cos\theta_k\\ v_k\sin\theta_k\end{bmatrix} + \begin{bmatrix}-\frac{l_R}{2}\dot{\theta}_k\cos\theta_k\\ -\frac{l_R}{2}\dot{\theta}_k\sin\theta_k\end{bmatrix} \tag{3.10}$$

$$= \begin{bmatrix}\left(v_k - \frac{l_R}{2}\dot{\theta}_k\right)\cos\theta_k\\ \left(v_k - \frac{l_R}{2}\dot{\theta}_k\right)\sin\theta_k\end{bmatrix} \tag{3.11}$$

The measured speed of the vehicle rear left wheel v_k^{RL} is the norm of the aforementioned vector,

$$v_k^{RL} = \sqrt{(\dot{x}_k^{RL})^2 + (\dot{y}_k^{RL})^2} \tag{3.12}$$

$$= v_k - \frac{l_R}{2}\dot{\theta}_k \tag{3.13}$$

The velocity of the rear right wheel is obtained from the same reasoning. The observation model for the tick measurements is

$$\Delta_k^{RL} = \frac{N_{wheel}}{\rho_{RL}} * T_{WT} * \left(v_k - \frac{l_R}{2}\dot{\theta}_k\right) \tag{3.14}$$

$$\Delta_k^{RR} = \frac{N_{wheel}}{\rho_{RR}} * T_{WT} * \left(v_k + \frac{l_R}{2}\dot{\theta}_k\right) \tag{3.15}$$

where N_{wheel} is the number of ticks per turns, ρ_{XX} is the circumference of the wheel XX, T_{WT} is the period of Wheel Top measurements.

For the measurement of the wheel rotation speed, the observation model is

$$\omega_k^{RL} = \frac{2\pi}{\rho_{RL}} * \left(v_k - \frac{l_R}{2}\dot{\theta}_k\right) \tag{3.16}$$

$$\omega_k^{RR} = \frac{2\pi}{\rho_{RR}} * \left(v_k + \frac{l_R}{2}\dot{\theta}_k\right) \tag{3.17}$$

Under the assumption of movement without slippage, the only errors affecting wheel tick measurements are the measurement errors due to the integer nature of tick measurements. The measurement error for any tick number XX_k is described by a uniform distribution between 0 and 1. Therefore the measurement error of the differential $\Delta_k^{XX} = XX_k - XX_{k-1}$ is described by a triangular distribution centered on 0 and ranging from -1 to 1. This error far outweighs other sensor noise that could occur and is therefore the only observation noise considered in this model.

Since this model is used in a Kalman filtering scheme, the error model needs to be approximated by a Gaussian distribution. The previous distribution is often approximated by a zero-mean Gaussian distribution with standard deviation $\sigma_k^{XX} = 1.0$.

Wheel speed measurements are not subject to these discretization errors. These errors can be modeled by Gaussian distributions. The numerical value for the wheel speed noise is given in the experimental results.

3.4.1.2. Front Wheels

The front wheel observation model is more complex but can be obtained using Ackermann steering geometry (see Figure 3.6). Unlike most methods using front wheel measurements, the angle of the virtual front wheel is not part of the state vector. The observation model has to be expressed using the yaw rate instead. Similarly to the rear wheels, the velocity of the front wheels is a combination of the longitudinal velocity v_k of the vehicle and of its yaw rate $\dot{\theta}_k$. Expressing Equation 3.8 for the front left wheel in the global reference frame results in,

$$\begin{bmatrix} \dot{x}_k^{FL} \\ \dot{y}_k^{FL} \end{bmatrix} = \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} l_{RF} \cos\theta_k - \frac{l_F}{2} \sin\theta_k \\ l_{RF} \sin\theta_k + \frac{l_F}{2} \cos\theta_k \end{bmatrix} \tag{3.18}$$

$$= \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \end{bmatrix} + \begin{bmatrix} -l_{RF}\dot{\theta}_k \sin\theta_k - \frac{l_F}{2}\dot{\theta}_k \cos\theta_k \\ l_{RF}\dot{\theta}_k \cos\theta_k - \frac{l_F}{2}\dot{\theta}_k \sin\theta_k \end{bmatrix} \tag{3.19}$$

$$= \begin{bmatrix} \left(v_k - \frac{l_F}{2}\dot{\theta}_k\right) \cos\theta_k - l_{RF}\dot{\theta}_k \sin\theta_k \\ \left(v_k - \frac{l_F}{2}\dot{\theta}_k\right) \sin\theta_k + l_{RF}\dot{\theta}_k \cos\theta_k \end{bmatrix} \tag{3.20}$$

Hence, the velocity of the front left wheel is

$$v_k^{FL} = \sqrt{l_{RF}^2 \dot{\theta}_k^2 + \left(v_k - \frac{l_F}{2}\dot{\theta}_k\right)^2} \tag{3.21}$$

The observation model of tick measurements is therefore,

$$\Delta_k^{FL} = \frac{N_{wheel}}{\rho_{FR}} * T_{WT} * \sqrt{l_{RF}^2 \dot{\theta}_k^2 + \left(v_k - \frac{l_F}{2}\dot{\theta}_k\right)^2} \tag{3.22}$$

$$\Delta_k^{FR} = \frac{N_{wheel}}{\rho_{FR}} * T_{WT} * \sqrt{l_{RF}^2 \dot{\theta}_k^2 + \left(v_k + \frac{l_F}{2}\dot{\theta}_k\right)^2} \tag{3.23}$$

For the measurement of the wheel rotation speed, the observation model is

$$\omega_k^{FL} = \frac{2\pi}{\rho_{FR}} * \sqrt{l_{RF}^2 \dot{\theta}_k^2 + \left(v_k - \frac{l_F}{2}\dot{\theta}_k\right)^2} \tag{3.24}$$

$$\omega_k^{FR} = \frac{2\pi}{\rho_{FR}} * \sqrt{l_{RF}^2 \dot{\theta}_k^2 + \left(v_k + \frac{l_F}{2}\dot{\theta}_k\right)^2} \tag{3.25}$$

It is important to note that the front wheels are more subject to slippage. Hence, the model is not valid when the vehicle experiences high accelerations. This limitation is acceptable since an autonomous vehicle should drive smoothly and limit accelerations for the passengers' comfort.

Also, the wheel tick being highly discretized measurements, it has been observed that using wheel ticks only for dead-reckoning results in highly unstable estimation of the vehicle yaw rate. This problem disappears as other observations (e.g. gyrometer and steering wheel angle) are used for the estimation.

The noise model used for the front wheels is the same as the one used for the rear wheels.

3.4.2. Speed observation

A measurement of the vehicle speed is also available through the vehicle Controller Area Network (CAN) bus. The exact source of this measurement is unknown. However, from the level of discretization, it was deduced that it most likely is an estimation of the vehicle longitudinal speed computed using the wheel rotation speed.

Because this measurement is most likely already computed using wheel rotation speeds, it will not be used for estimation when wheel rotation speeds are used (it is not used on the latest Renault ZOE). However, it is useful to have when only wheel ticks are used. Indeed, even though wheel ticks enable dead-reckoning, they result in unstable estimations. This is due to the resolution of the wheel encoders used on commercial vehicles. This is not a problem for localization as the errors due to the discretization will average out. However, it can be a problem for other systems (e.g. control) that expect a smooth estimation of the speed. For that reason this measurement is used only with wheel tick measurements. Since this observation almost directly corresponds to the vehicle longitudinal speed, it is only affected by a scaling factor a_v. Its observation model is simply,

$$v_k^{CAN} = \begin{bmatrix} 0 & 0 & 0 & a_v & 0 \end{bmatrix} \boldsymbol{x}_k \tag{3.26}$$

3.4.3. Gyrometer

The vehicle is equipped with several MEMS sensors among which is a gyrometer. The gyrometer provides measurements of the vehicle that, unlike information obtained from wheel speed sensors, are unaffected by slippage issues. However, the measurements are, as it is the case for every affordable gyrometer, noisy and subject to biases. The measurement ω of the sensor cannot be directly used as an observation of the vehicle yaw rate as it would result in an integration of the bias leading to highly inaccurate estimation of the vehicle heading. This results in the observation model described by,

$$\omega_k = h\left(\boldsymbol{x}_k\right) = \dot{\theta}_k + b_\omega \tag{3.27}$$

where b_ω is the gyrometer bias that has to be estimated.

The uncertainty attached to this observation has been evaluated by comparing ω_k to $h\left(\boldsymbol{x}_k\right)$ with $\boldsymbol{x}_k$ being the ground truth state. From this analysis, the measurement uncertainty has been found to be described by a Gaussian noise centered on the bias b_ω with standard deviation $\sigma_{\omega_k} = 0.003$ rad/s.

3.4.4. Steering Wheel angle

For the ESC to work properly, it needs to know the position of the steering wheel in order to prevent over or under steering. The steering wheel angle is therefore measured and available through the vehicle CAN Bus. The steering angle is useful because it defines fairly directly the angle of the front wheels.

The front left and front right wheels of the vehicles do not follow the same orientation, as shown in Figure 3.7. Indeed, taking the reasonable assumption that the vehicle is a rigid body, it always rotates around a point (possibly infinitely far away when there is no rotation). As such, the wheel on the inside of the curve has to perform a tighter turn than the other wheel. The steering wheel does not control directly the angles of the real wheels but rather the angle of a virtual wheel located on the longitudinal axis of the vehicle.

The virtual front wheel angle δ_k can be found using the longitudinal speed v_k and yaw rate $\dot{\theta}_k$. Using ground truth measurements, the virtual front wheel angle is

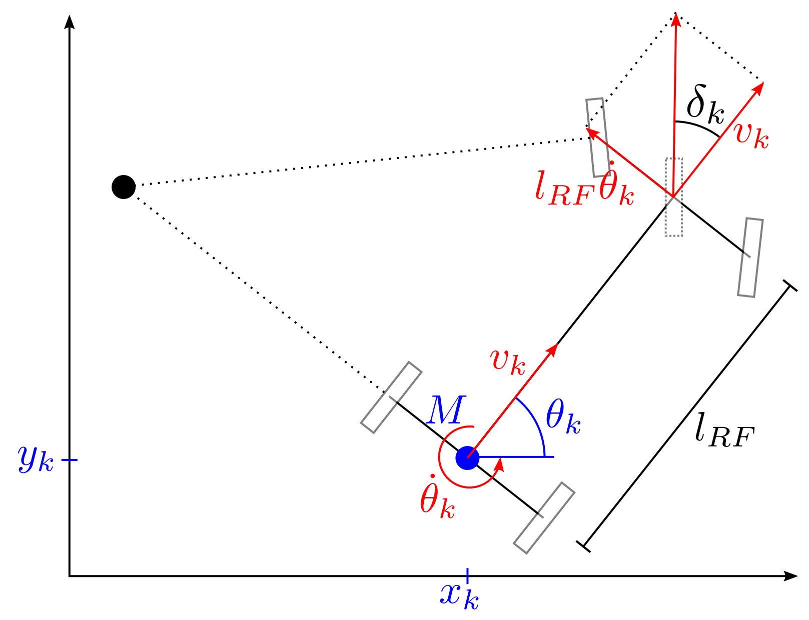


Figure 3.7.: Graphic of the virtual front wheel angle obtained assuming the vehicle is a rigid body.

computed, as shown in Equation 3.28, to find how it relates with the steering wheel observation.

$$\delta_k = \operatorname{atan2}\left(l_{RF}\dot{\theta}_k, v_k\right) \tag{3.28}$$

As can be seen on Figure 3.8, the relationship between the measured steering wheel angle Δ_k and the virtual front wheel angle δ_k is linear depending only of a reduction factor a_δ between the steering wheel and the virtual front wheel angle.

The observation model of the steering wheel angle is,

$$\Delta_k = a_\delta \operatorname{atan2}\left(l_{RF}\dot{\theta}_k, v_k\right) \tag{3.29}$$

3.4.5. GNSS

Inertial and odometric sensors are not sufficient for localization. These sensors produce estimates that inevitably drift with time. Using proper calibration, the drift can be significantly reduced. This is achievable by estimating the parameters (bias, scaling factors) that influence the observation models.

The sensors presented up to this point make only two pieces of information observable, the vehicle speed v_k and its yaw rate ω_k. The gyro provides information about the yaw rate only. The measurements of the wheel encoders involve both the speed

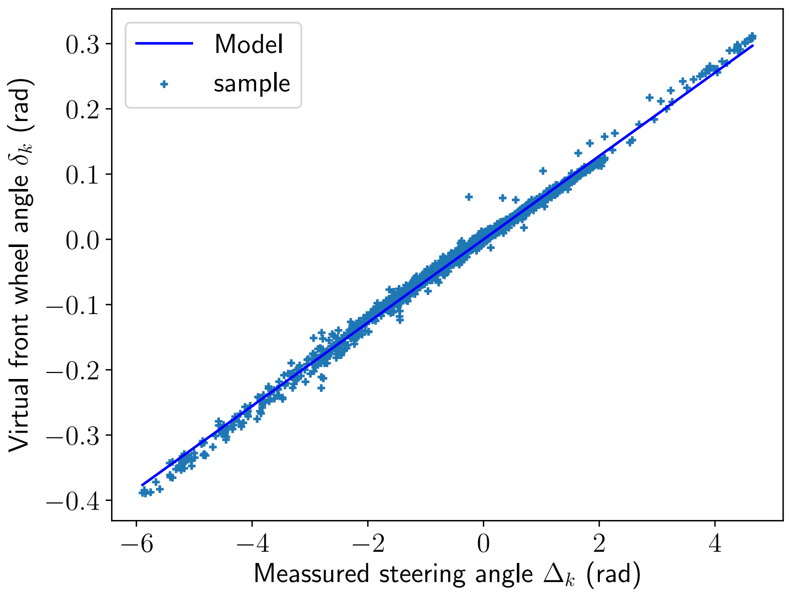


Figure 3.8.: Relationship between the measured steering wheel angle and the true angle of the virtual front wheel (obtained using the ground truth velocity and yaw rate). The two variable are clearly linearly dependent of each other.

and yaw rate and the steering wheel angle constraint the estimated yaw rate based on the vehicle speed. Therefore there is a redundancy in the source of information for these two variables. This enables partial calibration to be done without new sensors. For instance, the gyrometer bias can be partially estimated based on the yaw rate estimated using wheel speeds. However, some amount of bias will remain as wheel speeds themselves are not perfect measurements. To properly calibrate the observation models, an additional sensor is required. A sensor preventing the estimation to drift will result in better calibration. GNSS receivers offer such a solution.

By measuring the distance from the receiver to orbiting satellites, the positions of which are known, GNSS receivers are able to provide an estimation of the position of the receiver. The receivers provide positions obtained from pseudo-distances but also speed measures thanks to the Doppler effect affecting the satellite signals. The heading of the vehicle can be obtained from the direction of the speed vector assuming that the vehicle does not slide (unlike in marine applications).

In this work the GNSS pose is used as an observation. The observation model for the GNSS pose is obtained by considering the lever arm between the receiver position and the position of the vehicle reference frame (see Figure 3.9). The model described by Equation 3.30 only accounts for a lever arm aligned on the vehicle longitudinal axis.

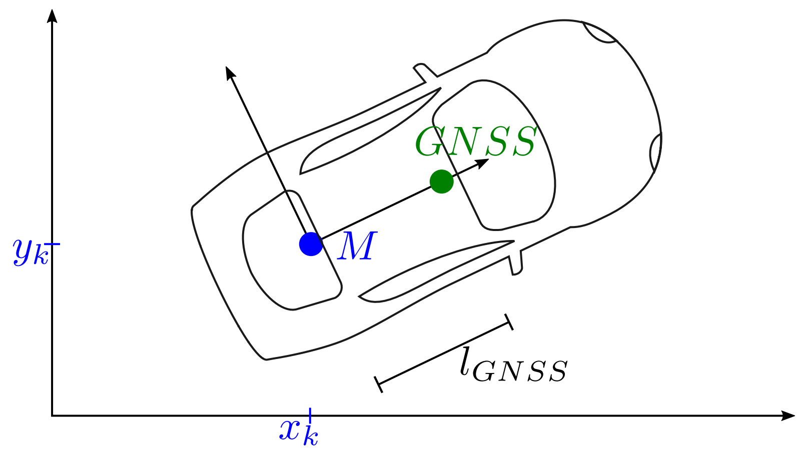


Figure 3.9.: Lever arm of the GNSS receiver antenna

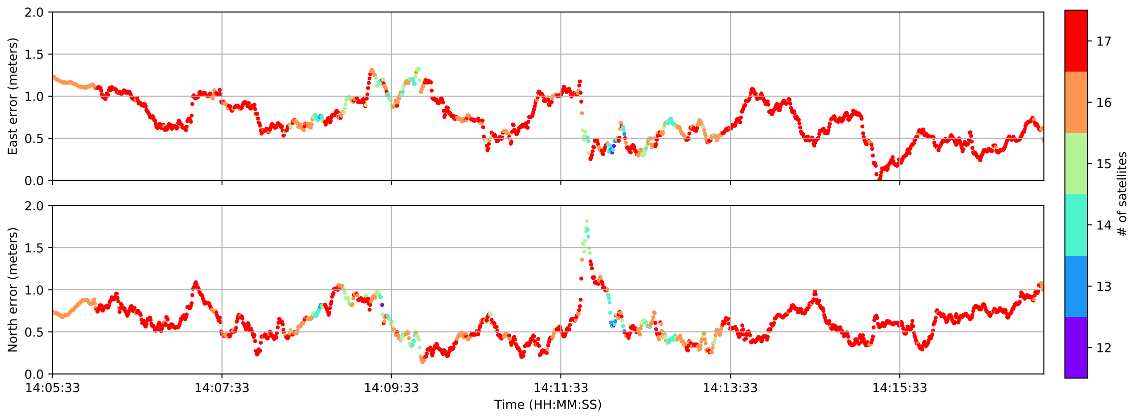


Figure 3.10.: Error of the GNSS position estimation in the East and North direction. The colors represent the number of satellites available. The data was obtained using a ublox-M8T receiver in Compiègne.

$$\begin{bmatrix} x_{GNSS} \\ y_{GNSS} \\ \theta_{GNSS} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + l_{GNSS} \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \\ 0 \end{bmatrix} \tag{3.30}$$

As the receiver measures the heading based on the observed speed, the heading cannot be trusted when the vehicle is static. Indeed, even when the vehicle is stopped, the receiver will measure small speeds. These small speeds are due to noise. The direction is therefore entirely a result of the noise and are not related the vehicle orientation in any way.

The main issue with automotive grade GNSS receivers is their noise model. Indeed such cheap receivers produce errors highly correlated in time (see Figure 3.10). This can be interpreted as a slowly moving bias affecting the GNSS estimate. The cause of this is twofold, the measured distances to the satellites suffer from correlated noise themselves (mainly from atmospheric delays) and the estimates produced by the receiver are not the result of a snapshot estimation (such as Least Square Estimation) but are rather the result of a filtering scheme. While this enables the estimates to be generally more accurate, it has the disadvantage of producing time-correlated noise that will affect further estimations. Because of this, the noise model used in the experiments is chosen larger than what is claimed by the receiver used. The specific noise model used is detailed in the experimental section.

3.5. Calibration

The parameters of the observation models change with time. Therefore it is essential to continuously re-calibrate the parameters. Since commercial vehicles cannot be expected to have high accuracy sensors able to provide ground truth measurements, ground truth calibration is not possible. However, autonomous vehicles host a wide range of sensors that together provide accurate localization for autonomous vehicles. Hence, the localization solution has to be close to the ground truth. In this work, the estimated states are used for calibration instead of the ground truth states. Theses states having been estimated using multiple sensors, the bias of one sensor is partially corrected by the others.

To further improve the filtered state estimates, Kalman smoothing is used. The buffer of state estimates is smoothed in order to propagate backward the accuracy gained from new observations. Once this is done, all states of the buffer have been estimated using all available measurements of the system. The observations older than the states have been considered through the filtering and the future observations have been considered by the smoothing. This is the best state estimate obtainable based on the available measurements.

The smoothed states $\hat{\boldsymbol{x}}_{k|N}$ are then used as a substitute to the ground truth states and the parameter calibration problem becomes,

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} \sum_{k} \left(\boldsymbol{z}^{(k)} - h\left(\hat{\boldsymbol{x}}_{k|N}, \boldsymbol{p}\right)\right)^2 \tag{3.31}$$

This minimization problem can be solved in our particular case using Least Squares because $\boldsymbol{h}$ is linear with respect to $\boldsymbol{p}$ for a given value of $\hat{\boldsymbol{x}}_{k|N}$. Since each parameter p_i appears in only one equation, the problem can be solved as multiple one-dimensional problems instead of one 6-dimensional problem.

As such, additive parameters such as the gyro bias is estimated using the mean as described in Equation 3.32

Table 3.2.: Standard deviations of the dead reckoning sensors.

Observation	Gyrometer	Speed	Wheel ticks	Wheel speeds	Steering
Std. Dev.	0.003 rad/s	0.1 m/s	1	0.23 rad/s	0.02 rad

$$b_\omega = \frac{1}{N} \sum_{k=1}^{N} \left(\omega_k - \hat{\dot{\theta}}_{k|N} \right) \tag{3.32}$$

Multiplicative parameters such as a_v are given by

$$a_v = \left(V^\top V \right)^{-1} V^\top \begin{bmatrix} v_1^{CAN} \\ \vdots \\ v_N^{CAN} \end{bmatrix} \tag{3.33}$$

with $V = \begin{bmatrix} \hat{v}_{1|N} & \dots & \hat{v}_{N|N} \end{bmatrix}$

3.6. Experimental Results

3.6.1. Experimental Setup

The calibration method has been tested on several trajectories recorded with two experimental vehicles. A Renault ZOE first edition first recorded some trajectories in 2018 with data from the gyrometer, wheel encoders, and steering wheels. Another Renault ZOE for which measurements from the gyrometer and wheel speeds are available was later used to record new trajectories in 2020. Both vehicles are equipped with a low-cost ublox M8T receivers configured to use the Global Positioning System (GPS) and Global Navigation Satellite System (GLONASS) constellations. The noise model used for this sensor is not the one provided by the receiver, as it is too optimistic, but instead a large constant variance noise will be used. The standard deviation of the model is chosen at 2 meters in both dimensions. This does not directly solve the problem of the biased noise but will enable more accurate sensors presented in Chapter 4 to influence the estimate more. Hence, the influence, on the state estimates, of the biased position observations will be reduced. The same high accuracy localization system (SPAN-CPT) with RTK corrections was also used to obtain ground truth states.

The standard deviations of the sensors used in these experiments is detailed in Table 3.2.

The evolution model noise used in the calibration experiments is presented in Table 3.3. It is important to note that at this stage the consistency of the estimation

Table 3.3.: Standard deviations of the evolution model noise. The values for the speed and yaw rate are chosen large such that their estimation is driven by the observations.

Dimension	Position	Orientation	Speed	Yaw rate
Standard Deviation	10^{-3} m	10^{-5} rad	10 m/s	1 rad/s

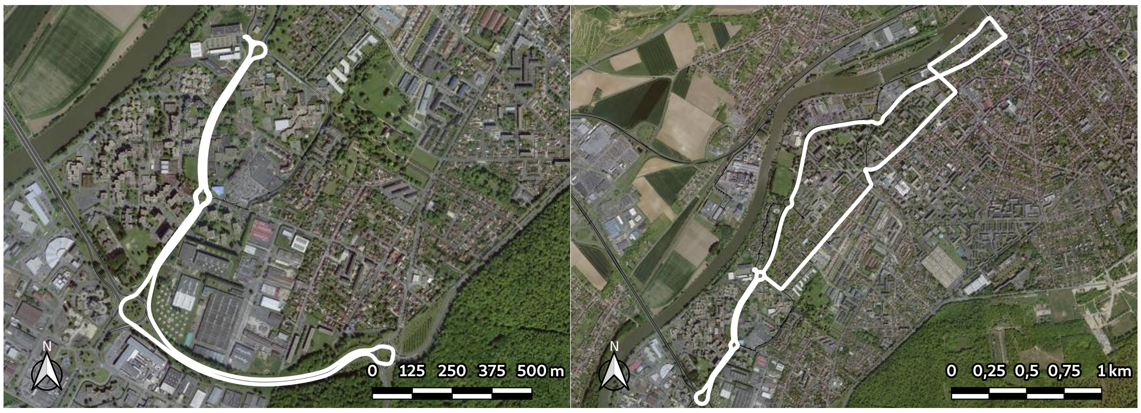


Figure 3.11.: Trajectories used to calibrate the parameters. (a) was used to calibrate trajectories recorded in 2018. (b) was used to calibrate trajectories recorded in 2020.

is not studied. The noise was chosen such that it correctly describes the drift of the localization starting from an accurate position. Because of the biased GNSS observations, the estimation would not be consistent with those values. In the following chapters, to improve the consistency of the estimates the standard deviations set for the position and yaw are increased respectively to 10^{-2} m and 10^{-4} rad.

For each set of experiments a calibration trajectory was used while the other trajectories were only used for evaluation. The two calibration trajectories are shown in Figure 3.11. More details about the vehicles and datasets can be found in Appendix A.

3.6.2. Parameter Estimation

The parameters calibration has been performed using the method described in the previous sections and compared to the well-known Zero Velocity Update (ZUPT) method. The method described in this work uses the smoothed estimates as substitutes for the ground truth. We therefore also compare it to a calibration where the ground truth is available. The ZUPT method estimates the yaw rate bias by averaging the yaw rate measurements when the vehicle is stopped and therefore not turning. The other parameters, speed scale factor and wheel circumferences are set

Table 3.4.: Parameters calibration results (ZUPT is used to calibrate b_ω only, the other parameters are the defaults).

Method	b_ω (rad/s)	a_v	ρ_{RL} (m)	ρ_{RR} (m)	ρ_{FL} (m)	ρ_{FR} (m)
GT	-0.00298	0.987	1.9166	1.9164	1.9134	1.9129
Ours	-0.00279	1.0	1.9198	1.9197	1.9177	1.9159
ZUPT	-0.00194	1.0	1.92	1.92	1.92	1.92

Table 3.5.: Dead reckoning drift using different calibration methods.

Method	position mean error	position error standard deviation
GT	0.79%	0.39%
Our method	0.81%	0.51%
ZUPT	0.93%	0.57%

at their nominal values (1.0 for the speed scale factor, 1.92 m for the wheel circumferences). Table 3.4 details the parameters estimated using the two methods. The estimated gyro bias is noticeably different between our method and the ZUPT technique. It can be noted that estimating the gyro bias by comparing the measurements to those of a high accuracy IMU yield a bias of -0.00298 rad/s on the calibration trajectory and -0.00269 rad/s on the testing trajectory. This method therefore manages to estimate the yaw rate bias within a 7% margin of error which is vastly better than by using the ZUPT technique (error of 35%). Regarding the speed scale factor and the wheel circumferences, the results are very close to the default values. This can be explained by the fact that the experimental vehicle is driven very little compared to personal vehicles. Hence, the wheels are not worn out.

To evaluate the quality of the calibration process and the performance of the dead reckoning method, we have carried out other experiments using a different, 5 km long, testing trajectory. The state has been initialized using the ground truth.

3.6.3. Drift Evaluation

We use the following methodology to evaluate the drift. When a DR distance of 100 m has been traveled, the difference between the estimated position and the ground truth is computed and recorded. The state is then corrected using the ground truth and the process continues. The quality of the dead reckoning is evaluated as the average distance drifted per 100 m. The results of the experiments are reported in Table 3.5 and the error distributions are shown in Figure 3.12.

In the context of autonomous driving it is also useful to know how long one can drive

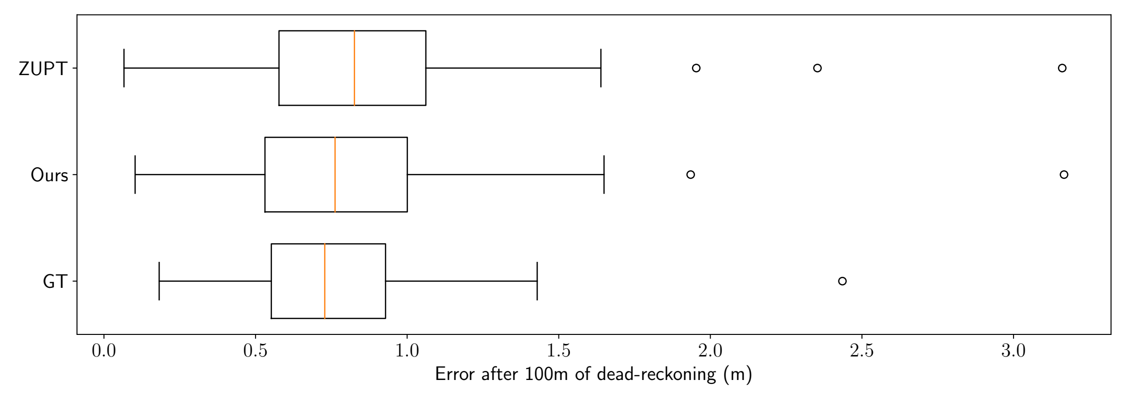


Figure 3.12.: Horizontal error distribution after 100 m traveled in DR.

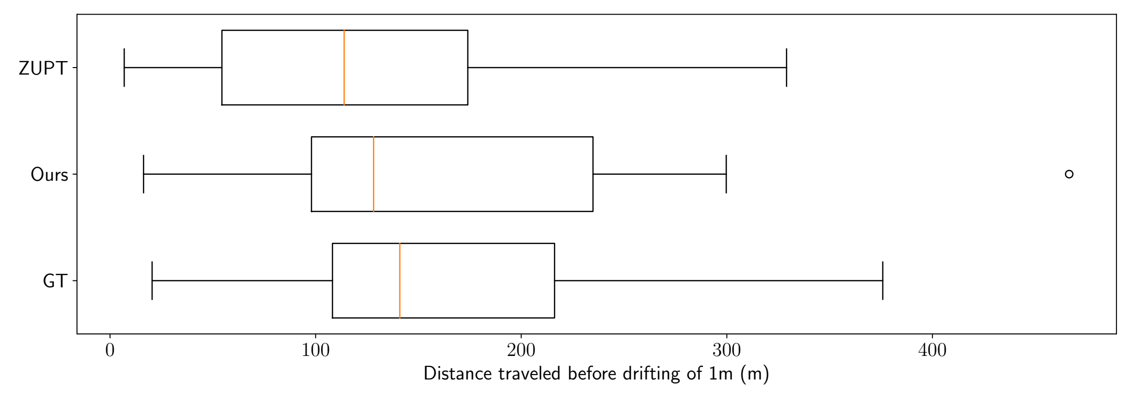


Figure 3.13.: Distance traveled in DR before reaching an error of one meter.

using DR without going outside of the lane. To evaluate this we apply a similar strategy than previously but this time the estimation is not corrected every 100 m but only when the estimation error reaches 1 m. The result of this test is shown in Figure 3.13. The proposed method increases quite significantly the DR navigation performance. Based on this experiment, we estimate that the car is able to maintain a 1-meter accuracy at least during 100 meters and sometimes it can be more than 200 meters. It should be noted that since the estimation is corrected only once the 1 m threshold is reached, the epochs at which the corrections happen are different for the two methods. This results in DR starting at different epochs and so comparing the results is difficult.

3.6.4. Dead Reckoning Estimation Results

The trajectories obtained by the different parameter calibration methods are compared in Figure 3.14. Both ZUPT and our method are significantly better than using standard parameters. This highlight the importance of calibrating the sensors. Our

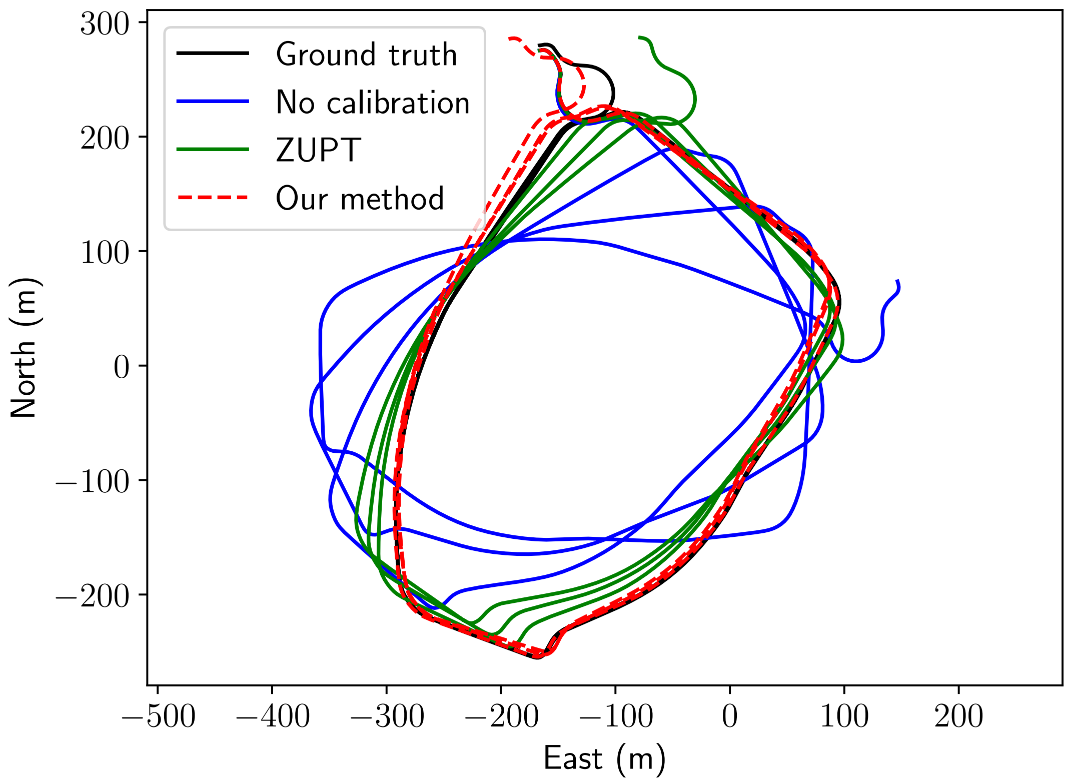


Figure 3.14.: Dead-reckoning results with different calibration methods.

method performs slightly better than ZUPT which is most likely due to the better estimation of the gyro bias.

3.6.5. Iterated estimation

An approach to further improve the parameter estimation is to perform it iteratively on the same trajectory. Indeed, the parameter estimation can only be as good as the state estimates it is using. The first estimated parameters are estimated using a trajectory computed with un-calibrated parameters. It stands to reason that the same trajectory would be estimated more accurately with calibrated parameters. That trajectory can then be used to re-estimate the parameters. The parameter estimation can therefore be performed iteratively using the same calibration trajectory to iteratively improve the parameter estimation.

The result of iterative parameter estimation is shown in Table 3.6. It can be seen that the gyro bias is already well estimated after the first estimation. The following iterations barely improve the estimation with the third iteration producing a slightly worse estimate. While the gyrometer bias was already well estimated, the

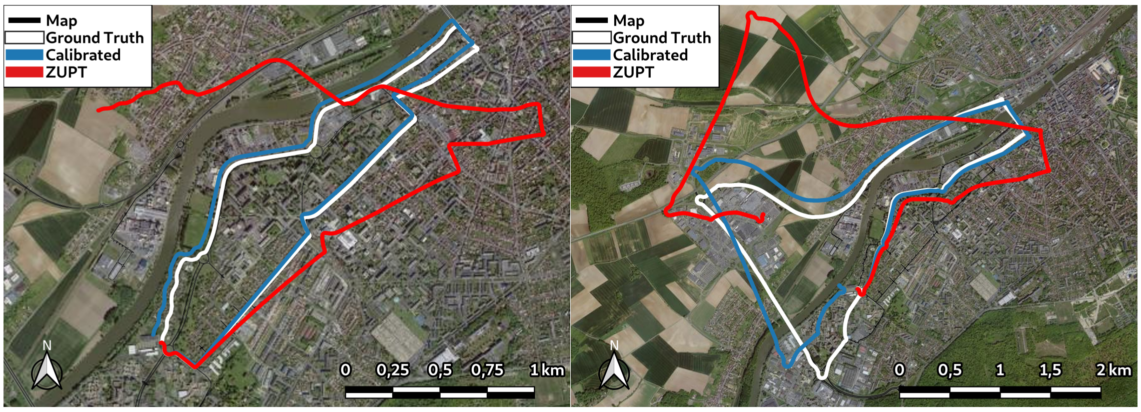


Figure 3.15.: Results of estimated trajectories using dead reckoning on two trajectories

Table 3.6.: Calibration of the parameter using the same trajectory iteratively.

Iteration	b_ω	ρ_{RL}	ρ_{RR}	ρ_{FL}	ρ_{FR}
1	-0.00245	1.9184	1.9174	1.9212	1.9221
2	-0.00253	1.9185	1.9176	1.9215	1.9223
3	-0.00254	1.9184	1.9175	1.9214	1.9222
Ground truth	-0.00250	1.9112	1.91364	1.9150	1.9159

wheel circumferences could be further improved. Across the iterations the wheel circumferences only change by tenth of a millimeter. There still remains an error of several millimeters in their estimate. Hence, a single iteration using the smoothed trajectory already enables to reach the best parameters. To improve calibration additional more accurate and redundant sensors would be required.

It can also be noted that the estimation of wheel circumferences using the new vehicle (which provides wheel speeds) is worse than that obtained on the old vehicle (which provides wheel ticks). In particular, the front wheel circumferences in Table 3.4 are closer to the ground truth estimates than those of Table 3.6. This could be explained by the fact that wheel ticks are used in the estimation with a higher amplitude noise model. Hence, wheel ticks contribute only partially to the estimation of the vehicle speed (which the wheel circumferences influence most). By providing pose observation, the GNSS receiver also enables the Kalman filter to estimate the vehicle speed. The lower noise model of wheel speed measurements might cause the speed estimates to be too directed by these observations. Using more accurate position observations might alleviate this problem but more testing is required.

Table 3.7.: Comparison of parameters estimated using Kalman filtered and Kalman smoothed trajectories

	b_ω	ρ_{RL}	ρ_{RR}	ρ_{FL}	ρ_{FR}
Filtering	-0.00002	1.9180	1.9178	1.9209	1.9225
Smoothing	-0.00245	1.9184	1.9174	1.9212	1.9221

3.6.6. Comparison of Filtering and Smoothing

This work uses trajectories obtained from Kalman smoothing to estimate the calibration parameters. Smoothing the trajectory estimated using Kalman smoothing has some computational cost, albeit small. The necessity of this step should therefore be studied.

Table 3.7 compares the results of parameter estimation using the trajectory estimated with Kalman filtering directly and the ones obtained with the same trajectory smoothed. The gyrometer bias estimate clearly shows that smoothing produces better estimates. Using Kalman filtering only, the gyrometer bias cannot be estimated. A meaningful bias is only obtained using Kalman Smoothing. Because the speed and yaw rate have an unknown evolution model, their associated noises are chosen high such that their values are driven by the observation. Hence, biased observations result in biased state estimates. With the sensor set used in these experiments, the GNSS orientations are the only direct observations of the vehicle yaw. Because the evolution model links the yaw and yaw rate, during the filtering their errors become correlated. This enables Kalman filtering to improve the yaw rate estimate when an observation of the yaw is obtained. However, because of the high noise model put on the yaw rate, this improvement is only effective at the times when GNSS observations are available. Their contribution on other states is entirely drowned by the noise model of the evolution. Smoothing addresses this issue. The improved yaw estimates obtained from the GNSS observation are propagated backward to other states leading to overall better yaw estimate. This in turn leads to better yaw rate estimates and therefore results in better parameter calibration.

3.7. Conclusion

Calibration is essential to accurate localization for intelligent vehicles. The systematic errors caused by an un-calibrated system are detrimental to the localization accuracy. Moreover, DR needs to be reliable in emergency situations as it is the only system relying only on proprioceptive sensors. For that reason, vehicles need to be calibrated regularly. Wheel circumferences do not change quickly but still need to be updated regularly. Although gyrometer biases have been found to be relatively stable, they do change by about 100 μrad/s in a day.

Using Kalman smoothing, the proposed approach manages to estimate the calibration parameters affecting the proprioceptive sensor observation model. This approach does not rely on specific trajectories and does not increase the cost of localization. Instead, the method works in parallel of the localization system.

Using proper calibration, the drift of the dead reckoning system is reduced. This enables the localization system to rely more on DR. This in turn enables the localization to be more tolerant to delays and to use processes leading to more delays.

This contribution has been presented at the International Conference on Robotics and Automation (ICRA) [Welte et al., 2019a].

Further research is needed to evaluate the influence of the length of the calibration trajectory on the parameter estimation. Parameters, such as the gyrometer bias, that have a strong influence on the state estimation should be easier to estimate. These parameters could be estimated with a few seconds of trajectory while others might require more estimates.

Additionally, it would be interesting to hybridize the estimation with a good IMU. The vehicle does have accelerometers (from the ESC) but their low accuracy has made them impractical to use for localization.

4. Map-Aided Localization with a Spatio-Temporal Matching

Contents

4.1. Introduction

Localization using GNSS has known recent advances (e.g. Precise Point Positioning [Domínguez Tijero et al., 2018]), however, it does not yet yield, on its own, the accuracy required by intelligent vehicles [Tijero et al., 2019]. Moreover, its accuracy can be affected by the environment (multipath and non-line-of-sight measurements) [Jiang et al., 2019]. Adding dead-reckoning to GNSS improves significantly the performance of the localization but may still not be sufficient for autonomous driving. To reach a satisfying accuracy, methods using maps have shown some success [Tao et al., 2017, Bürki, Mathias et al., 2019, Fouque et al., 2008, Ghallabi et al., 2019b].

Vehicles are starting to get equipped with cameras and lidars that give the vehicle the capacity to perceive its environment. Beyond the usefulness of these sensors to detect moving objects (pedestrian, vehicles), they can also be used to obtain a local representation of the vehicle surroundings. Hence, by comparing this perceived representation with mapped information, localization can be achieved.

Using maps for localization brings challenges that other sensors do not suffer from. Localizing the vehicle within a map, using the observations obtained from perception sensors, requires a matching step to be performed. Matching, that is, finding correspondences between measurements and map features, is challenging. Indeed, state estimation errors combined with maps numerous candidate features often make the matching ambiguous. Maps often contain thousands of features that can lead to measurements. Even in a local vicinity around the vehicle, tens of features are potential matching candidates. The problem is all the more difficult as maps are now often provided by third parties (e.g. HERE, TomTom, Baidu) rather than self-built using SLAM techniques.

Several strategies exist to deal with this problem. Using robust estimation schemes, unreliable matching can be dealt with at the estimation stage. This can be done by adding a Fault Detection and Exclusion (FDE) step before using the observations in the estimation [Al Hage et al., 2020]. With this method the matching system does not have to produce perfect matches. In this work, a different approach is used. Instead of dealing with errors at the estimation stage, the problem is dealt with at the matching stage. The matches are chosen carefully to limit the risk of incorrect matches. The two approaches, are of course, not mutually excursive and should both be used in a complete localization system.

In this work, the matching problem using traditional matching strategies is improved using prior processing. The matching relies on buffers of observations and state estimates to increase the details of the representation of the world available when matching. Smoothing and adjustment of the state estimates used to perform the matching is also used to reduce the influence of bad localization.

In the next section, different matching methods used in the literature are presented. The measurements, lane markings, road signs and crosswalks, used in this work are then detailed. The matching strategy performing an adjustment step is finally explained and experimental results of the method are presented.

4.2. State of the Art of Matching Methods

4.2.1. Traditional Matching Strategies

Traditional matching strategies use distances between measurements and features to choose the correct matches. In this work, the Mahalanobis distance is used as it accounts for the measurement uncertainty. Three matching strategies are detailed next and a comparison of matching results is shown in Figure 4.1.

4.2.1.1. Nearest Neighbor (NN)

Nearest neighbor association is widely used to associate high level observations to map features [Ghallabi et al., 2019a, Tsuchiya et al., 2019, Leonard et al., 1992].

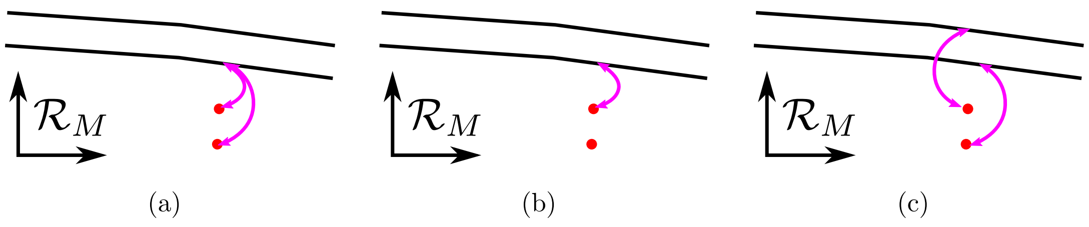


Figure 4.1.: Comparison of matches selected using traditional matching methods (assuming the measurements have similar uncertainties). The two black lines are the map features (lane markings) and the two red dots correspond to measurements. (a) shows nearest neighbor matching (NN), (b) shows unique nearest neighbor matching (UNN), (c) shows Hungarian matching (HG).

To account for the measurement uncertainty and the propagated uncertainty of the state estimates, the nearest neighbor association is performed using Mahalanobis distances rather than Euclidean distances. It is also useful for rejecting unlikely associations.

Hence, an observation $\boldsymbol{z}_j^{(k)}$ is associated with the map feature $\boldsymbol{m}_i$ of the map M such that

$$\boldsymbol{m}_i = \underset{\boldsymbol{m}_i \in \boldsymbol{M}}{\arg\min} \left(\sqrt{\boldsymbol{y}_{i,j}^\top \boldsymbol{S}_{i,j}^{-1} \boldsymbol{y}_{i,j}} \right) \tag{4.1}$$

where $\boldsymbol{y}_{i,j}$ is the residual of observation $\boldsymbol{z}_j^{(k)}$ if matched to feature $\boldsymbol{m}_i$,

$$\boldsymbol{y}_{i,j} = \boldsymbol{z}_j^{(k)} - h_i\left(\hat{\boldsymbol{x}}_k\right) \tag{4.2}$$

and $\boldsymbol{S}_{i,j}$ its covariance matrix,

$$\boldsymbol{S}_{i,j} = \boldsymbol{H}_i \boldsymbol{P}_k \boldsymbol{H}_i^\top + \boldsymbol{R}_j \tag{4.3}$$

$\hat{\boldsymbol{x}}_k$ is the estimate of the vehicle state at time k, $\boldsymbol{H}_i$ the Jacobian of h_i, $\boldsymbol{P}_k$ and $\boldsymbol{R}_j$ the covariance matrices of respectively the state estimate and the measurement.

With this method, the measurements are associated with the nearest (uncertainty weighted) feature. To prevent matches to be too far from each other, a maximum value for the association is set. As for the gating step seen in Subsubsection 2.3.3.2 on page 27, the maximum distance is set using a χ^2 test with a risk α of rejecting good matches,

$$\boldsymbol{y}_{i,j}^\top \boldsymbol{S}_{i,j}^{-1} \boldsymbol{y}_{i,j} < F_{\chi^2}^{-1}\left(1-\alpha, d_{\boldsymbol{z}_k}\right) \tag{4.4}$$

This test is used in all subsequent matching methods.

4.2.1.2. Unique Nearest Neighbor (UNN)

The previous matching allows a single feature to be associated with multiple measurements. This may be useful if the detection system has a tendency to over-segment. In most applications with state-of-the-art detection systems, over-segmentation either rarely occurs or is dealt with before providing the measurements. As such the matching can be performed under the assumption of one-to-one associations between measurements and features. To achieve this, the previous method can be modified in order to remove matches that would result in multiple measurements matched to the same feature.

Thus in the case where several observations are associated with the same feature, only the one with the shortest Mahalanobis distance is kept. The other matches are discarded from the matching process. These observations will not have corresponding features and will therefore not be used for the state estimation.

4.2.1.3. Hungarian method (HG)

The Unique Nearest Neighbor can either match an observation with its closest feature, or not match the observation at all (if the closest feature is closer to another observation). Munkres algorithm [Munkres, 1957] performs a global matching considering all observations at time k. Like the UNN method, it only allows features to be associated with a single observation. However, it does not necessarily associate an observation with its closest feature. The method finds matches that minimize the summed Mahalanobis distances of all selected matches [Kaess et al., 2008]. Hence, at time k for an observation set $\boldsymbol{Z}^{(k)}$ of size J, it will find the subset $\boldsymbol{M}_k \subseteq \boldsymbol{M}$ containing J features so that

$$\boldsymbol{M}_k = \underset{\boldsymbol{M}_k \subseteq \boldsymbol{M}}{\arg\min} \left(\sum_{\boldsymbol{m}_i \in \boldsymbol{M}_k} \sqrt{\boldsymbol{y}_{i,j}^{\top} \boldsymbol{S}_{i,j}^{-1} \boldsymbol{y}_{i,j}} \right) \tag{4.5}$$

where $\boldsymbol{m}_i$ is the feature associated with the observation $\boldsymbol{z}_j^{(k)}$.

As such the Hungarian allows measurements to be associated to any feature if it decreases the overall matching cost. Although the traditional Hungarian method always associates every observation to a feature (assuming enough features are available), in most practical cases a match is considered valid if the matching distance is small enough. To account for this, the distances $\sqrt{\boldsymbol{y}_{i,j}^{\top} \boldsymbol{S}_{i,j}^{-1} \boldsymbol{y}_{i,j}}$ are saturated to a maximum value. Doing so results in observations far from all features to have the same association cost for all features. Therefore such observation can be matched to any feature. Not accounting for this, an observation far from any feature would be matched to its closest feature to reduce the overall cost. Saturating the distance enables this observation to be matched to any feature leaving other observations to be matched with their corresponding features.

4.2.2. SLAM-oriented approaches

Simultaneous Localization And Mapping (SLAM) domain led the development of a variety of approaches pertinent for the matching problem addressed in this work. Indeed, a mandatory step in SLAM consists in matching current observations with previously established estimates of environment features.

4.2.2.1. Multi-Hypothesis Estimation

When the matching cannot be performed with enough confidence, one approach is to compute solutions for all potential matches. When two matching candidates are possible, the two solutions are computed. The two solutions represent two matching hypotheses that cannot be discarded at the time of the estimation. With new observations one of the two hypotheses will become clearly wrong while the other will hopefully stay relevant. The wrong hypothesis can then be discarded. This process can happen every time there is a matching ambiguity. Thus, the number of hypotheses that need to be tracked can grow quickly as a new ambiguity doubles the number of hypotheses.

[Hsiao and Kaess, 2019] have extended the iSAM estimation strategy to account for such ambiguous associations. While standard iSAM relies on a factor graph composed of nodes (variables) linked by factors (function linking the variables). MH-iSAM adds multi-hypothesis factors. These factors essentially consist of two potential standard factors. By adding such factors, multi-hypothesis nodes need to be added as well. Depending on whether the estimation is performed by taking one hypothesis or the other, the estimates will differ. Nodes therefore become MH-nodes when multiple solutions need to be stored. Every time a new MH-factor is added, the number of potential hypothesis doubles. In practice, this is not the case as some hypothesis can be incompatible with each other. A hypothesis tree is created as MH-factors are added to track potential hypothesis. The branch of the tree can be eliminated when hypotheses incompatible with each other are found.

With this strategy, MH-iSAM is able to deal with multiple hypothesis efficiently. As the computational complexity can double each time a new hypothesis is added, it is more computationally expensive than the standard iSAM. It is mainly aimed at handling ambiguous loop closures in SLAM problems rather than ambiguous matching of specific observations to features.

4.2.2.2. Multimodal Noise Models

In most state estimation strategies, the noises affecting the system are assumed to follow Gaussian distributions. When there are multiple potential hypothesis, one way to represent these hypotheses is through multimodal distributions. Researchers have extended the standard estimation strategies to non-Gaussian distribution to deal with heavy tailed distributions or non-symmetric ones [Rosen et al., 2013]. Instead of considering multiple hypothesis as different factors following each Gaussian distribution, they can also be represented as a single factor with a multimodal noise model.

[Olson and Agarwal, 2013, Doherty et al., 2019] have extended iSAM2 techniques to multimodal noise models through a clever approximation. Although multimodal noise models do not enable the same efficient iterative estimation used by iSAM, approximating a multimodal distribution by whichever mode is maximum, does. With this approximation, multimodal problem can be solved efficiently with iSAM. The mode which is not maximal at the current linearization point is not irrelevant as during the estimation, other measurements might shift the state estimate thus shifting the linearization point where another mode becomes maximal. This, however, has the disadvantage of resulting in the most recent estimate to heavily depend on maximal mode. The most recent estimate does not greatly benefit from this strategy. The main benefit comes to past estimates which have added enough measurements to stabilize their linearization point. Hence this method only accounts for multiple hypothesis in the long run. For autonomous driving applications where the real-time estimate is the most important, this method is not suited. Safety critical applications cannot afford to pick whichever mode is maximum and wait for new measurements to be added to stabilize the estimate. The matches that are used for estimation need to be accurate when they are used. It would be safer not to perform matching and wait for the ambiguity to be resolved.

4.2.3. Combined Constraint Data Association (CCDA)

[Dube et al., 2017] have proposed a matching strategy using graph cliques to find globally consistent matches. A graph is built by adding a node to the graph for every pair $\left(\boldsymbol{z}_j^{(k)}, \boldsymbol{m}_i\right)$ of measurement and feature that are candidates. The candidates are found using vector descriptors obtained from point cloud clusters. Each node (candidate matches) are connected to each other if the distance $d_{\boldsymbol{z}} = \left\|\boldsymbol{z}_{j_X}^{(k)} - \boldsymbol{z}_{j_Y}^{(k)}\right\|^2$ between the observations of one match $X : \left(\boldsymbol{z}_{j_X}^{(k)}, \boldsymbol{m}_{i_X}\right)$ and the other $Y : \left(\boldsymbol{z}_{j_Y}^{(k)}, \boldsymbol{m}_{i_Y}\right)$ is the same (within a predefined interval) as the distance $d_{\boldsymbol{m}} = \|\boldsymbol{m}_{i_X} - \boldsymbol{m}_{i_Y}\|^2$ between the features of one match and the other. Therefore, graph edges are added for every two pairs $\left(\boldsymbol{z}_{j_X}^{(k)}, \boldsymbol{m}_{i_X}\right)$ and $\left(\boldsymbol{z}_{j_Y}^{(k)}, \boldsymbol{m}_{i_Y}\right)$ which verify $|d_{\boldsymbol{z}} - d_{\boldsymbol{m}}| < \epsilon$ (with ϵ a chosen threshold).

Once the graph is built, the matches are selected by finding the biggest clique (fully connected nodes). The biggest clique corresponds to the highest number of matches that are consistent with each other (in terms of measurement and feature distances). The matches of the clique are finally used for the state estimation.

This method is applied by the author in a context of place recognition with two-dimensional measurements. In this work measurements of different dimensions: lane markings (one-dimensional) and road signs (two-dimensional) are used. The distance between those two types of features depends on the vehicle orientation. Moreover as the state estimate uncertainty is not considered, ambiguities would arise as the width of lanes is fairly similar for all roads and little information is available to filter out candidate matches (no description vector is available).

This method is interesting for its use of relative distances between a set of observations and a set of potential features. Indeed, information on relative distance is usually under-exploited in the matching process, though it can enable discarding wrong matches.

In the following work, traditional matching strategies are used. The method presented in this thesis aims at improving the quality of the matches obtained with these methods.

4.3. Observation Models of Features Georeferenced in Maps

The concept of map is understood here as prior knowledge about the road network with georeferenced features. This section details the observation models a vehicle can use for localization when it is equipped with exteroceptive sensors (e.g. cameras and lidars) that are able to detect features stored in the map. In this section, we are interested in painted side markings on pavements and road signs. These are quite easy to obtain with optical sensors and their information is often recorded in the map.

We were also interested in the detection of crosswalks with a lidar. The results of this study are presented in Appendix E. They are not detailed in this chapter as they provide very infrequently and less accurate measurements than measurements on road signs.

4.3.1. High Definition Maps

Digital maps become a constitutive element of any vehicle oriented applications, from almost any passenger car, to autonomous vehicle prototypes, including smartphone applications dedicated to navigation.

Data included in maps as well as the format according to which this data is organized vary substantially depending on the targeted usage. However, in the intelligent vehicle domain, vector maps are widely adopted. In contrast with raster maps in which geographical data is organized in geo-referenced grids, vector maps contains road features individually described with its position, attributes and relationships [Buckley, 1992]. Large coverage vector maps commercially available, take the road as elemental to describe the road network. Attributes are associated to roads (e.g. number of lanes, speed limits, driving direction). This is well suited for human guidance and navigation assistance applications, however, highly automated and autonomous vehicle applications require a more detailed description. So-called High Definition maps (HD maps) therefore take driving lane as elementary geometry.

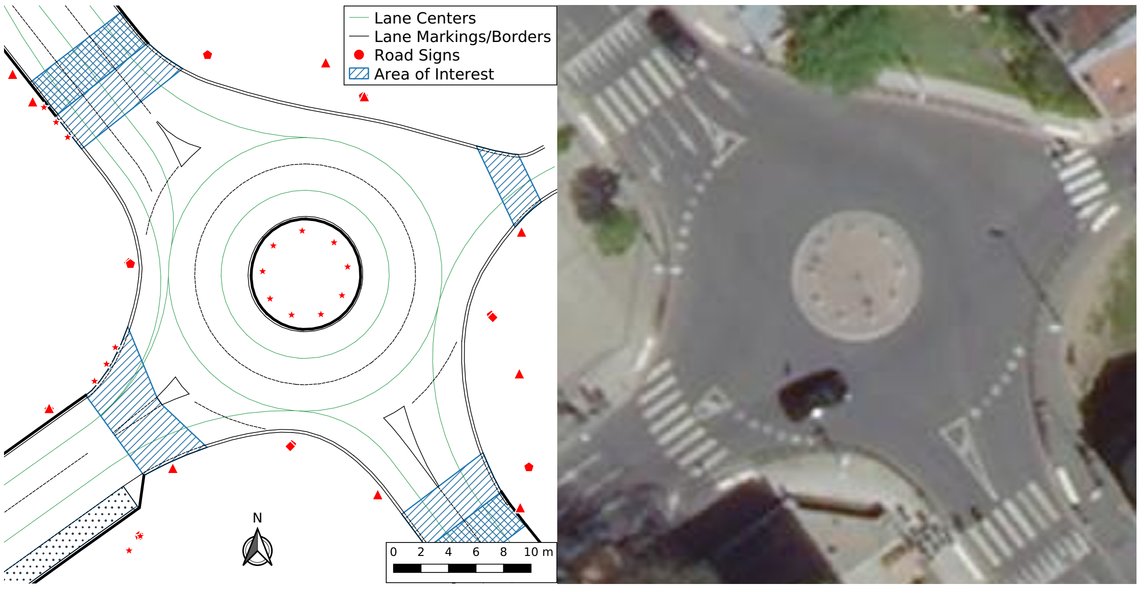
Map aided localization requires a map with referenced information. The measurements can then be compared to the map to improve localization. The techniques used for map-aided localization vary greatly as existing maps are diverse. Using coarse maps of the road geometry every satellite navigation system can match an imprecise position to a road. In this instance, the goal is not so much to improve localization but rather provide the user with a smooth display of the localization.

HD maps are not a clearly defined type of maps in the literature. They themselves differ greatly depending on their application and building method. Among HD maps, two categories can be identified. Dense maps record points or pixels irrespective of what the points represent. These maps represent the world in a low-level representation with little understanding or consideration for specific features. The size of such maps make them hard to scale to large areas. Vector maps on the contrary do not store points but rather specific features that are useful. Each feature can have attributes describing it in more detail and relational information can also be saved in the map. Unlike dense maps, vector maps are much lighter therefore do not suffer from scalability issues. However, they require processing to extract features, that can often still require human intervention and can result in mapping errors. In both cases the maps need to be referenced with centimeter levels of accuracy to be useful for localization. In this work, HD vector maps are used.

Many formats exist to store vector maps. Here, maps are stored in Spatialite database. Spatialite consists of an SQL database with additional features useful to retrieve geographic data. Each type of feature is contained in tables which can be linked with themselves (e.g. lane centers link to next and previous lanes) or with others (lane markings refer to the lane center along which they are). Spatialite adds to SQL, a spatial indexing feature that enables quick retrieval of information based on areas of interest. For instance, Spatialite enables to retrieve data locally within a given radius around the vehicle.

Spatialite as most vector map databases handles three main types of geometry that are used to encode the features:

- points: two or three dimensional
- polylines: ordered series of points
- polygons: connected points forming a close shape



(a) (b)

Figure 4.2.: (a) Visualization of the map of the roundabout near the Benjamin Franklin building of the University of Compiègne. (b) Aerial view of the same area provided by the *Institut géographique national (IGN)*.

Each element of the map is described using these elements with additional attributes. Links describe the geometry of each of the lanes of the road with polylines (in green on Figure 4.2 (a)). They also record attributes such as maximum allowed speed, lane width, whether the line is within a crosswalk or a speed hump. Link borders describe using polylines the lane markings, curbs, stop lines, etc. (in black on Figure 4.2 (a)). They also contain identifiers for the type of feature (full, dashed, curb, etc.).

The road signs are also referenced in the map using points (in red on Figure 4.2 (a)). As for link borders, they also contain identifiers to the type of sign.

Some areas are also referenced in the map. Areas of interest are referenced using polygons. They identify areas containing pedestrian crossings, speed humps, parking spaces (in blue on Figure 4.2 (a)). The map contains additional features useful for the navigation and control systems that are not detailed here.

4.3.2. Side Lane Markings using Vision

The road networks in most parts of the word are delimited by clearly identifiable features. Roads are typically built out of asphalt which contrasts with the rest of the ground. Roads are also often delimited by curbs and/or lane markings. Other lane markings also separate the roads in different lanes or are used to signal stop lanes, crosswalks, etc. These features provide important information for localization.

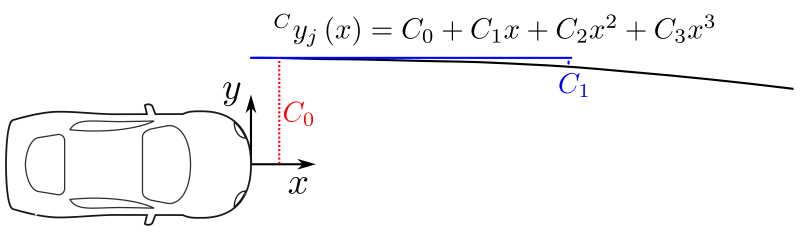


Figure 4.3.: Polynomial representation of detected lane markings returned by a Mobileye camera. The four parameters, C_0, C_1, C_2 and C_3, returned for each marking describe the polynomial ${}^{C}y_j\,(x) = C_0 + C_1 x + C_2 x^2 + C_3 x^3$ where x is the distance in front of the vehicle.

Although road features are fairly sporadic along their track, there are numerous features that can be detected across the road track.

Car manufacturers have started to equip vehicles with sensor able to detect such features for lane crossing warnings or even lane keeping systems. Cameras provide an affordable solution to detect these features leading to off the shelves smart camera systems being installed on vehicles.

4.3.2.1. Camera-based Lane Detection

Smart cameras are nowadays equipping every middle and high grade passenger vehicles. In this thesis, a Mobileye smart camera was used to detect lane markings. Mounted on the windshield and behind the rear mirror, it takes pictures of the road in front and provides a description of the road markings thanks to internal image processing. Lane markings are described using polynomials as illustrated in Figure 4.3. Up to four polynomials can be transmitted by such a sensor. Additionally, the sensors provide information on the feature type, width and color (although this is not relevant for French roads).

The sensors being a black box from which the input image used for the detection is not even available it raises some challenges. The sensor does not only perform the detection but also the projection in a vehicle frame. This results in projection that can be inaccurate as roads are not flat but often curved. Another issue is that, although it provides the lateral distance to the lane markings, this distance cannot be seen by the camera. The sensor therefore performs a tracking of the detection combined with vehicle odometry to provide this information. This results in observations bound to be temporally correlated which needs to be considered when the fusion is performed.

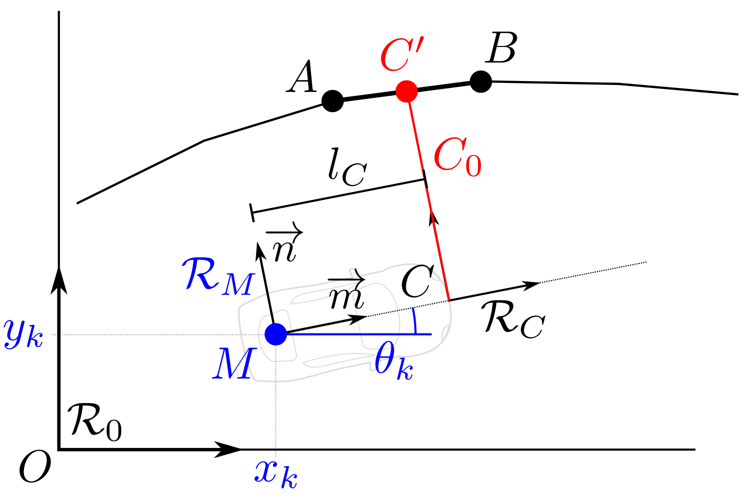


Figure 4.4.: Graphic of a lane marking observation.

4.3.2.2. Observation Model

Despite the sensor providing a third degree polynomial, only the first parameter C_0 will be used for localization. C_1 is accurate enough in most situations to be used but it has been observed to have unacceptable errors in particular when turning at an intersection. It will only be used to filter out potential matches rather than being directly used in the estimation.

The observation model of C_0 can be derived from geometric considerations. Figure 4.4 illustrates the notations used. $\mathcal{R}_M$ denotes the vehicle body frame, centered on M and oriented as the vehicle. The set of vectors $\overrightarrow{m}$, $\overrightarrow{n}$ forms a base of this frame, with $\overrightarrow{m}$ pointing to the front of the vehicle and $\overrightarrow{n}$ pointing to its left side. $\mathcal{R}_C$ is the sensor frame, centered on the front bumper at point C and oriented as $\mathcal{R}_M$ (see Subsection A.1.1 for a full description of each frames used). The observed segment is defined using points A and B whose coordinates $\begin{bmatrix} {}^0x_A & {}^0y_A \end{bmatrix}^\top$ and $\begin{bmatrix} {}^0x_B & {}^0y_B \end{bmatrix}^\top$ are obtained from an HD map. The point C' represents the point measured on segment AB such that $\overrightarrow{CC'} = C_0 \cdot \overrightarrow{n}$.

The observation model of C_0 can be obtained by realizing that $\overrightarrow{OC'}$ can be expressed in two ways, as shown by Equation 4.6 and Equation 4.7, either using a point of the map (i.e. A) as intermediary or using points of the vehicle (M and C).

$$\overrightarrow{OC'} = \overrightarrow{OA} + \overrightarrow{AC'} \tag{4.6}$$

$$\overrightarrow{OC'} = \overrightarrow{OM} + \overrightarrow{MC} + \overrightarrow{CC'} \tag{4.7}$$

Thus C_0 can be found using the scalar product such that,

$$\begin{aligned} C_0 &= \overrightarrow{CC'} \cdot \overrightarrow{n} && (4.8)\\ &= \left(\overrightarrow{OA} + \overrightarrow{AC'} - \overrightarrow{OM} - \overrightarrow{MC}\right) \cdot \overrightarrow{n} && (4.9)\\ &= \overrightarrow{MA} \cdot \overrightarrow{n} + \overrightarrow{AC'} \cdot \overrightarrow{n} - \underbrace{\overrightarrow{MC} \cdot \overrightarrow{n}}_{=0} \qquad = \overrightarrow{MA} \cdot \overrightarrow{n} + \overrightarrow{AC'} \cdot \overrightarrow{n} && (4.10) \end{aligned}$$

When the vehicle is perfectly aligned with the observed marking, $\overrightarrow{AC'}$ and $\overrightarrow{n}$ are orthogonal therefore the observation model is simply

$$C_0 = \begin{bmatrix} x_A - x_k \\ y_A - y_k \end{bmatrix} \cdot \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix}. \tag{4.11}$$

In practice, there is always some level of misalignment thus an expression for $\overrightarrow{AC'} \cdot \overrightarrow{n}$ is needed. To obtain this expression, $\overrightarrow{AC'}$ can be expressed relative to $\overrightarrow{AB}$ such that $\overrightarrow{AC'} = \lambda\overrightarrow{AB}$, with λ a scaling factor. Therefore $\lambda = 0$ implies $C' = A$ and $\lambda = 1$ implies $C' = B$. This variable also enables to discriminate whether or not the measurements can be associated with a given segment since λ has to belong to $[0, 1]$. From this definition and by following vector decomposition rules, it follows that

$$\lambda\overrightarrow{AB} = \overrightarrow{AC'} = \overrightarrow{AC} + \overrightarrow{CC'} \tag{4.12}$$

This expression links λ to $\overrightarrow{AB}$ and $\overrightarrow{AC}$, which can be known from the map and the vehicle state. However, it also involves $\overrightarrow{CC'}$ which directly depends on the observation. To eliminate this term, the scalar product with the vector $\overrightarrow{m}$ is used as $\overrightarrow{m}$ is orthogonal to $\overrightarrow{CC'}$.

$$\begin{aligned} \lambda\overrightarrow{AB} \cdot \overrightarrow{m} &= \left(\overrightarrow{AC} + \overrightarrow{CC'}\right) \cdot \overrightarrow{m} && (4.13)\\ \lambda\overrightarrow{AB} \cdot \overrightarrow{m} &= \overrightarrow{AC} \cdot \overrightarrow{m} && (4.14)\\ \lambda &= \frac{\overrightarrow{AC} \cdot \overrightarrow{m}}{\overrightarrow{AB} \cdot \overrightarrow{m}} && (4.15) \end{aligned}$$

Hence, λ only exists when the segment AB and the vehicle are not orthogonal which is the case in practice as the sensor only provides measurements somewhat aligned with the vehicle. Using this expression of λ, the observation model obtained in Equation 4.10 becomes

$$\begin{aligned} C_0 &= \overrightarrow{MA} \cdot \overrightarrow{n} + \lambda\overrightarrow{AB} \cdot \overrightarrow{n} && (4.16)\\ &= \overrightarrow{MA} \cdot \overrightarrow{n} + \frac{\overrightarrow{AC} \cdot \overrightarrow{m} \times \overrightarrow{AB} \cdot \overrightarrow{n}}{\overrightarrow{AB} \cdot \overrightarrow{m}} && (4.17) \end{aligned}$$

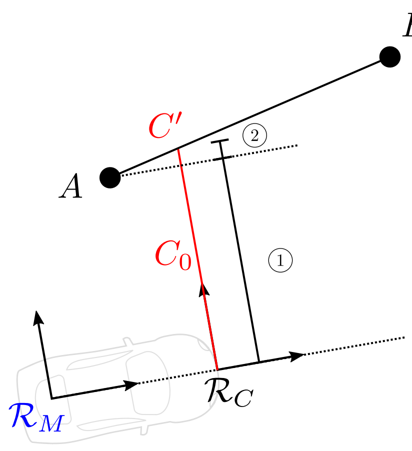


Figure 4.5.: Visualization of the terms of the lane marking observation model. The first term is the lateral distance to the marking not accounting for the relative angle between the marking and measured segment. The second term accounts for this relative angle.

By expressing the scalar products using the vector coordinates, we obtain the observation model,

$$C_0 = h^{LM}(\boldsymbol{x}_k) \tag{4.18}$$

$$= \begin{bmatrix} x_A - x_k \\ y_A - y_k \end{bmatrix} \cdot \begin{bmatrix} -\sin\theta_k \\ \cos\theta_k \end{bmatrix} + \frac{\left(\begin{bmatrix} x_k - x_A \\ y_k - y_A \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} + l_x \right) \begin{bmatrix} x_B - x_A \\ y_B - y_A \end{bmatrix} \cdot \begin{bmatrix} -\sin\theta_k \\ \cos\theta_k \end{bmatrix}}{\begin{bmatrix} x_B - x_A \\ y_B - y_A \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix}} \tag{4.19}$$

This model is equivalent to other forms found in the literature [Tao et al., 2017]. The aforementioned formulation is preferred as it preserves a geometrical interpretation. As shown in Figure 4.5, the lane marking observation is composed of two parts. The first part is the distance to the segment not accounting for misalignment between the segment and the vehicle. The distance is expressed by the first term of Equation 4.19. The second distance accounts for the misalignment between the marking and the vehicle (second term of Equation 4.19).

4.3.2.3. Camera Calibration

Although the smart camera used in this work has been installed and calibrated by the manufacturer, the experimental nature of such sensors forces us to further calibrate the sensor output. Despite the camera providing the lateral distances to markings directly in a horizontal frame bound to the vehicle, theses distances have not been found to be perfectly calibrated. By comparing the measurements with the

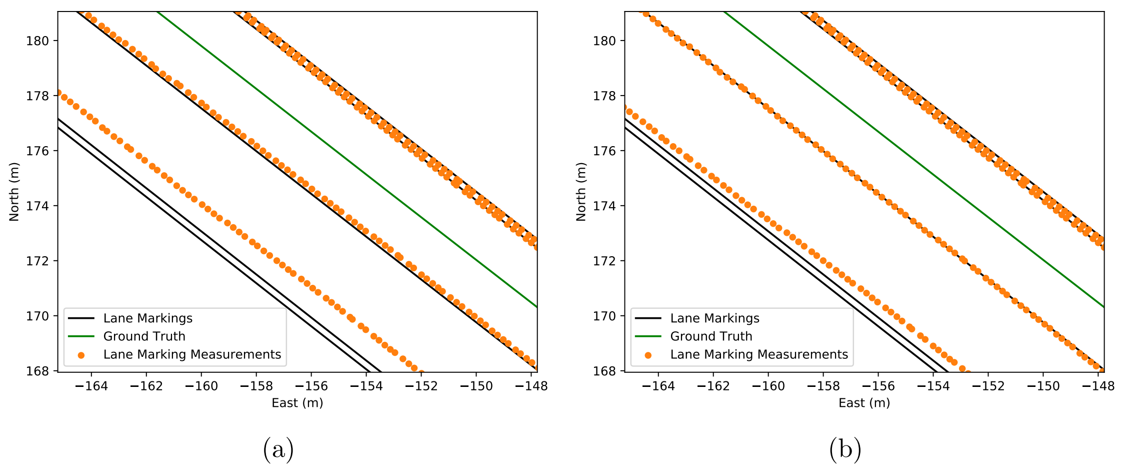


Figure 4.6.: Comparison of lane marking detection with uncalibrated sensors (a) and after calibration (b).

map using ground truth positioning, some markings have been found to be further in reality than their measures would suggest. This would result in a lateral bias in the estimation which might not be dangerous on straight roads but will become relevant when nearing intersections. With our sensor, we have found that marking measurements on the left of the vehicle are systematically shorter than they should be (see Figure 4.6). Hence, we compensate this effect by applying a linear correction to left measurements. In our experiments, the left markings have been found to be about 10% too short. The measurements provided to the localization system are thus C_0 for the right side and $1.1C_0$ for the left side.

4.3.2.4. Variance of the Noise of the Camera

The noise characteristics of the observation model have been studied by comparing real observations to expected ones computed using the true vehicle pose. The detections are matched to their closest map feature (within one meter) using the ground truth states. The observation errors can thus be evaluated by studying the distribution of these errors. Due to the limited field-of-view of the camera, external markings (further on the sides) are detected further away. However the camera returns a polynomial defined from the end of the vehicle front bumper (as shown in Figure 4.3). The camera performs tracking of the markings as the vehicle moves forward to provide the lateral distance to the marking. Since it cannot be directly observed, uncertainty associated with C_0 increases when the lateral distance to the marking (i.e. $|C_0|$) increases. It is difficult to obtain a continuous view to the error depending on $|C_0|$ because lanes have regular width. Therefore, there are many measurements for lane markings directly at the left and right sides of the vehicle that describe the lane in which the vehicle is. There are also measurements of lane markings bordering the left lanes (as most lanes in right-side driving countries have left lanes). However, there is almost no measurement of lane markings in between those three distances or on the right side (as right lanes are rarer). Nevertheless, there is a noticeable dependency of the measurement uncertainty with the observed distances. Although with only three main data point the actual dependency cannot be obtained, we assume in this work that the uncertainty varies linearly with $|C_0|$. This assumption is not critical as most measurements will fall near of the three points anyway. The standard deviation of the lane marking observation is therefore found to be,

$$\sigma_{LM} = 0.1C_0. \tag{4.20}$$

4.3.3. Road Signs Measurements using Lidar

Road networks do not only contain unidimensional boundaries, features such as road signs provide two-dimensional constraints and in some cases an orientation. As such, road signs should enable to greatly improve the localization accuracy. Road signs are easily detectable using a lidar thanks to the intensity measurement. Indeed, road signs are highly retro-reflective as required by European norms [EN 12899-1, 2007]. Therefore, the points hitting a road sign can be isolated using a threshold on the point intensity.

4.3.3.1. Observation Model

Road sign measurements and map features are already in Cartesian coordinates. They only differ by the reference frame in which they are expressed. Road sign measurements $\begin{bmatrix} {}^V x_k^j & {}^V y_k^j \end{bmatrix}^\top$ are obtained in the sensor frame $\mathcal{R}_V$, whereas the referenced feature (the road sign position $\begin{bmatrix} {}^0 x_i & {}^0 y_i \end{bmatrix}^\top$) is known in the global frame $\mathcal{R}_0$. Therefore, the observation model for road signs consists of a frame change,

$$\begin{bmatrix} {}^0 x_i \\ {}^0 y_i \end{bmatrix} = {}^0 h^{RS}\left(\boldsymbol{x}_k\right) \tag{4.21}$$

$$= \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix} \underbrace{\left(\begin{bmatrix} \cos {}^M\theta_V & -\sin {}^M\theta_V \\ \sin {}^M\theta_V & \cos {}^M\theta_V \end{bmatrix} \begin{bmatrix} {}^V x_k^j \\ {}^V y_k^j \end{bmatrix} + \begin{bmatrix} {}^M x_V \\ {}^M y_V \end{bmatrix} \right)}_{\begin{bmatrix} {}^M x_k^j \\ {}^M y_k^j \end{bmatrix}} + \begin{bmatrix} x_k \\ y_k \end{bmatrix} \tag{4.22}$$

where $\begin{bmatrix} {}^M x_V & {}^M y_V \end{bmatrix}^\top$ is the translation between the sensor frame and the mobile frame (see Figure 4.7) and ${}^M\theta_V$ is the rotation between the two frames.

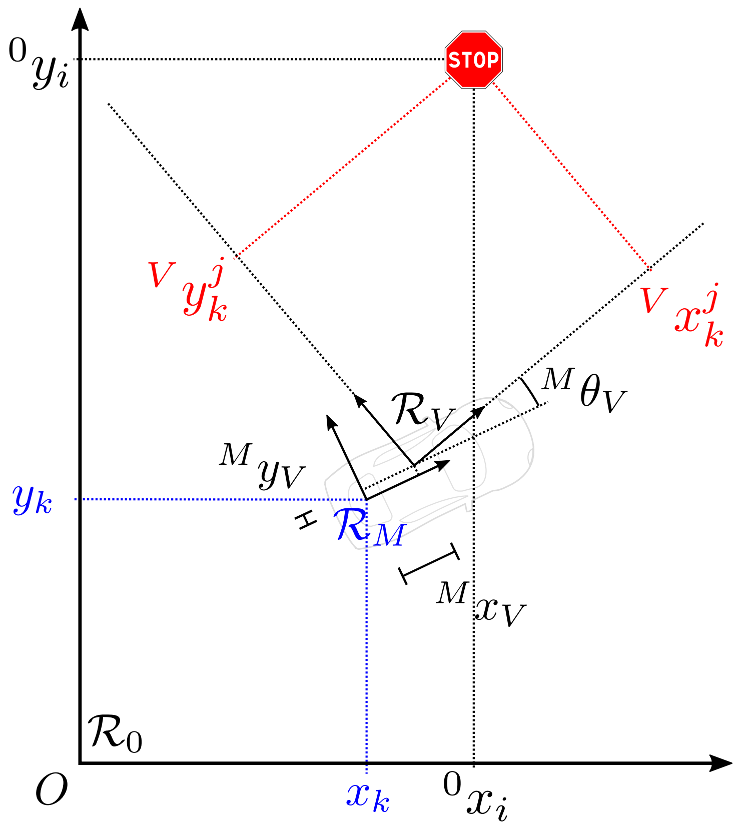


Figure 4.7.: Graphic of a road sign observation. The lidar sensor measures the coordinates of the road signs in its sensor frame $\mathcal{R}_V$, the road sign referenced coordinates are in the global frame $\mathcal{R}_0$

The observation model could also be expressed in the sensor frame $\mathcal{R}_V$ as

$$\begin{bmatrix} {}^V x_{RS} \\ {}^V y_{RS} \end{bmatrix} = {}^V h_i^{RS}(\boldsymbol{x}_k) \tag{4.23}$$

$$= \begin{bmatrix} \cos {}^M\theta_V & \sin {}^M\theta_V \\ -\sin {}^M\theta_V & \cos {}^M\theta_V \end{bmatrix} \left(\begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \begin{bmatrix} {}^0x_i - x_k \\ {}^0y_i - y_k \end{bmatrix} - \begin{bmatrix} {}^M x_V \\ {}^M y_V \end{bmatrix} \right) \tag{4.24}$$

The former is preferred as it produces residuals that are expressed in the same reference frame (the global frame) whereas Equation 4.24 produces residuals expressed in the mobile frame $\mathcal{R}_M$. Hence, the residuals can be compared to each other directly which will be useful for error detection. This is explained more thoroughly in Chapter 5.

4.3.3.2. Sign Detector

Most lidars aimed at automotive applications provide a list of Cartesian points $(x_i,\, y_i,\, z_i)$ with additional information (intensity: i_i). A lidar scan of N points is therefore defined as,

$$\Omega = \{p_i\}_{i \in [\![1,N]\!]} \,.$$

where $p_i = \begin{bmatrix} x_i & y_i & z_i & i_i \end{bmatrix}^\top$. From this scan the points with intensities above the selected threshold I_{min} can be isolated,

$$\Omega_{RS} = \{p_i \in \Omega \,|i_i > I_{min}\}$$

Ω_{RS} contains points of high intensities that correspond to road signs and other highly reflective surfaces. As such Ω_{RS} also contains points from vehicle license plates and metallic surfaces that, with the right incidence angle, can be highly reflective. The remaining points form small clusters. The clusters can be found using the Euclidean clustering algorithm given in [Rusu, 2010]. Each cluster corresponds to a potential road sign. The previous algorithm provides clusters as lists of points. The centroids of these lists of points are estimated to be used as measurements. Therefore for each cluster we compute

$$\begin{bmatrix} {}^V x_k^j \\ {}^V y_k^j \end{bmatrix} = \begin{bmatrix} \dfrac{\max\limits_{p_i \in \Omega_j} x_i + \min\limits_{p_i \in \Omega_j} x_i}{2} \\ \dfrac{\max\limits_{p_i \in \Omega_j} y_i + \min\limits_{p_i \in \Omega_j} y_i}{2} \end{bmatrix}$$

Optionally the orientation of the road sign can be estimated using Principal Component Analysis (PCA).

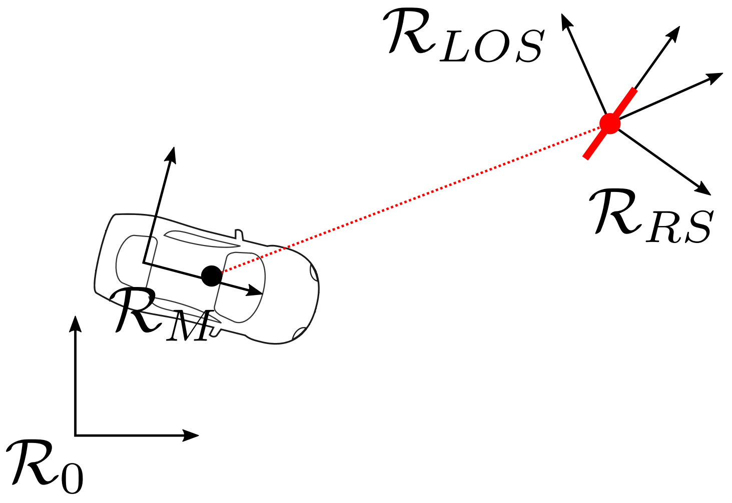


Figure 4.8.: Graphic of frames in which the road sign measurement error has been studied. The global frame $\mathcal{R}_0$ is the main reference frame in which the localization solution and ground truth is obtained. The line-of-sight $\mathcal{R}_{LOS}$ frame is oriented in the direction of the line of sight whereas the road sign frame $\mathcal{R}_{RS}$ is oriented in the direction of the road sign (with its axes orthogonal and along the road sign plate).

4.3.3.3. Variance of the Noise of Road Signs Measurements

To model the noise of the observation, the ground truth states $\check{\boldsymbol{x}}_k$ have been used along with the measurements and observation model ${}^0h^{RS}(\cdot)$. The observation error is computed as,

$$\begin{bmatrix} {}^0e_k^j \\ {}^0e_k^j \end{bmatrix} = {}^0h^{RS}\left(\check{\boldsymbol{x}}_k\right) - \begin{bmatrix} {}^0x_i \\ {}^0y_i \end{bmatrix} \tag{4.25}$$

where $\begin{bmatrix} {}^0x_i & {}^0y_i \end{bmatrix}^\top$ are the geo-referenced coordinates of the map feature corresponding to observation $\boldsymbol{z}_k^j$. As a reminder, the sensor measurement does not appear in this equation because it is part of the observation model ${}^0h^{RS}(\cdot)$ as defined in Equation 4.22.

Several noise models have been studied, because of the sensor and the type of feature, one might expect the noise to be affected by the direction of the line of sign (LOS) or the sign orientations (see Figure 4.8).

The noise model expressed in the global frame is found by computing the covariance matrix of the observation error. We find

$$R = \begin{bmatrix} (0.1146)^2 & -0.00045 \\ -0.00045 & (0.1073)^2 \end{bmatrix} \tag{4.26}$$

The error expressed in the global frame has roughly the same variance along the east and north direction. Also the two dimensions have almost no correlation.

Most road signs are flat. Because of this the accuracy of the road sign detection should be better along the depth of the sign rather than along its width. Although the road sign orientation is not stored in the map, it can be estimated for each cluster using Principal Component Analysis (PCA). The median is used to find the best estimated orientation θ_i^{RS} of each observed road signs. Using these orientations, the road sign oriented error is computed in the frame of each road sign (RS) as

$$\begin{bmatrix} {}^{RS}e_k^j \\ {}^{RS}e_k^j \end{bmatrix} = \begin{bmatrix} \cos\theta_i^{RS} & \sin\theta_i^{RS} \\ -\sin\theta_i^{RS} & \cos\theta_i^{RS} \end{bmatrix} \begin{bmatrix} {}^0e_k^j \\ {}^0e_k^j \end{bmatrix}. \tag{4.27}$$

The distribution of the errors is shown in Figure 4.9. Although there are some points with higher errors along the road sign direction, this is not the case for most points. This is confirmed by the covariance matrix obtained in this frame,

$${}^{RS}R = \begin{bmatrix} (0.1170)^2 & 0.00009 \\ 0.00009 & (0.1075)^2 \end{bmatrix} \tag{4.28}$$

From the estimated covariance matrix, no significant difference can be found between the error distribution along the depth of the road sign and across its width. Hence larger errors are not found in the road sign direction in most cases.

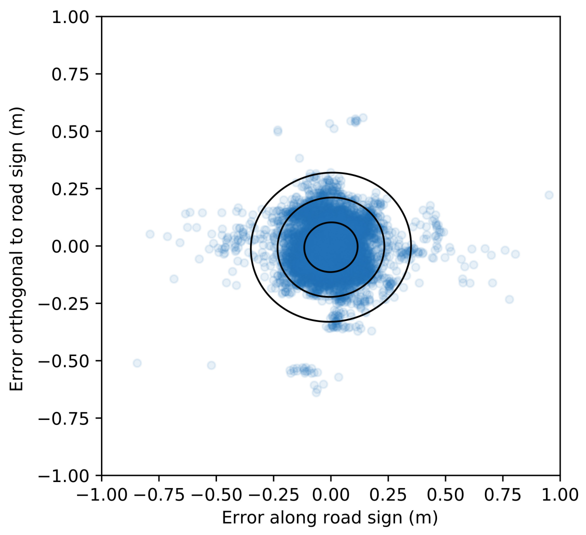


Figure 4.9.: Distributions of road sign errors in the Road Sign (RS) frame. In black are represented the 1σ, 2σ and 3σ ellipses.

As the lidar produces points using a rotating laser, one might expect the uncertainty of the point position to be different in the Line-Of-Sight (LOS) direction and in its non-LOS direction (see Figure 4.8). Hence the error has also been studied in this frame by rotating the aforementioned error in the LOS frame (which is different for every observation). This is achieved using the ground truth estimate of the vehicle yaw $\check{\theta}_k$ and the line-of-sight orientation in the vehicle frame θ_{LOS}, as

$$\begin{bmatrix} {}^{LOS}e_k^j \\ {}^{LOS}e_k^j \end{bmatrix} = \begin{bmatrix} \cos\left(\check{\theta}_k + \theta_{LOS}\right) & \sin\left(\check{\theta}_k + \theta_{LOS}\right) \\ -\sin\left(\check{\theta}_k + \theta_{LOS}\right) & \cos\left(\check{\theta}_k + \theta_{LOS}\right) \end{bmatrix} \begin{bmatrix} {}^0e_k^j \\ {}^0e_k^j \end{bmatrix} \tag{4.29}$$

The error distribution is the Line-of-Sight frame is shown in Figure 4.10. Unlike in the RS frame, there seem to be some differences between the errors in the LOS direction and those in the non-LOS direction. This is confirmed by the estimated covariance matrix,

$${}^{LOS}R = \begin{bmatrix} (0.0916)^2 & -0.00018 \\ -0.00018 & (0.1125)^2 \end{bmatrix} \tag{4.30}$$

There seem to be a difference, albeit small, between the uncertainties along the LOS direction and across the LOS direction. This difference can be explained by the different sources causing the measurement error. The error along the LOS direction is due to errors in the measurement of distances by the sensor. These errors are due to errors in the measurement of travel time of the signal (timing errors). Errors in the non-LOS direction are due to errors in the measurement of the sensor angle. The two dimensions are affected by different measurement errors which can lead to different uncertainties.

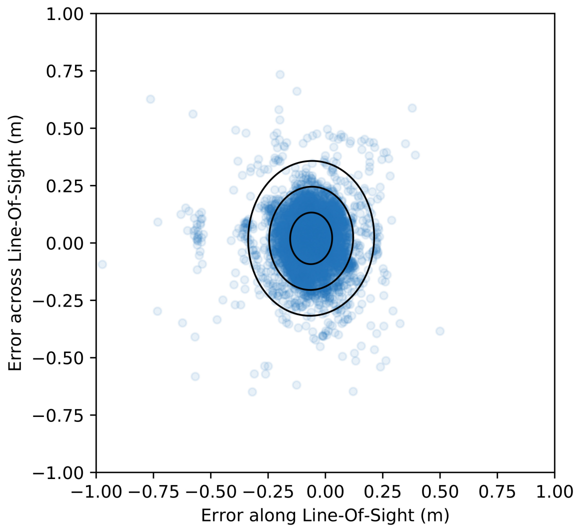


Figure 4.10.: Distributions of road sign errors in the Line-of-Sight (LOS) frame. In black are represented the 1σ, 2σ and 3σ ellipses.

The dependency of the errors to factors such as the distance to the detection has been studied. As can be seen on Figure 4.11, the distance to detection does not seem to affect the error. This can be expected in the LOS direction as the lidar used measure distances based on time of flight whose accuracy is only dependent on the clock accuracy. However, a dependency could be expected in the non-LOS direction as the lidar does not produce measurements using pure lasers but rather spots caused by the divergence of the beams. The horizontal beam divergence for the VLP-32C lidar is 3.0 mrad which results in a 15 cm wide spot at 50 meters. Despite this effect, no noticeable dependency has been observed. This is probably due to the observation not being produced using a single point but rather by averaging several ones.

The final noise model used in the estimation is chosen to follow the Line-Of-Sight model. This model is chosen as it is the only one of the three studied ones that exhibits a difference between the two dimensions. In other frame the dependency is hidden as each road signs is oriented differently and they are observed with different line-of-sights. Only accounting for the direction of the line-of-sight, can the dependency be observed. Therefore, the noise model used in the state estimation is,

$$R =^0 R_{LOS} \cdot {}^{LOS}R \cdot {}^0R_{LOS}^\top \tag{4.31}$$

where ${}^0R_{LOS}$ is the rotation matrix from the Line-Of-Sight frame to the global frame.

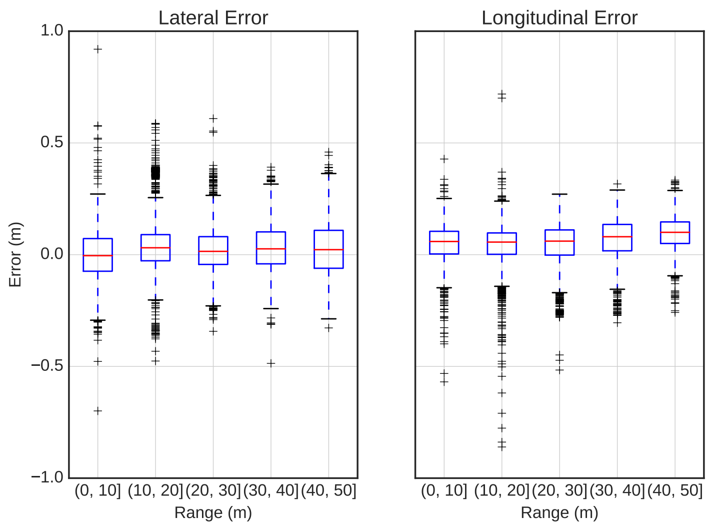


Figure 4.11.: Road sign error distributions in the LOS frame depending on the distance to the detection. The left boxplot shows the effect of the distance on the error along the LOS direction, and the right boxplot shows the error in the non-LOS direction. There is no noticeable influence of the distance on the observation error.

4.3.3.4. Extrinsinc Calibration of the Lidar

Proper calibration of the sensor relative to the rest of the vehicle is essential for localization. Bad calibration will result in systematic errors which introduces biases in the estimation process. Calibration of the sensor position on the vehicle can be done using high accuracy calibration equipment. This enables to localize the sensor with millimeter accuracy in the mobile frame. Calibrating the sensor orientation is more challenging. The Velodyne VLP-32C lidar is a cylinder offering no easy point of reference of the sensor orientation from the outside. Thankfully the road sign detector combined with the map and a ground truth positioning provide a simple way to calibrate the sensor. Indeed, an incorrect calibration of the sensor orientation will result in an observation error in the non-LOS direction as shown on Figure 4.12. The best sensor orientation is found by finding the orientation that minimizes this error. Although this has been done manually as it only needs to be done once per dataset series and vehicle, it can be easily automated to be applied at larger scales.

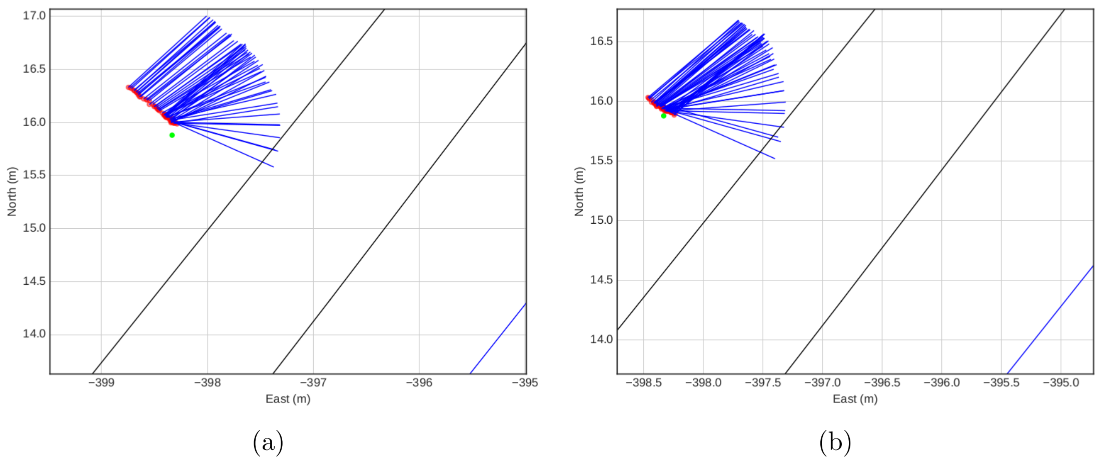


Figure 4.12.: Effect of a bad calibration on observations of a single road sign. In green is the geo-referenced road sign, in red are the measurements and the blue lines represent the direction of line-of-sight of these measurements (the length of the line is proportional to the distance to the vehicle). (a) shows the result of a badly calibrated lidar, the error is high when the vehicle is far and decreases as the vehicle gets closer to the sign. (b) shows a properly calibrated lidar, the measurements are spread around the sign with little effect of the distance to the vehicle.

4.4. Improving the Matching using an Adjustment Step over the Parallel Window of Data

As introduced in the previous sections, we aim at taking benefit of detected road features to improve localization accuracy and integrity. Observations are of two kinds, namely one-dimensional (i.e. lane markings that provide no information about the longitudinal position of the vehicle along the line) and multi-dimensional (i.e. traffic signs and pedestrian crossings for which a position and orientation can be observed). This section presents the matching processes developed for lane markings only (one-dimensional) and its extension to any measurements (multi-dimensional).

4.4.1. One-dimensional Case

Associating an observed lane marking with a map marking is often done using the smallest distance between observation and map [Schreiber et al., 2013, Tao et al., 2017]. This method highly relies on an accurate state estimates to prevent miss matches between observations and map features. When the map contains a large number of features and when the camera detects several lanes, matching ambiguities

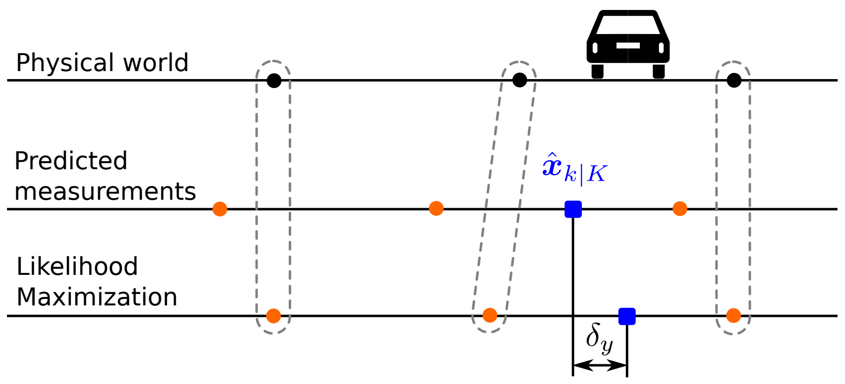


Figure 4.13.: Cross-section representing the steps used to associate observations with markings. The top row illustrates the map and the true vehicle position. The middle row shows the measurement position relative to the state. The last row shows the result after likelihood maximization where the associations (dashed lines) are selected.

may become an issue for localization purposes. For that reason, the association is performed in two steps. The first step is to find the transformation to apply to the observations so that they best overlap the map. Then, the observations can be associated with a marking by finding the closest marking after the transformation. This process is illustrated in Figure 4.13.

4.4.1.1. Lane Markings Observations

As mentioned in Chapter 2, the marking measurements are not matched in real time but rather are saved in a buffer to be matched in batches. Therefore, at an epoch K, when the measurements will be matched to map features, a buffer from epoch $K-S$ to epoch K is available (where S is the size of the buffer). The set of lane marking measurements is:

$$\{C_k^j\}_{j\in\Lambda_k, k\in[\![K-S,K]\!]} \tag{4.32}$$

where Λ_k is the set of lines observed at time k, K is the end of the buffer and S is the size of the buffer.

We aim at finding the best overlap between the observations and the map. To do so, we first convert the measurements to points using the estimated vehicle state. For each measurement, the two-dimensional points they correspond to are computed in the global frame $\mathcal{R}_0$. These points are computed using $\hat{\boldsymbol{X}}_K = \{\hat{\boldsymbol{x}}_{k|K}\}_{k\in[\![K-S,K]\!]}$ the state estimates smoothed over the buffer as so,

$$\begin{cases} {}^0X_k^j = & \hat{x}_{k|K} + l_c \cdot \cos\left(\hat{\theta}_{k|K}\right) - C_k^j \cdot \sin\left(\hat{\theta}_{k|K}\right) \\ {}^0Y_k^j = & \hat{y}_{k|K} + l_c \cdot \sin\left(\hat{\theta}_{k|K}\right) + C_k^j \cdot \cos\left(\hat{\theta}_{k|K}\right) \end{cases} \tag{4.33}$$

where l_c is the distance between the vehicle frame and the camera frame, and $\hat{x}_k$, $\hat{y}_k$, $\hat{\theta}_k$ are respectively the estimated east and north coordinates and heading at time k of the observations.

The list of points is therefore written as:

$$\left\{ \begin{bmatrix} {}^0X_k^j & {}^0Y_k^j \end{bmatrix}^T \right\}_{j \in \Lambda_k, k \in [\![K-S,K]\!]} \tag{4.34}$$

It corresponds to multiple lane markings measurements. The coordinates of these points are in the global reference frame. To simplify the rest of the computation, both the map and the observed points are expressed into the vehicle frame $\mathcal{R}_B$ (a buffer frame) which is positioned and oriented based on the most recent state $\hat{\boldsymbol{x}}_{K|K}$ of the buffer. This is done by applying the transformation:

$$\begin{bmatrix} {}^BX_k^j \\ {}^BY_k^j \end{bmatrix} = \begin{bmatrix} \cos\theta_{K|K} & \sin\theta_{K|K} \\ -\sin\theta_{K|K} & \cos\theta_{K|K} \end{bmatrix} \begin{bmatrix} {}^0X_k^j - \hat{x}_{K|K} \\ {}^0Y_k^j - \hat{y}_{K|K} \end{bmatrix} \tag{4.35}$$

In the rest of this section, the observations in this local frame are referenced as:

$$\left\{ \begin{bmatrix} {}^BX_k^j & {}^BY_k^j \end{bmatrix}^\top \right\}_{j \in \Lambda_k, k \in [\![K-S,K]\!]} \tag{4.36}$$

4.4.1.2. Transformation Definition

To correctly associate the observations to the map markings, an adjustment step is first performed to best overlap the measurements and the map. Indeed, there is always a localization error due mainly to dead-reckoning drift.

The aforementioned points are two-dimensional and would therefore require estimating a two-dimensional translation and a rotation. However, since lane markings can only be properly detected in relatively straight lines, the longitudinal translation cannot be estimated. Hence, only the lateral translation will be estimated. Also, in this work, the estimation of the vehicle heading is assumed to be fairly good since the evolution model used is well calibrated [Welte et al., 2019a]. Therefore, the rotational component will not be estimated. Finally, only the lateral shift to apply has to be found. Since the local frame of the vehicle is used, it is the shift along the y-axis that needs to be estimated. We assume that the lateral correction to be applied is the same for all points in the buffer. This hypothesis is very much in line with observations we made experimentally during our first tests. It is valid only for a buffer of rather short length. Therefore, the measurements will be shifted by δ_y resulting in the following measurements at the end of the adjustment stage:

$$\left\{ \begin{bmatrix} {}^BX_k^j & {}^BY_k^j + \delta_y \end{bmatrix}^T \right\}_{j \in \Lambda_k, k \in [\![K-S,K]\!]} \tag{4.37}$$

4.4.1.3. Map Modeling

To find the transformation δ_y that makes the observed points and the map best overlap, the value δ_y that maximizes the likelihood of obtaining the set of points $\left\{\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top\right\}_{j\in\Lambda_k, k\in[\![K-S,K]\!]}$ with the set of lane markings $\{\boldsymbol{m}_i\}_{i\in[\![1,M]\!]}$ is searched for.

From the law of total probabilities, the likelihood of measuring a point at the coordinates $\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top$ given a map composed of M markings is

$$\begin{aligned}
L_{\hat{\boldsymbol{x}}_{k|K}}\left(\delta_y;\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j\end{bmatrix}\right) =& p\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j\end{bmatrix};\delta_y\right) = p\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\right)\\
=& p\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix} \mid \begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\leftrightarrow \boldsymbol{m}_1\right)\mathbb{P}\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\leftrightarrow \boldsymbol{m}_1\right)\\
&+\ldots+\\
&p\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix} \mid \begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\leftrightarrow \boldsymbol{m}_M\right)\mathbb{P}\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\leftrightarrow \boldsymbol{m}_M\right)\\
&+C,
\end{aligned} \tag{4.38}$$

where $\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top \leftrightarrow \boldsymbol{m}_i$ means that the observation $\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top$ corresponds to the marking $\boldsymbol{m}_i$, and C is a constant used to account for the case where the observation does not correspond to any known marking.

Since no prior information is available as to which marking a particular observation is associated with, $\mathbb{P}\left(\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top \leftrightarrow \boldsymbol{m}_i\right)$ is considered identical for each observation and marking and equal to $\frac{1}{M+1}$ (C is also taken equal to $\frac{1}{M+1}$).

To simplify notations, we define

$$L_{k,j}\left(\delta_y\right) = L_{\hat{\boldsymbol{x}}_{k|K}}\left(\delta_y;\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j\end{bmatrix}\right) \tag{4.39}$$

and

$$L_{k,j,i}\left(\delta_y\right) = p\left(\begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix} \mid \begin{bmatrix}{}^BX_k^j\\{}^BY_k^j+\delta_y\end{bmatrix}\leftrightarrow \boldsymbol{m}_i\right) \tag{4.40}$$

$L_{k,j,i}$ is modeled as a Gaussian distribution centered on the marking and spreading along the y-axis. Hence, for a marking $\boldsymbol{m}_i$ the likelihood of measuring the point $\begin{bmatrix}{}^BX_k^j & {}^BY_k^j + \delta_y\end{bmatrix}^\top$ associated with the marking $\boldsymbol{m}_i$ is

$$L_{k,j,i}\left(\delta_y\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(Y_k^j+\delta_y-y_i({}^BX_k^j)\right)^2}{2\sigma^2}\right), \tag{4.41}$$

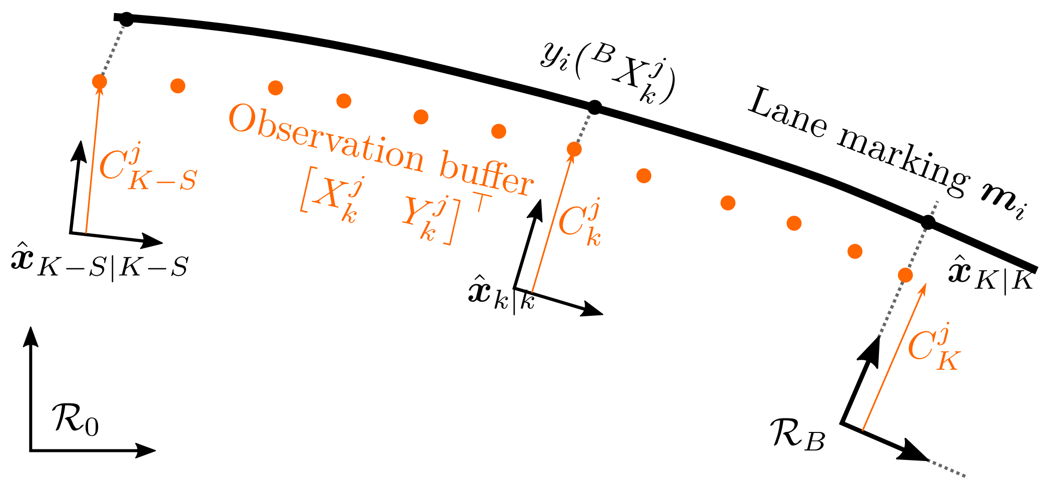

$y_i(^B X_k^j)$
Lane marking $\boldsymbol{m}_i$
Observation buffer
$[X_k^j \quad Y_k^j]^\top$
C_{K-S}^j
$\hat{\boldsymbol{x}}_{K-S|K-S}$
C_k^j
$\hat{\boldsymbol{x}}_{k|k}$
$\hat{\boldsymbol{x}}_{K|K}$
C_K^j
$\mathcal{R}_0$
$\mathcal{R}_B$

Figure 4.14.: Representation of the observation buffer and its projection on the lane marking $\boldsymbol{m}_i$. In orange is displayed the measurements j, in black the map marking and projections of the observations.

where $y_i(^B X_k^j)$ is the projection along the y-axis of the observation $\left[{}^B X_k^j \quad {}^B Y_k^j + \delta_y\right]^\top$ on the marking $\boldsymbol{m}_i$ (this is illustrated in Figure 4.14), and σ^2 is the variance representing the combined uncertainty of the marking (σ_m^2), the measurement ($\sigma_{k,j}^2$), and estimated state (σ_{lat}^2),

$$\sigma^2 = \sigma_m^2 + \sigma_{k,j}^2 + \sigma_{lat}^2 \tag{4.42}$$

The map variance σ_m^2 corresponds to the uncertainty of the lane marking reference in the map. It is chosen equal to 0 since the map is supposed to be highly accurate, $\sigma_{k,j}^2$ stands for the uncertainty of the measurement and σ_{lat}^2 is the lateral uncertainty of the state estimate.

The likelihood of having a lane marking measurement at the point $\left[{}^B X_k^j \quad {}^B Y_k^j + \delta_y\right]^\top$ with a map having multiple lane markings is

$$L_{k,j}\left(\delta_y\right) = \frac{1}{M+1}\left(\sum_{i\in[\![1,M]\!]} L_{k,j,i}\left(\delta_y\right) + 1\right) \tag{4.43}$$

The likelihood of a single observation is therefore modeled as a multi-modal Gaussian distribution with M modes (plus a constant).

Figure 4.15 shows an example for three referenced markings with the same σ^2 (in practice σ^2 varies for each observation because $\sigma_{k,j}^2$ depends on the observation).

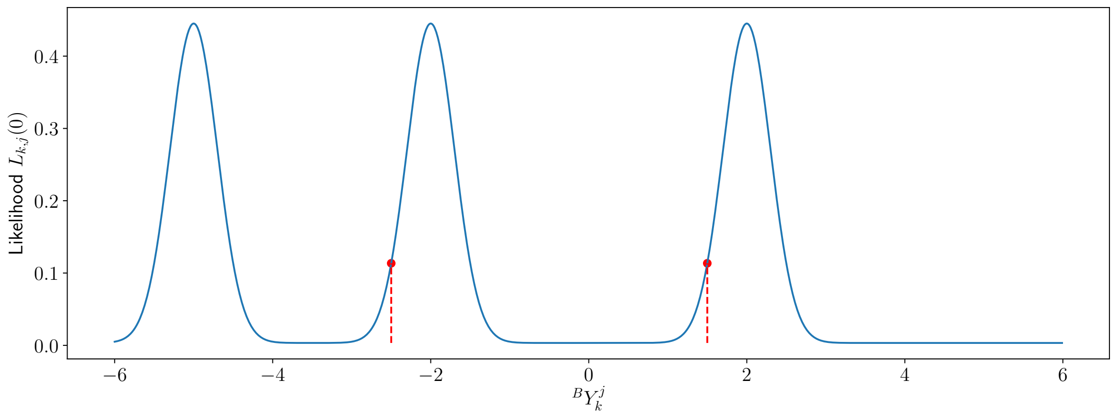


Figure 4.15.: The likelihood of an observation with a map referencing three lane markings. The map contains lane markings at coordinates −5, −2, 2. The likelihood of an observation knowing this map is in blue as computed by Equation 4.43. The two observations $Y_0^0 = -2.5$ and $Y_0^1 = 1.5$ used in Figure 4.16 are shown in red.

Assuming the measurements are independent of each other, the likelihood of detecting the list of points $\left\{\left[{}^{B}X_k^j \quad {}^{B}Y_k^j + \delta_y\right]^\top\right\}_{j\in\Lambda_k, k\in[\![K-S,K]\!]}$ in the buffer knowing M mapped lane markings is

$$L(\delta_y) = \prod_{\substack{j\in\Lambda_k,\\ i\in[\![k-S,k]\!]}} L_{k,j}(\delta_y) \tag{4.44}$$

$$= \prod_{\substack{j\in\Lambda_k,\\ i\in[\![K-S,K]\!]}} \left(\frac{1}{M+1}\left(\sum_{i\in[\![1,M]\!]} L_{k,j,i}(\delta_y) + 1\right)\right) \tag{4.45}$$

The transformation δ_y that maximizes this likelihood is the transformation that enable the measurements to best overlap the map. As shown in Figure 4.16, in the example of two observations knowing a three lane markings map, $L(\delta_y)$ is maximized around $\delta_y = 0.5$ m which corresponds to the gap between the observation positions and the feature positions (see Figure 4.15). In this work, as in most others where such a maximization is needed, the log-likelihood is maximized. This is done to improve the numerical stability of the problem. Products of small values will lead to even smaller values which are hard to manage by the processor while sum of large values are still manageable. The best transformation corresponds to the value δ_y

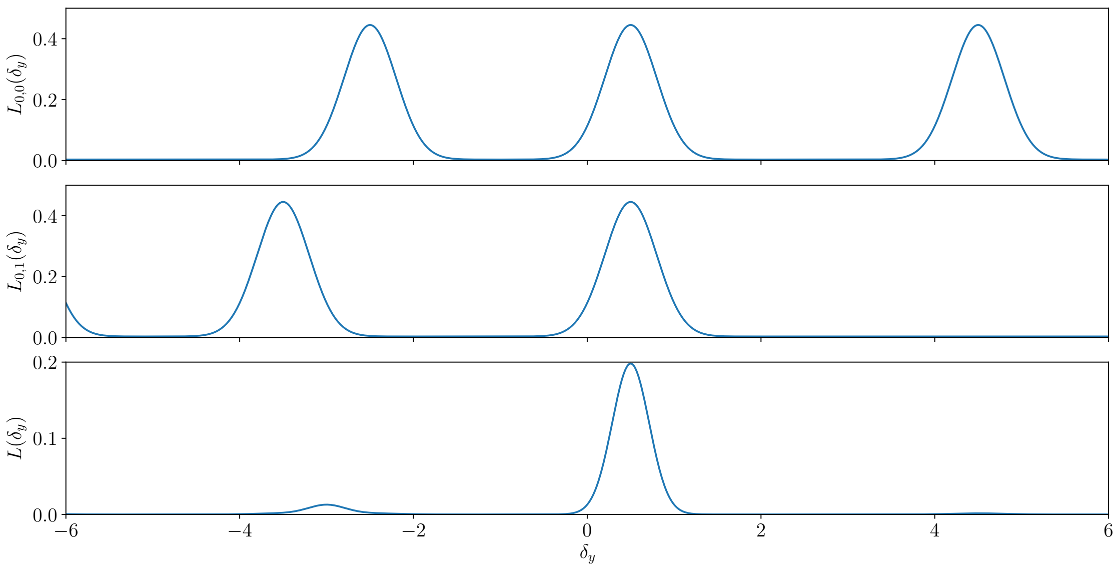


Figure 4.16.: Combination of the likelihoods of δ_y of multiple observations. The likelihood $L_{0,0}$ from the first observation (top) and the one ($L_{0,1}$) from the second observation (middle) are combined to give the curve of the lower figure as described by Equation 4.44

that maximizes the log-likelihood, thus

$$\hat{\delta_y} = \arg\max_{\delta_y} \left(\log\left(L\left(\delta_y\right)\right)\right) \tag{4.46}$$

$$= \arg\max_{\delta_y} \left(\sum_{\substack{j\in\Lambda_k,\\ k\in[\![K-S,K]\!]}} \log\left(L_{k,j}\left(\delta_y\right)\right) \right) \tag{4.47}$$

4.4.1.4. Likelihood Maximization

To solve this problem a gradient descent technique is used. The maximum of the following function is found by successive iterations along the direction of its gradient.

$$\frac{d(\log L)}{d\delta_y}\left(\delta_y\right) = \sum_{\substack{j\in\Lambda_k,\\ k\in[\![K-S,K]\!]}} \frac{d\ \log\left(L_{k,j}\left(\delta_y\right)\right)}{d\delta_y}\left(\delta_y\right) \tag{4.48}$$

This method converges after a few iterations, and so more advanced resolution techniques are unnecessary. The transformation δ_y is initialized at 0. At each iteration, δ_y is increased by

$$\gamma \cdot \frac{d\left(\log L\right)}{d\delta_y}\left(\delta_y\right) = \gamma \times \sum_{\substack{j\in\Lambda_k,\\ k\in[\![K-S,K]\!]}} \frac{\sum_{i\in[\![1,M]\!]} \frac{dL_{k,j,i}}{d\delta_y}\left(\delta_y\right)}{\sum_{i\in[\![1,M]\!]} L_{k,j,i}\left(\delta_y\right)+1} \tag{4.49}$$

where γ is chosen empirically. This process stops when the update from an iteration to the next is smaller than a given threshold η or when the process reaches a given maximum number of iterations.

4.4.1.5. Association

Once $\hat{\delta}_y$ has been found, each measurement can be associated with a marking. To do so, for each measurement, the likelihood that it matches a marking $\boldsymbol{m}_i$ is computed as

$$L_{k,j,i}\left(\hat{\delta}_y\right) = \frac{1}{\sqrt{2\pi\sigma_{LM}^2}} \exp\left(-\frac{\left(Y_k^j + \hat{\delta}_y - y_i({}^BX_k^j)\right)^2}{2\sigma_{LM}^2}\right), \tag{4.50}$$

The measurement j is associated to the marking for which this likelihood is maximum. Therefore, the measurement j is associated to the marking

$$\hat{\boldsymbol{m}}_i = \arg\max_{\boldsymbol{m}_i} L_{k,j,i}\left(\hat{\delta}_y\right) \tag{4.51}$$

4.4.1.6. Outliers Rejection

The associations found previously are not directly used to update the state. Several outlier rejection steps are performed. Indeed, missing markings in the map, missing observations or wrong observations can create scenarios where the associations found previously are not accurate. In particular, in situations where the road only has a central marking but the map has only the road edges referenced, the camera will be able to detect the unreferenced marking. Because the detected marking type is not reliable enough, the referenced road edges have to be considered as potential association candidates. The likelihood maximization will converge to overlap the detected central marking with the mapped road edges (see Figure 4.17a). This can be easily solved by discarding every association where δ_y is too high. In the current implementation of the method, the threshold to discard the markings is taken as a constant:

$$|\delta_y| > \Delta_{max} \tag{4.52}$$

Also, the previous method will result in every measurement being associated with a marking regardless of their distance to one another after the gradient descent. Since multiple markings can be observed at once, the likelihood maximization can

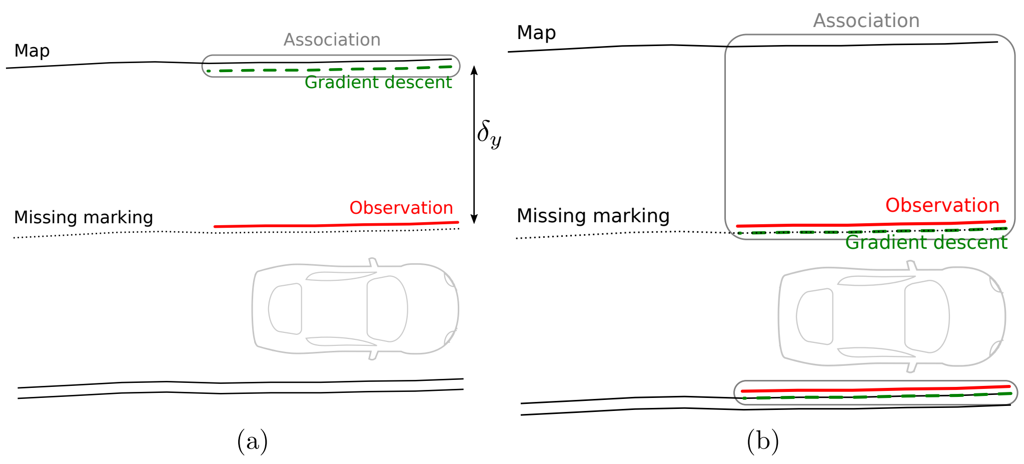


Figure 4.17.: Graphical representation of cases requiring an outlier rejection step. Left: The maximization converges on the wrong association because of a missing marking in the map. Right: The maximization converges properly thanks to the right observation but the left observation does not match to any marking.

accurately converge even if one marking has no correct association. The method will converge so that most measurements overlap a marking but it can happen that some observations do not overlap any map markings either because the marking is missing in the map or the camera measurements are erroneous (see Figure 4.17b). The ability to manage these problematic situations for matching methods is a big strength of our method which is able to solve them thanks to the global association it achieves. To account for this, a second threshold is used to remove individual measurements if their residual $y_k^j = Y_k^j + \delta_y - y_i\left(X_k^j\right)$ (computed using the measurements compensated by δ_y) is too high:

$$\left|y_k^j\left(\delta_y\right)\right| > \epsilon \tag{4.53}$$

The two thresholds have been chosen empirically in order to correctly reject erroneous matches in typical driving situations. The numerical values are provided in Subsection 4.5.1.

4.4.2. Multi-dimensional Case

Unlike lane markings which offer no information on the vehicle longitudinal position, road signs offer a two-dimensional information. The previous approach is therefore extended here to two-dimensional features. When using in real-time high level features, such as road markings or signs, the observation set $\boldsymbol{Z}^{(k)}$ at time k is of limited size (see Figure 4.18 (a)). Having few observations can lead to wrong data association as many ambiguities can arise due to a large pose uncertainty. Moreover, if the

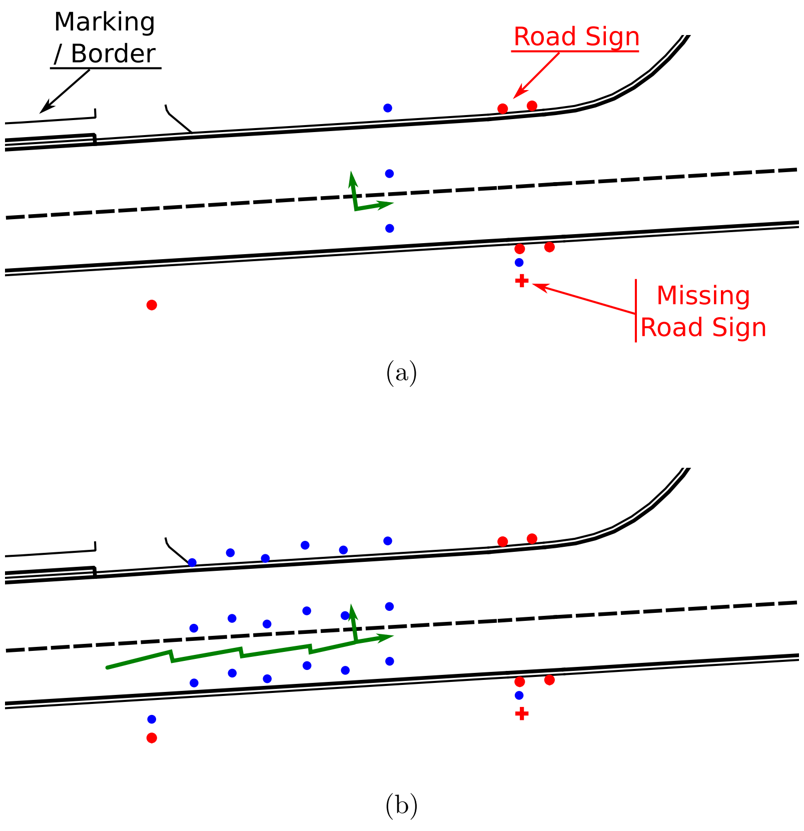
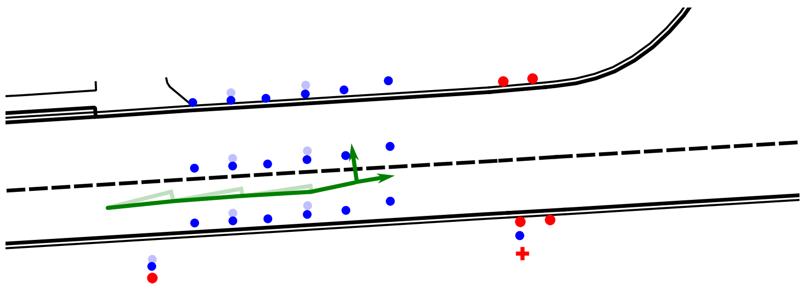


Figure 4.18.: Available observation set using snapshot matching (a) and by using observation and state estimate buffers (b). The scene presented here is a simplification of a real scene. Results for this scene are presented in Subsection 4.5.3. Blue: measurements, Green: pose estimates, Red: road signs (dots are mapped, the cross is not)

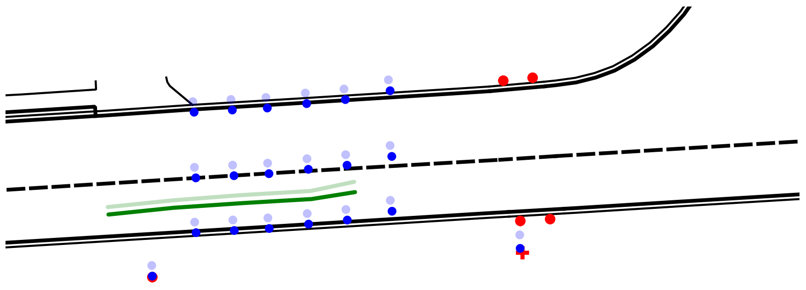
perception modules are asynchronous, the markings from the camera and the signs from the lidar are likely to arrive at different times.

The final objective is to use the exteroceptive measures with the best association in the update of the filter. Consider for instance the real situation of Figure 4.18 (a) where a missing road sign is detected but unfortunately there are other signs nearby. In order to limit association ambiguities, the observations are not matched in a snapshot manner. Buffers containing the observations are used with filtered state estimates (see Figure 4.18 (b)). Hence, the matching is performed using more observations thus limiting the risk of ambiguous matching. The observation buffer provides a larger number of measurements and allows a globally consistent matching, which is known to be more robust.

Thanks to this approach another problem can be addressed. Estimation errors which are time-correlated can cause incorrect matchings. A Kalman smoothing is first applied on the buffer to correct the jaggedness of the filtered estimates (see



(a)



(b)

Figure 4.19.: Effect of the smoothing (a) and trajectory adjustment steps (b) in the state estimate and observation buffers. Blue: measurements, Green: pose estimates, Red: road signs (dots are mapped, the cross is not), Black: markings. Extracted from a real situation encountered in Rambouillet.

Figure 4.19 (a)). Then, an adjustment of the whole trajectory over the buffer is made to compensate as much as possible the filter errors to find the best data association. Figure 4.19 (b) shows the final result. It can be noticed that the faulty road sign measurement has been sufficiently shifted away such that a wrong matching will be rejected in the last filter update. The observation buffer is processed in parallel of the real-time localization at a low frequency (e.g. every 250 ms) since it cannot always be treated within the filter update period. Measurements for which matches have been found will become usable as observations by the real-time estimation layer. At the next estimation time, the filter will re-estimate the states to account for the new observations. This is explained in more details in Subsection 2.3.3.

Let K be the most recent time of the observation buffer, with which the matching is solved. Let $\boldsymbol{Z}_K = \left\{ \boldsymbol{Z}^{(k)} \,| k \in [\![K-S, K]\!] \right\}$ be the set of all the observations over a buffer of size S. The goal is to associate all the elements of $\boldsymbol{Z}_K$ simultaneously. The following sections detail the steps used to achieve this. The vehicle trajectory is first smoothed. It is then adjusted using the observations. The adjusted trajectory is finally used to match observations to map features.

4.4.2.1. Pose Smoothing

As for the one-dimensional case, the filtered state estimates are not directly used for matching. Instead, the smoothed trajectory $\hat{\boldsymbol{X}}_K = \left\{ \hat{\boldsymbol{x}}_{k|K} \mid k \in [\![K-S, K]\!] \right\}$ is used. Lower uncertainty will be very useful for the data association process.

4.4.2.2. Pose Adjustment

An additional way to improve the state estimation is to maximize the likelihood of all the states in the buffer given all the observations. Let us first consider this problem in a deterministic formulation in which the states are unknown but not random variables:

$$L\left(\boldsymbol{x}_{K-S}, \ldots, \boldsymbol{x}_K; \boldsymbol{Z}^{(K-S)}, \ldots, \boldsymbol{Z}^{(K)}\right) = p\left(\boldsymbol{Z}^{(K-S)}, \ldots, \boldsymbol{Z}^{(K)}; \boldsymbol{x}_{K-S}, \ldots, \boldsymbol{x}_K\right) \quad (4.54)$$

where $p\left(\boldsymbol{Z}^{(K-S)}, \ldots, \boldsymbol{Z}^{(K)}; \boldsymbol{x}_{K-S}, \ldots, \boldsymbol{x}_K\right)$ is the evaluation of the joint probability density function of the observations $\boldsymbol{Z}^{(K-S)}, \ldots, \boldsymbol{Z}^{(K)}$ parameterized by a given buffer of states $\boldsymbol{x}_{K-S}, \ldots, \boldsymbol{x}_K$. This is indicated by the ";" in the equation.

Maximizing this likelihood function in Equation 4.54 for all the poses in the buffer is computationally expensive. Moreover, the temporal coherence between the successive state estimates coming from the evolution model could be degraded. Instead, we propose to estimate a unique 2D local rigid transformation, *i.e.*, a translation and a rotation, $\boldsymbol{\delta} = \left[\delta_x, \delta_y, \delta_\theta\right]$ to all the state estimates $\hat{\boldsymbol{x}}_{k|K}$ so that the likelihood of the resulting states is maximized.

Let $\widetilde{\boldsymbol{X}}_K(\boldsymbol{\delta}) = \left\{\widetilde{\boldsymbol{x}}_{k|K}\left(\boldsymbol{\delta}\right) | k \in [\![K-S, K]\!]\right\}$ be the $\boldsymbol{\delta}$-adjusted state estimates. These states are defined by Equation 4.55 from the smoothed state using the global translation from $\left[\delta_x, \delta_y\right]$ and the rotation δ_θ around a point chosen arbitrarily as the most recent state.

$$\widetilde{\boldsymbol{x}}_{k|K}\left(\boldsymbol{\delta}\right) = \begin{bmatrix}\widetilde{x}_{k|K}\\ \widetilde{y}_{k|K}\\ \widetilde{\theta}_{k|K}\end{bmatrix} = \begin{bmatrix}\cos\delta_\theta & -\sin\delta_\theta & 0\\ \sin\delta_\theta & \cos\delta_\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{x}_{k|K}\\ \hat{y}_{k|K}\\ \hat{\theta}_{k|K}\end{bmatrix} + \begin{bmatrix}\delta_x\\ \delta_y\\ \delta_\theta\end{bmatrix} \tag{4.55}$$

A new likelihood function only on $\boldsymbol{\delta}$ is defined as

$$L_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{\delta};\boldsymbol{Z}_K\right) = p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K;\boldsymbol{\delta}\right) \tag{4.56}$$

where

$$p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K;\boldsymbol{\delta}\right) = p\left(\boldsymbol{Z}_K;\widetilde{\boldsymbol{X}}_K\left(\boldsymbol{\delta}\right)\right) \tag{4.57}$$

which leads to

$$\begin{aligned}
L_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{\delta};\boldsymbol{Z}_K\right) &= p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}^{(K-S)},\ldots,\boldsymbol{Z}^{(K)};\boldsymbol{\delta}\right) && \text{For all sets of obs in the buffer}\\
&= \prod_{k\in[\![K-S,K]\!]} p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}^{(k)};\boldsymbol{\delta}\right) && \text{Conditional independence}\\
&= \prod_{k\in[\![K-S,K]\!]} p\left(\boldsymbol{Z}^{(k)};\widetilde{\boldsymbol{x}}_{K-S|K}\left(\boldsymbol{\delta}\right),\ldots,\widetilde{\boldsymbol{x}}_{K|K}\left(\boldsymbol{\delta}\right)\right) && \text{Rewriting}\\
&= \prod_{k\in[\![K-S,K]\!]} p\left(\boldsymbol{Z}^{(k)};\widetilde{\boldsymbol{x}}_{k|K}\left(\boldsymbol{\delta}\right)\right) && \text{Markov hypothesis}\\
&= \prod_{k\in[\![K-S,K]\!]} p_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{Z}^{(k)};\boldsymbol{\delta}\right) && \text{Rewriting}\\
&= \prod_{k\in[\![K-S,K]\!]} p_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{z}_1^{(k)},\boldsymbol{z}_2^{(k)},\ldots;\boldsymbol{\delta}\right) && \text{For each elementary obs}\\
&= \prod_{k\in[\![K-S,K]\!]}\prod_{\boldsymbol{z}_j^{(k)}\in\boldsymbol{Z}^{(k)}} p_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{z}_j^{(k)};\boldsymbol{\delta}\right) && \text{Conditional independence}\\
&= \prod_{k\in[\![K-S,K]\!]}\prod_{\boldsymbol{z}_j^{(k)}\in\boldsymbol{Z}^{(k)}} L_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{\delta};\boldsymbol{z}_j^{(k)}\right) && (4.58)
\end{aligned}$$

Let us now consider the data association issues.

The observation $\boldsymbol{z}_j^{(k)}$ is related to the state estimate $\widetilde{\boldsymbol{x}}_{k|K}$ via the map features $\boldsymbol{M} = \{\boldsymbol{m}_1,\boldsymbol{m}_2,\ldots\}$. Let us consider the case when $\boldsymbol{z}_j^{(k)}$ (one of the observations at time k) is associated to the map feature $\boldsymbol{m}_i$. Then, the corresponding likelihood is expressed as follows with a Gaussian assumption:

$$L_{i,j}\left(\boldsymbol{\delta};\boldsymbol{z}_j^{(k)}\right) = \frac{1}{\sqrt{(2\pi)^d|\widetilde{\boldsymbol{S}}_{i,j}|}}\exp\left(-\frac{1}{2}\widetilde{\boldsymbol{y}}_{i,j}^\top\widetilde{\boldsymbol{S}}_{i,j}^{-1}\widetilde{\boldsymbol{y}}_{i,j}\right) \tag{4.59}$$

with

$$\widetilde{\boldsymbol{y}}_{i,j} = \boldsymbol{z}_j^{(k)} - h_i\left(\widetilde{\boldsymbol{x}}_{k|K}(\boldsymbol{\delta})\right) \ , \quad \widetilde{\boldsymbol{S}}_{i,j} = \boldsymbol{H}_i \boldsymbol{P}_{k|K} \boldsymbol{H}_i^\top + \boldsymbol{R}_j \tag{4.60}$$

and $d = 1$ for road markings and $d = 2$ for road signs. Because the association between $\boldsymbol{z}_j^{(k)}$ and $\boldsymbol{m}_i$ is not known at this stage, all the map features $\boldsymbol{M}$ in the vicinity have to be considered.

From the law of total probability, we have

$$L_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{\delta}; \boldsymbol{z}_j^{(k)}\right) = \sum_{\boldsymbol{m}_i \in \boldsymbol{M}} L_{i,j}\left(\boldsymbol{\delta}; \boldsymbol{z}_j^{(k)}\right) \mathbb{P}_{i,j} \tag{4.61}$$

where $\mathbb{P}_{i,j}$ is the probability that $\boldsymbol{z}_j^{(k)}$ corresponds to $\boldsymbol{m}_i$.

A discrete uniform distribution over $\mathbb{P}_{i,j}$ is chosen, as no additional information such as appearance or shape cues on the detections or on the features is available. So, there is no reason to favor one association over another. The likelihood function in Equation 4.61 does not take into consideration the fact that the sensors may detect features that either do not exist or have not been mapped. To account for this, we propose to add a non-association probability $\mathbb{P}_{\emptyset,j}$ and a likelihood $L_{\emptyset,j}$ which leads to

$$L_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{\delta}; \boldsymbol{z}_j^{(k)}\right) = \sum_{\boldsymbol{m}_i \in \boldsymbol{M}} L_{i,j}\left(\boldsymbol{\delta}; \boldsymbol{z}_j^{(k)}\right) \mathbb{P}_{i,j} + L_{\emptyset,j}\mathbb{P}_{\emptyset,j} \tag{4.62}$$

In practice, maximizing the likelihood in Equation 4.61 may lead to a large adjustment $\boldsymbol{\delta}$, especially when there are some spatial invariance, *i.e.*, along a straight lane.

In order to constrain the spacial adjustment, we propose to reformulate the maximum likelihood estimation into a maximum *a posteriori* (MAP) one in a Bayesian context. For that purpose, we set the *a priori* distribution over $\boldsymbol{\delta}$ as $\boldsymbol{\delta} \sim \mathcal{N}\left(0; \boldsymbol{P}_{K|K}\right)$. We chose arbitrarily the most recent covariance matrix estimate $\boldsymbol{P}_{K|K}$ from the buffer as a representative measure of the pose uncertainty.

The resulting MAP estimation on the whole buffer is then defined as

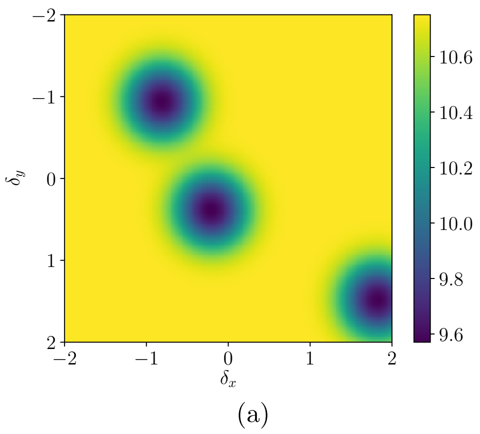
$$\begin{aligned}\hat{\boldsymbol{\delta}} &= \arg\max_{\boldsymbol{\delta}} \left(p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K|\boldsymbol{\delta}\right) \cdot p\left(\boldsymbol{\delta}\right)\right) \\ &= \arg\max_{\boldsymbol{\delta}} \left(p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K|\boldsymbol{\delta}\right) \cdot \exp\left(-\frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{P}_{K|K}^{-1}\boldsymbol{\delta}\right)\right)\end{aligned} \tag{4.63}$$

Please note that $p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K|\boldsymbol{\delta}\right)$ is now a conditional density function.

It is common to write it equivalently as minimizing the negative log-likelihood

$$\hat{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\delta}} \left(-\log p_{\hat{\boldsymbol{X}}_K}\left(\boldsymbol{Z}_K|\boldsymbol{\delta}\right) - \log p\left(\boldsymbol{\delta}\right)\right) \tag{4.64}$$

$$= \arg\min_{\boldsymbol{\delta}} \left(\sum_{k \in [\![K-S, K]\!]} \sum_{\boldsymbol{z}_j^{(k)} \in \boldsymbol{Z}^{(k)}} -\log p_{\hat{\boldsymbol{x}}_{k|K}}\left(\boldsymbol{z}_j^{(k)}|\boldsymbol{\delta}\right) + \frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{P}_{K|K}^{-1}\boldsymbol{\delta}\right) \tag{4.65}$$



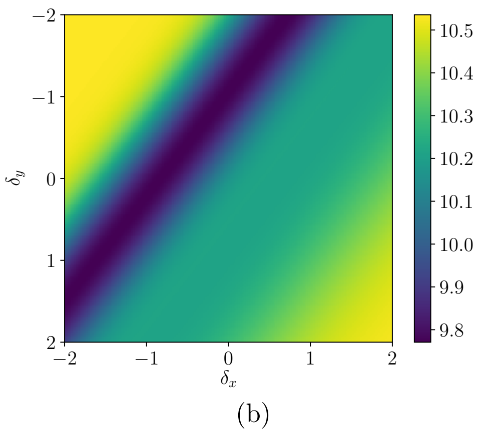


Figure 4.20.: Negative log-likelihood depending on δ_x and δ_y for a single observation as described in Equation 4.62. (a) and (b) show different scenes at different moment of an experimental trajectory. (a) shows a scene where an road sign observation is close to three referenced road signs. (b) shows a scene where a lane marking observation is obtained on a road with multiple referenced markings.

To better visualize the effect each observation has on the cost function being minimized, the function has been sampled for several values of δ_x and δ_y (δ_θ is kept null).

Figure 4.20 shows the negative log-likelihood depending on $\boldsymbol{\delta}$ for a single observation. With a single observation, several local minimum exist. When only a road sign measurement is considered, a local minimum will exist for every road sign referenced in the map. Similarly when only a single lane marking is considered, there will be a local minimum (or rather lines of local minimum) for every mapped marking (although only one is seen in Figure 4.20 (b) as markings are far apart).

Figure 4.21 and Figure 4.22 show how the negative log-likelihood changes as more observations are added. When all observations of the buffer are added (as described in Equation 4.58), the significance of local minimums decreases often leading to one clear solution. In some cases, however, as seen in Figure 4.22 (b), the cost function is ill-conditioned with a ambiguity along one of the directions. This is the case when only lane marking measurements are available. In the single dimension case, this problem did not arise because only the dimension in which there was no ambiguity was considered. To address this problem, adding the *a priori* distribution of $\boldsymbol{\delta}$ is useful. Figure 4.23 shows the result for the final cost function. Adding the *prior* distribution reduces the spatial invariance. This makes converging to a sensible $\boldsymbol{\delta}$ more likely.

The minimization problem is solved using the Broyden–Fletcher–Goldfarb–Shanno

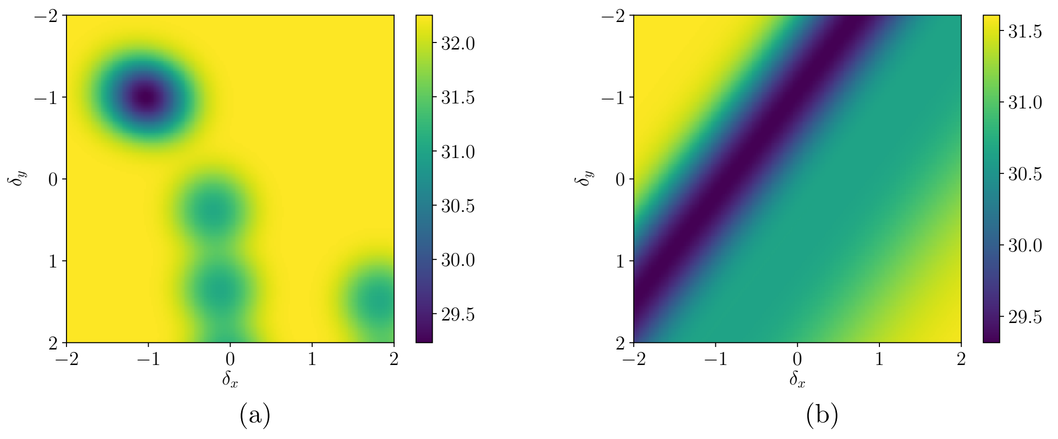


Figure 4.21.: Negative log-likelihood for three observations. While in Figure 4.20 (a) it was not possible to identify which minimum was correct, adding more observation makes one minimum stand out. In (b) there is no significant difference beyond the scale of the minimum. This is due to the three measurements probably corresponding to the same feature.

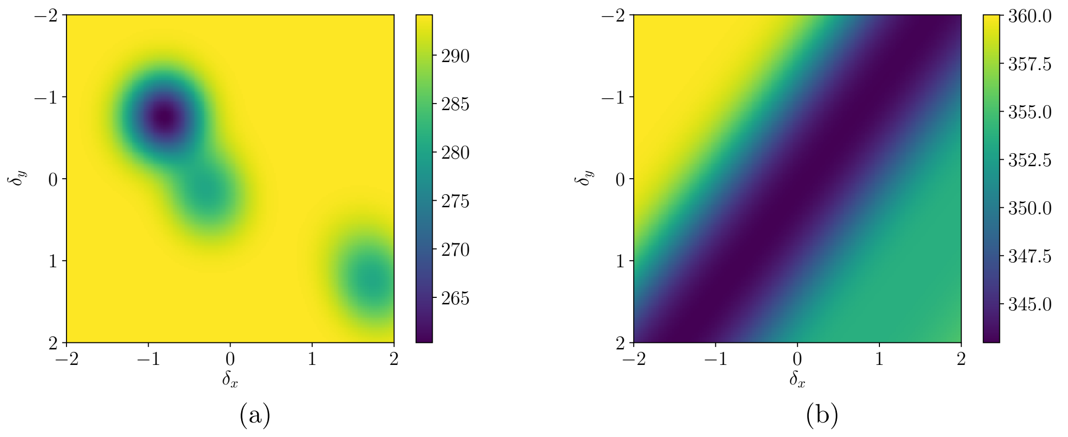


Figure 4.22.: Negative log-likelihood for all observations of the buffer. In (a), it can be seen that adding more measurements enables some of the local minimums to disappear leading to a more robust matching. Here, some local minimums remain due to the scarcity of measurements in that section of the trajectory. Moreover the minimum in (b) has shifted down to the right. Therefore, the first lane marking observation probably does not have a corresponding map feature.

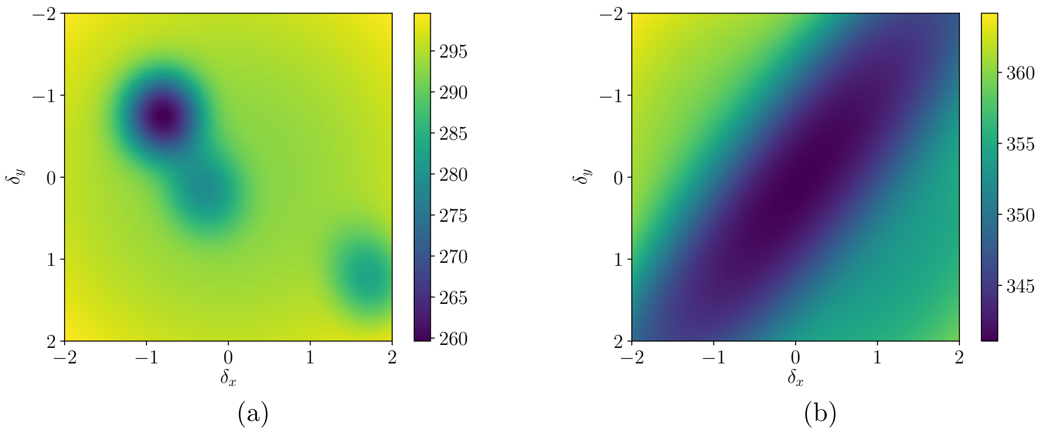


Figure 4.23.: Negative log-likelihood for all observation and the *a priori* distribution of $\boldsymbol{\delta}$. The main purpose of the *prior* distribution is to avoid a spatial invariance. While the effect when road signs are used (a) is marginal, for lane markings (b) it removes the spatial invariance that would have made finding a maximum difficult.

(BFGS) algorithm [Broyden, 1970, Fletcher, 1970, Goldfarb, 1970, Shanno, 1970] which is an approximation of the Newton method. The algorithm converges in fewer than 10 iterations in two thirds of the cases but a maximum number of iterations is also set such that the matching step always ends in time to provide matches.

4.4.2.3. Matching and Integration in the Filter

The adjusted trajectory is an improvement over the filtered trajectory. However, as quantifying the resulting uncertainty attached to it is not easy, it is only used to improve the matching of features. This matching is then used in the filter. Once the δ-adjusted trajectory has been found, it can be used to associate observations by applying methods presented in Subsection 4.2.1. The newly associated observations can now be used in the filtering scheme. Because the matching cannot always be performed within the filter update period, it is instead run at a higher period Δt_m. Therefore, when using the buffer pose adjustment method, road sign and road marking observations are not used directly to estimate the state. The state is estimated without these observations for some time before they are associated and can be used. The matching starts at time sample K using observations and state estimates from time $K - S$ to K. In parallel, the filter keeps estimating states without map feature observations to always get a real-time estimate. When the matching is done, the filter might have estimated states beyond time K. The filter is then run again starting at time $K - S$ up to the current time to account for the new matches that have been made.

4.5. Map-Aided Localization Experimental Results

The method has been evaluated on real data. The sensor data was recorded using an experimental Renault ZOE (see Figure 4.24), equipped with a u-blox M8T GNSS, an intelligent camera by Mobileye providing up to four simultaneous road-marking measurements at 3.7 Hz[1], and a Velodyne VLP-32C lidar used to detect road signs at 10 Hz using the threshold $I_{\min} = 230$ (where diffuse reflectors return 0 for no return, 100 for 100% return and retroreflectors return from 101 to 255, 255 being ideal reflection). The vehicle was also equipped with a Novatel SPAN-CPT that combines Real-Time Kinematic (RTK) GNSS and high accuracy IMU data to provide centimeter-level accuracy. This system is only used as a ground truth to evaluate the localization quality. The sensor data were recorded using the Robot Operating System (ROS) framework and all sensors except the camera were synchronized to the GNSS time. The non-accurate synchronization of the camera has not been found to be detrimental to the localization as these lateral measurements do not change significantly as the vehicle moves.

The data has been recorded on open roads through the commercial district of the city of Rambouillet, France. The trajectory used to evaluate in detail the method consists in several stretches of straight roads separated by roundabouts, as shown in red in Figure 4.24. Further results will be presented in Subsection 4.5.5 on the yellow trajectory and on trajectories recorded in Compiègne. The Mobileye camera was able to detect road markings only during the straight portions. The road sign detector provided measurements along the entire trajectory, although there were more road signs close to intersections and roundabouts.

The map used in this experiment is an HD map that references road markings as line segments and road signs as points in two dimensions. The map was built by a third party map provider and is expected to have centimeter level of accuracy.

Results are first presented for the one-dimensional matching strategy using lane markings only (no road sign measurements). Then, for the multi-dimensional approach, several configurations have been tested in these experiments. Results are presented when the two matching strategies (UNN and HG) are applied in real time. In this case, measurements are used as they are received and matched directly to be used in the next filter update. The same matching strategies are also applied after the method presented in this section has been used (Buffered UNN and Buffered HG). The observations are not used directly but are saved to be matched every Δt_m. The entire matching step has been found to take 31 ms on average on an Intel i7-7820HQ processor. Δt_m was chosen high enough (250 ms) such that the matching always ends within this period. In both cases, the localization is performed at 50 Hz.

[1]Original frequency is 37 Hz. Only one out of ten measurements is used to limit the effect of the temporal correlation of these measurement errors.

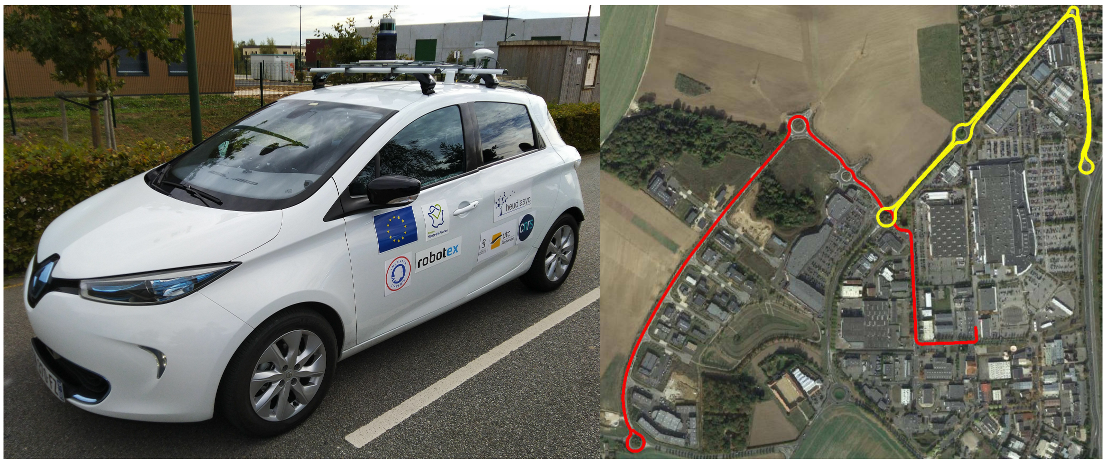


Figure 4.24.: (a) Experimental vehicle with GNSS, lidar and camera. (b) Trajectory used to test the algorithm (red: Rambouillet 1, yellow: Rambouillet 2).

The buffer length is chosen at 5 s ($S = 250$), the reason for this choice is explained in section Subsection 4.5.4.

The following subsection details the availability of observations when the proposed matching strategy is used. The effect of the matching on the localization consistency is then presented. Since the matching strategy uses buffers, the influence of the size of the buffer is studied. Finally, more extensive localization results are presented using trajectories recorded in Compiègne.

4.5.1. Proportion of correct lane marking matches

The number of correct matches has been studied for the one-dimensional matching strategy. The approach using the adjustment is compared to the Nearest Neighbor (NN) matching strategy. This strategy is the most commonly used for such matching problems [Ghallabi et al., 2019a, Tsuchiya et al., 2019, Leonard et al., 1992].

There is no direct way to obtain the ground truth matches. Here, the matches obtained using nearest neighbor matching with the ground truth states are used as ground truth of the association.

For the NN method the maximum matching distance allowed is set as in Equation 4.4 with a risk $\alpha = 0.05$. For the matching with prior adjustment by δ_y, the maximum δ_y is chosen at $\Delta_{max} = 1$ m and the maximum residual is limited at $\epsilon = 0.5$ m. Matches that are rejected by one of the rejection steps are not counted. The scale

Table 4.1.: Comparison of the numbers of wrong and correct matches by adjusting the observation and using NN only.

	Wrong	Correct
Adjusting by δ_y	66 (33%)	133 (67%)
Nearest Neighbour	84 (43%)	112 (57%)

used for the gradient descent is chosen empirically at $\gamma = 0.2$ and the iteration stops when δ_y improves by less than $\nu = 0.001$ m.

Table 4.1 shows the numbers of wrong and correct matches for the two compared methods. These results are obtained using the first drive of the Rambouillet dataset (see Appendix A). Correct matches are especially hard to get with this trajectory. The sides of the road have multiple ground features in close proximity. The distances separating features being of the same order of magnitude as the sensor noise, finding correct associations is difficult. For that reason, both approaches have a high rate of wrong matches. Using the δ_y adjustment, the amount of correct matches is noticeably improved.

4.5.2. Observations Availability

The number of observations that can be matched to features directly affect the localization accuracy. As expected, increasing the rejection factor α (defined in Equation 4.4) results in fewer observations for every configuration. However, our method performs better as the adjustment step reduces the distances to features and therefore allows matching more map features that will be used in the filter. Moreover, since the number of matches remains almost constant until the rejection rate reaches $\alpha = 0.7$ (see Figure 4.25), its value can be set high which limits the chance of bad associations.

The availability of road marking feature matches is similar for every method. The UNN approach performs slightly worse than the Hungarian association as it only matches if an observation is closest to the features. When two side-by-side ground features are detected, the estimation error can cause the UNN approach to only match a single observation while the Hungarian approach can match both, see Figure 4.1. Using the proposed method, the advantage of using the Hungarian method is reduced as the adjustment step compensates for the estimation error.

Please note that the road sign matching is greatly improved using the observation buffer and the adjustment step. The adjustment step enables to reduce the observation residuals used for the matching, limiting the effect of the non-linearity of the model. Also, because all observations contained in the buffer are being matched,

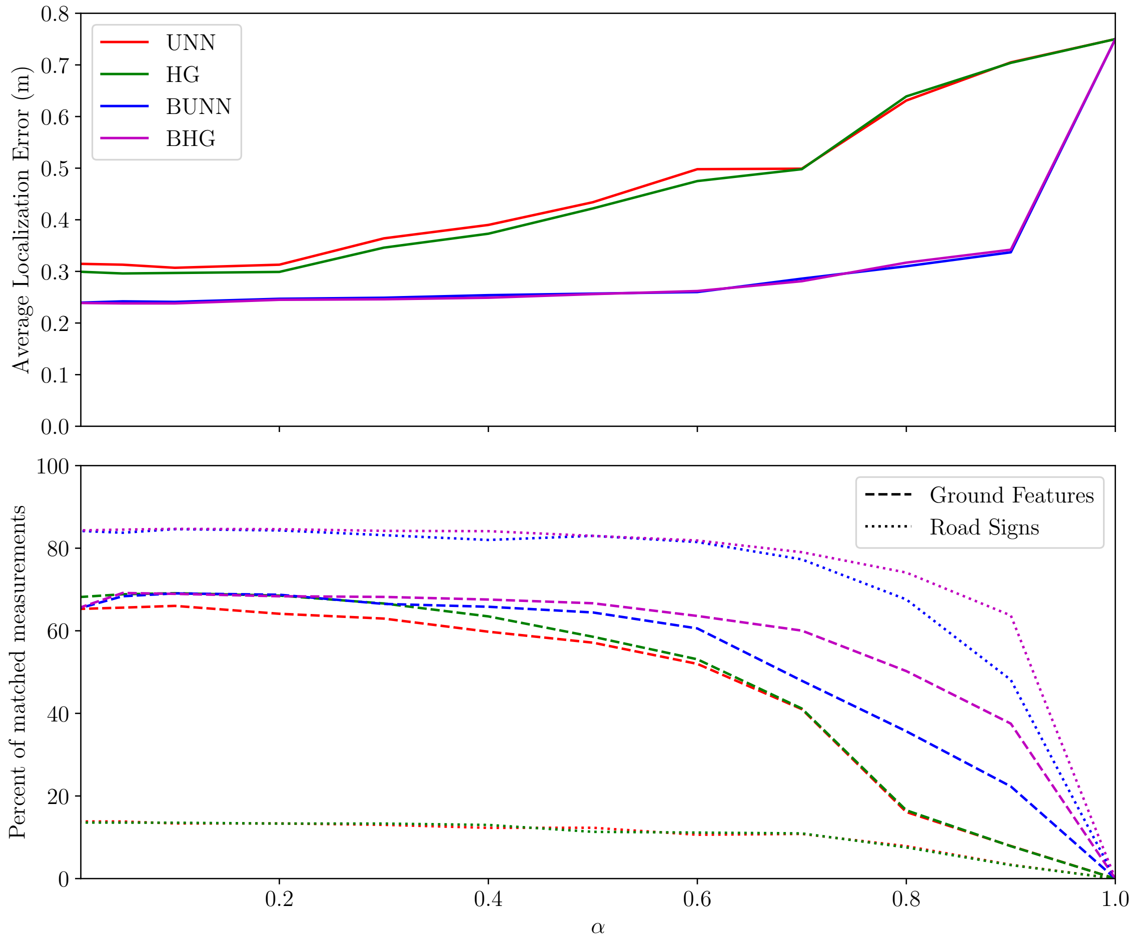


Figure 4.25.: Localization error and observation availability depending on the rejection rate α. Results are given for the UNN, Hungarian (HG), Buffered UNN (BUNN) and Buffered HG (BHG). The solid lines show the localization error. The dashed line (road markings) and dotted lines (road signs) describe the percentage of matched features.

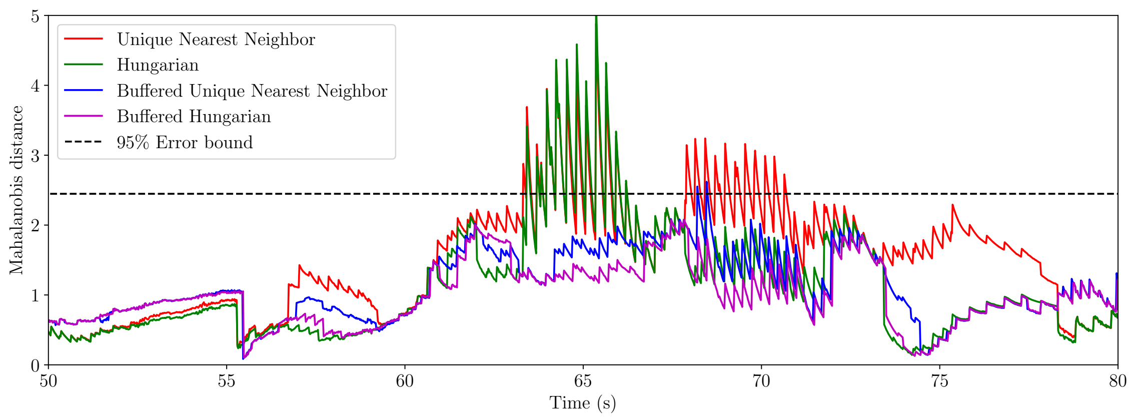


Figure 4.26.: Mahalanobis distance for the four configurations during a difficult matching (scene similar to the one shown in Figure 4.18 and 4.19). It shows the covariance weighted error (without unit) over a 30 s stretch of the test trajectory for the four tested configurations. Samples above the threshold line pose a risk of loosing the consistency of the localization. The results are presented for a rejection rate of $\alpha = 0.05$ for every configuration.

observations that could not have been matched earlier might be matched in the following iterations of the matching process. For road signs association, the Hungarian method does not provide significant improvement compared to UNN.

4.5.3. Localization Accuracy and Consistency

If the matching can be correctly done, a higher number of observations will result in a greater accuracy. This is true for all configurations, as the rejection rate α is chosen bigger, the number of matches decreases and the localization error increases. The observation error increases steadily when the UNN and Hungarian method are used alone. Because of the adjustment step, the number of matches for the buffered configurations only starts to decrease for high rejection rates. The localization error follows a similar pattern: it increases slowly until $\alpha = 0.6$ at which point the number of matches drops and the error increases.

When the matching is not buffered, the Hungarian method results in errors smaller by around 2 cm. Its ability to associate more road markings explains this improvement. However, when matching using an adjusted trajectory, the two methods are not noticeably different whichever rejection risk is chosen. Hence, the UNN method is preferred as it is computationally less expensive to run.

Configurations without adjustment are less accurate because of the fewer measurements that can be matched. To increase the number of matches, a solution would be to lower the rejection rate. This, however, increases the chance of erroneous matching. This has been evaluated by looking at the Mahalanobis distance $\sqrt{\left(\hat{\boldsymbol{x}}_{k|k} - \boldsymbol{x}_k\right)^\top P_{k|k}^{-1} \left(\hat{\boldsymbol{x}}_{k|k} - \boldsymbol{x}_k\right)}$ for the different configurations. As shown on Figure 4.26, for rejection rate lower than 0.05%, a road sign is incorrectly matched greatly affecting the consistency of the state estimate. Although decreasing the rejection rate α should increase the number of observations thus increase the accuracy, it also results in bad matches for methods UNN (red) and HG (green). Since consistency is very important for the integrity of localization for autonomous vehicle navigation, we believe that snapshot matching approaches are not well suited in this context.

Given the localization for different rejection rate, α is chosen at 0.5 such that the maximum association distance is as small as possible while still keeping a margin to avoid risking the loss of accuracy occurring for $\alpha > 0.6$. With this value, the localization error averages at 0.28 m and has not been found to go beyond 1 m at any point of the trajectory.

4.5.4. Buffer Size Influence

Our method has two components that contribute to improving matching. The optimization step enables to reduce residuals, making matching easier thus increasing the number of observations. The second component is the use of past observations in the optimization step. Indeed, the adjustment step is dependent on the number of available observations. If the optimization is performed in a snapshot fashion using only the current observations, the optimization converges toward the closest map feature to the observations. This would not help the matching as when a single road sign is observed, it would be matched to the closest feature. Hence, erroneous matchings would occur. Using a buffer of observations provides a more detailed picture of what is observed, thus constraining the optimization and preventing the most recent measurements to be the sole influence on the matching.

The length of the buffer affects the number of observations considered. A longer buffer should therefore result in fewer ambiguous matches. The buffer should, however, be small enough as the smoother state estimates are transformed in a rigid manner by the optimization, see Equation 4.55.

Figure 4.27 shows the localization error of the vehicle for different buffer lengths. It can be seen that buffers shorter than 3 s are not sufficient as outliers start to appear. This is due to the lack of unambiguous measurements in small buffers to compensate for ambiguous ones. In this case, the error is due to an erroneous road sign matching in a situation similar to that described in Figure 4.18. With small buffers, there are not enough measurements to constrain the adjustment in orientation and laterally and the observed road sign is matched to the wrong feature (the observed sign was not referenced in the map).

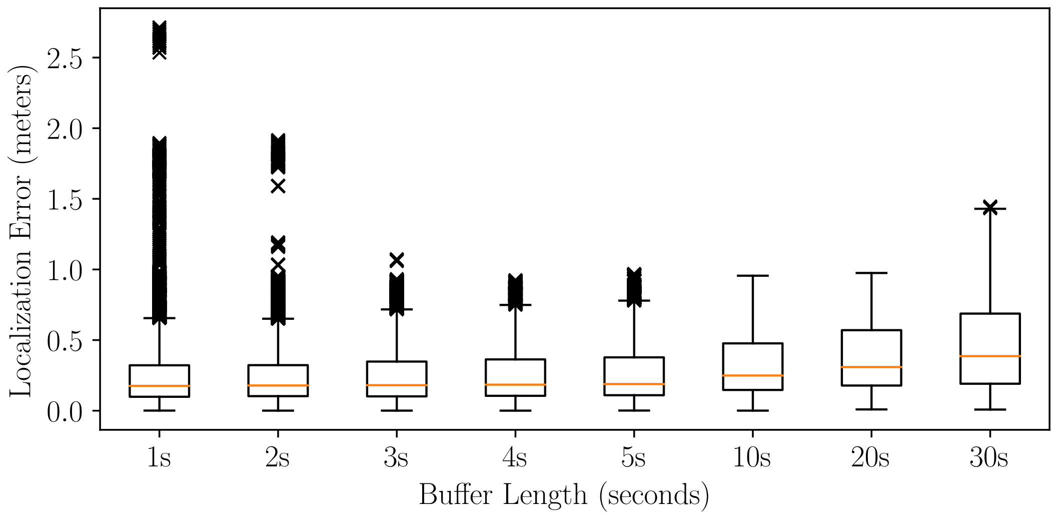


Figure 4.27.: Distribution of the localization error depending of the buffer size (the rejection rate is chosen at $\alpha = 0.5$). Buffers too short lead to ambiguous associations thus larger errors. Additionally, a rigid transformation cannot properly adjust the trajectory leading to fewer matches and decreased accuracy.

Longer buffers enable to better adjust the trajectory but to a point. For buffers longer than 10 s, the localization error starts to increase. This is the result of the assumption made for the adjustment that the trajectory only needs to be moved rigidly. Hence, too long buffers result in higher residuals after adjustment. This causes more observations to be rejected, finally affecting the localization error. For a rejection rate of $\alpha = 0.5$, a buffer length of 5 s has been found to be sufficient to enable unambiguous matches while still providing enough measurements.

4.5.5. Evaluation Using Different Datasets

To further evaluate the performance of the proposed approach, it has been tested on another recording done in Rambouillet and on three other experiments carried out in Compiègne obtained with a similarly equipped vehicle. The five trajectories amount to 21 km of roads (50 minutes).

Table 4.2 contains the average localization error for different datasets. It can be seen that in all cases the proposed method performs better (by 16% on average) although sometime marginally (Compiègne 3). The parameters used for the buffered method were kept the same as those identified with the first Rambouillet trajectory. Even with a low α, the UNN and HG methods rarely achieve accuracy similar to our approach. These results also show that optimized parameters on a particular

Table 4.2.: Average localization error (in meters) for each method tested on different trajectories.

	BUNN	UNN		HG	
α	0.5	0.05	0.5	0.05	0.5
Rambouillet 1	**0.25**	0.31	0.42	0.3	0.43
Rambouillet 2	**0.68**	1.25	2.75	1.26	2.75
Compiègne 1	**0.75**	0.94	0.97	0.88	0.96
Compiègne 2	**0.57**	0.65	0.73	0.60	0.73
Compiègne 3	**0.77**	0.78	0.83	0.78	0.82

Table 4.3.: Computing times for the tasks of the localization system. The times are obtained using an Intel© Core™ i7-7820HQ processor with 16GB of memory.

	Mean	Min	Max	3rd Quartile
High-Frequency Estimation	0.237 ms	0.034 ms	5.672 ms	0.222 ms
Smoothing (Matching Layer)	0.398 ms	0.030 ms	2.057 ms	0.509 ms
Adjustment (Matching Layer)	25.098 ms	0.198 ms	201.311 ms	36.344 ms

trajectory lead to a good accuracy in different environments which shows a good robustness of the method.

4.5.6. Computational Load of the Localization System

One of the main requirements of the localization system is to operate in real-time with low delays. The computation times of the most straining tasks of the localization system have been studied. Table 4.3 shows statistics of the filtering task performed at high frequency, and of the smoothing and adjustment tasks performed at low frequency.

The program is implemented in C++ using the ROS Noetic framework. The Eigen library is used to perform matrix operations. The program is running on a laptop with an Intel© Core™ i7-7820HQ processor (2.9 GhZ, 4 cores, 8 threads). The system is also equipped with 16 GB of RAM.

The high frequency task being performed at 50 Hz, the estimation has to be done within 20 ms. Here, the filtering task is performed in the vast majority of cases within a single millisecond as shown by the third quartile. The most straining events for the high frequency estimation are when the *Matching Layer* finishes and old measurements become observations. In those instances, the high-frequency filtering has to re-compute more states. The maximum duration of the high frequency estimation is 5.672 ms which is still small. If that delays is detrimental to other systems of the autonomous vehicle, the future state could be predicted in advance to be provided without delay. This would sacrifice some accuracy but would ensure that the localization solution is provided without delays.

The *Matching Layer* has two main tasks to perform to be able to find matches. It first needs to smooth the entire state buffer, then it needs to adjust the trajectory in order to improve matching. While the filtering step of the *Filtering Layer* estimate only a few states at a time, here the smoothing step has to smooth the entire 5 s buffer. Despite the computational complexity of this step being constant there is still a lot of variance in the processing time. It is performed in almost no time to about 2 ms which, considering that the smoothing is done on the entire buffer, is much faster to do than filtering. The most expensive task that the *Matching Layer* has to perform is the computation of the adjustment δ of the trajectory. The length of this step depends a lot on the number of observations contained in the buffer and on the number of map features in the local map. The two steps have to be done within the $\Delta t_m = 250$ ms period of this layer. The maximum time taken is below that threshold. This is a result of the maximum number of iterations allowed to the BFGS algorithm.

These results show that the architecture that has been studied is very efficient and that a real-time implementation is very feasible.

4.6. Conclusion

Matching measurements to the right features is an essential step to reach accurate localization with high integrity. Matching is a difficult process as state estimation errors and small number of measurements contribute to making matching ambiguous.

The method detailed in this chapter aimed at improving this process by using a buffered of observations and an adjustment step to address these issues. By matching measurements in buffers rather than in a snapshot manner, the matching has a better representation of the environment, limiting ambiguities in matching. Adjusting the trajectory prevents the state estimation error to result in incorrect matches.

The results have shown that both the number of matches that can be achieved and the accuracy of the localization are improved. Moreover, matches are more reliable as cases with ambiguous matches can be prevented with the buffered method.

The matching strategy for the one-dimensional case was presented at the Intelligent Vehicles Symposium (IV) [Welte et al., 2019b]. The extension to the multi-dimensional case has been published in the Robotics and Automation Letters (RA-L) [Welte et al., 2020] and presented at the International Conference on Intelligent Robots and Systems (IROS).

5. HD Map Error Detection and Mitigation

Contents

5.1. Introduction

Even though using maps for localization enables great improvement in terms of localization accuracy, particular attention is required to prevent sacrificing integrity. Map features are expected to be accurately referenced when they are used for localization. However the map accuracy cannot be guaranteed in the long term. Indeed, roads and their surroundings change (albeit slowly). Regardless if the map was built automatically or by careful surveyors, eventually every map will have errors.

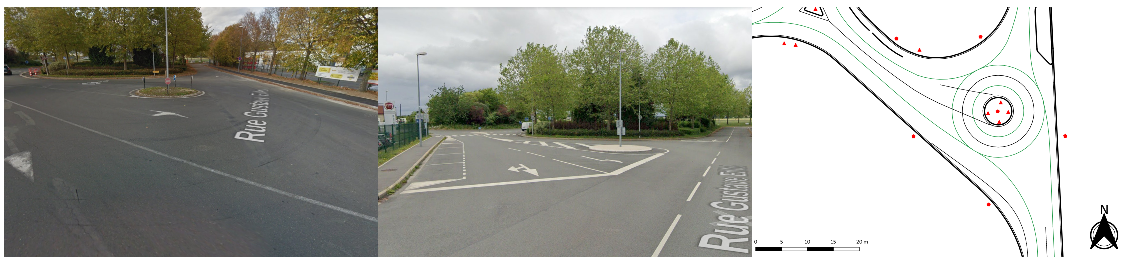


Figure 5.1.: Comparison of Street View images of an intersection in Rambouillet at the time of mapping (a) to a few months later (b) and the corresponding HD map of the area (c)

Road networks change faster than one might expect. In fact the two test sites used in this thesis, Rambouillet and Compiègne, which had their maps built respectively in 2018 and 2020, already contain errors at delivery (as shown in Figure 5.1). In both cases, intersections have been modified right after the mapping was performed. This has resulted in maps with missing features and others that no longer exist. In the case of Rambouillet, the mapping was also done while road work was going on, leading to incorrect referencing of some lane markings. In both maps, some features have also been found to be incorrectly referenced or moved most likely due to accidents.

This suggests that the localization system needs to be able to deal with such errors. Therefore, it either has to be robust to errors during the localization process or be able to monitor the accuracy of map features.

Map errors can be separated into multiple categories:

- Missing features: not referenced during the mapping or did not yet exist when the mapping was performed.
- Old features: existed at the time of the mapping but no longer exist.
- Moved features: the reference of the features has changed since mapping, e.g. road markings redrawn.
- Misclassified features: the type of the feature referenced in the map is not correct anymore, either because of an error during the mapping, or the type as changed since then, e.g. yield sign changed to a stop sign.

Although all error types can be detrimental to localization, the change in a feature reference is the most critical. Indeed missing features will simply not be matched and therefore will not affect localization (although they could become a problem if faulty measurements coincide with these errors). Similarly, old features are not detectable. Misclassified features can make the matching harder. In this work, the detection system does not provide information on the type of feature or the information is not reliable. Hence, the class of the feature is not used in this localization system. The type of error this work addresses are moved features. Moved features are particularly harmful to the localization since all vehicles using the faulty map will have localization error when the faulty feature is detected.

In this work, a post-processing strategy is developed to evaluate map features. Using estimations and observations obtained during a drive, the map features are evaluated based on their residuals. Based on these residuals, two approaches have been developed. The absolute value of the residuals, lane markings will be weighted differently. Higher residuals will be used to weight down the influence of those lane markings in the localization. The residuals and their covariance matrix have also been used to detect errors in road signs. By fusing residuals from multiple trajectories, faulty road signs are detected to be discarded from the estimation.

The following section presents an overview of existing map error detection methods. The two methods using absolute post-smoothing residuals and covariance weighted residuals are then detailed and experimental results are presented for each.

5.2. State of the Art

The widespread use of maps for intelligent vehicles has led to questions about the accuracy of these maps [Paul and Wan, 2008] and to the development of methods to detect and correct map errors. There are different types of map errors leading to widely different strategies to detect them.

The main element of the map that needs to be accurate for intelligent vehicles is the general road geometry. The road geometry changes slowly as there is no easy way to affect it. It can only be affected by road work. Errors on this aspect of the map would have disastrous effects for control and planning systems.

[Hartmann et al., 2014] have trained a neural network to detect maps errors. To detect errors, the network is fed with the difference between observed road characteristics (distance to lane markings, curvature, etc.) and characteristics obtained from a map. With this information, the network outputs a probability of error. The authors show that this approach is able to detect errors such as missing roundabouts or intersections. It is also important to note that unlike many map error detection methods, this approach is able to detect changes before the vehicle goes through an area using a front-facing camera. As intersections across Europe are increasingly being converted into roundabouts to improve traffic flow, it is important to detect when an intersection is changed. [Zinoune et al., 2012a] have managed this by studying a buffer of vehicle pose. Using the trajectory profile on the x and y axis as a descriptor of the buffer, the authors classify the trajectory within a finite set of classes (composed of multiple roundabout and non-roundabout classes). After a new roundabout is detected, parameters of the roundabout are estimated to update the map. The same authors [Zinoune et al., 2012b] have also developed a method using a Page test to check whether the road geometry of standard definition maps has changed or not. For this they use the Page test on the difference between the map matched vehicle pose and the estimated vehicle pose. This method enables to detect where the map has changed and to figure out where the change starts and ends.

Although some have used the general road geometry for localization purposes (e.g. in [Fouque et al., 2008]), specific accurately referenced features are more useful for localization. Hence, research has also been done to address this problem.

Point clouds are also used as georeferenced data in so-called dense maps. Even features maps used in this work have been produced by cartographer using dense point clouds of the area. Hence, detecting map changes can be made using this format. [Hyyppa et al., 2009] compare point cloud maps using a point to point approach. Points from one map version are classified based on their distance to points from another map version. New points are far from any points of the old map and points from the old version no longer existing are too far from any point in the new map. This method requires the original dense map format to be kept which, as mentioned in Subsection 4.3.1, is problematic as such maps are heavy.

In this work, the detection of errors in feature maps is studied.

High accuracy feature maps are fairly recent, the literature on changes detection in those maps is scarce. Nevertheless, similarities can be established with existing literature on change detection in Simultaneous Localization And Mapping (SLAM) applications.

[Pannen et al., 2019] have addressed the HD map change detection problem using a dual particle filtering approach. Two particle filters are used, one estimate the vehicle state without using the map, the other uses lane marking measurements. From both sets of particles, descriptors such as the mean lane marking innovation and the mean particle weights are computed. Features are classified based on these descriptors using thresholding techniques.

[Nguyen et al., 2016] have also used an estimation with and without map features to figure out if the map is reliable. They compare estimates obtained using map features for localization to dead-reckoning estimates. A random forest classifier trained to classify features as reliable or not is then used to infer the features reliability.

Evaluating map features is not only limited to evaluating the accuracy of its reference. The existence of the feature might also be in question. Also adding new features to the map is not straightforward as some feature might not actually be static, e.g. vehicles license plates are often detected by the method presented in Subsection 4.3.3 but should not be stored as map features. [Jo et al., 2018] have proposed an approach using Dempster-Shafer theory to deal with this issue. When new features are detected, they are initialized with *unknown* existence. At every new observation of the feature, its mass shifts toward *existence*. Inversely, features that have not been observed in a while will see their mass shifts from *existence* to *deleted*. In this framework, features that have moved will be classified as *deleted* and a new feature will be added at the correct position.

While detecting faulty features is important for localization, some have taken the different approach of detecting features that are valuable for localization, rather than those that might hinder on localization. In [Berrio et al., 2019b], the authors use descriptors of landmarks such as the number of measurements, range by which the feature is visible, the density of features in a particular area, etc., to evaluate the importance of a particular feature in the map. This enables the authors to reduce the size of their map to keep only useful features while limiting the inevitable loss of accuracy. This work is applied to pole and corner features. For intelligent vehicles, removing features of the map might not be a good idea since features might be needed for other applications than localization. Road signs, for instance, are useful features for localization that cannot simply be removed. However, it is not inconceivable to include in future maps a usefulness layer indicating if features should be used or not for particular applications. [Berrio et al., 2019a] have also looked at whether features are even visible to the sensors. Maps do not need to store information that cannot be observed by the vehicle sensors from a localization perspective. This was not a problem when maps were predominantly built using SLAM techniques using the same sensors set. As third party HD map providers start to develop, it will become significant.

5.3. Map Features Attenuation using Smoothed Residuals

5.3.1. Method

The detection of map errors depends on the matching process. Indeed some map errors are large enough to make the matching system ignore the measurement. These errors will not affect the localization. The map errors leading to errors in the localization are the ones that are small enough to result in measurements being matched to them thus influencing the localization solution. Such errors are too small to be detected in real time. To detect such errors, post-processing or multiple trajectories are required.

Here the proposed approach is to evaluate features based on their residuals. Faulty features have high residuals. Using residuals the features are not discarded but rather are weighted down in future estimations.

In the following method, the post-smoothing residuals are used to evaluate lane markings. The post-smoothing residuals are used to compute a reliability factor for each observed features that is used to weight the observation in future estimation.

5.3.1.1. Residuals Computation

After a drive, each map feature $\boldsymbol{m}_i$ will have a set of M_i observations $\mathcal{Z}_i$ attributed to it. For each observation z_k^j, the post smoothing residuals $y_{k|N}^j$ are computed following Equation 5.1

$$y_{k|N}^j = z_k^j - h_i\left(\hat{\boldsymbol{x}}_{k|N}\right) \tag{5.1}$$

The residuals obtained from all the observations of $\mathcal{Z}_i$ are then averaged to obtain a single value for each map feature

$$\bar{y}_i = \frac{1}{M_i} \sum_{\mathcal{Z}_i} y^j_{k|N} \tag{5.2}$$

5.3.1.2. Weighting Factor

In Chapter 4, the uncertainty of a lane marking observation was identified as $\sigma_{LM} = 0.1 \times C_0$, where C_0 is the measured distance to the marking. To account for unreliable markings, the uncertainty used in the estimation is increased. Previously, the map was assumed to be perfectly accurate. The accuracy of the map is now taken into account for the fusion.

The variance of a lane marking measurement is redefined as

$$\sigma^2_{LM} = \sigma^2_z + p_i \cdot \sigma^2_{map} + (1 - p_i) \cdot \sigma^2_{bad} \tag{5.3}$$

where σ^2_z is the variance of the measurement itself as defined in Chapter 4. Two extra terms are added to account for the map accuracy. The first term is the nominal variance of the map σ^2_{map}. When no information on a marking exists, it is assumed accurate and has a variance of σ^2_{map}. If the marking is found to be unreliable, it will instead be attributed a variance of σ^2_{bad} (chosen arbitrarily high at 1 m^2). The variance of a marking varies between these two values based on a factor $p_i \in [0, 1]$.

The factor p_i is chosen so that when $\bar{y}_i = 0.0$, the factor tends to 1 and when $\bar{y}_i \to +\infty$, the factor tends to 0.0. Hence, as illustrated in Figure 5.2 a lane marking producing small residuals will be considered as reliable and be used in the estimation with the nominal map variance. A lane marking producing high residual will be weighted down in the estimation through an increase of its associated variance.

To achieve the expected result, the value of p_m is chosen for each marking using the following equation,

$$p_i = \exp\left(-\frac{\bar{y}_i^2}{\alpha^2}\right) \tag{5.4}$$

where α is a tuning parameter affecting the size of the errors to consider. Intuitively, features producing high residuals higher than α will have their effect on the state estimation reduced.

5.3.2. Experimental Results

This method was tested using drives performed in Rambouillet. Three markings of the map of Rambouillet used in these experiments are known to be incorrectly referenced because of road works being done during the mapping. This error is on a straight road oriented north-south and concerns the marking separating both lanes.

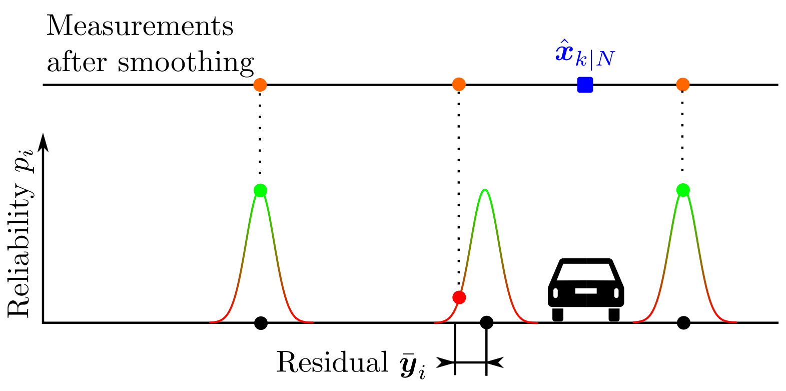

Measurements
after smoothing
$\hat{\boldsymbol{x}}_{k|N}$
Reliability p_i
Residual $\bar{\boldsymbol{y}}_i$

Figure 5.2.: Attribution of the reliability p_i based on $\bar{\bar{y}}_i$. The top row shows the observations (orange dots) relative to the smoothed state. The residuals between observations and the map (black dots) are used to compute the reliability.

The actual marking is located roughly 0.5 meter west of the mapped marking. To our knowledge, this is the only marking that is erroneous along the testing path.

In these experiments, the localization system only uses dead-reckoning sensors, lane marking measurements and GNSS positions. No road sign measurements are used at this stage. The lane marking measurements are matched to map features using the matching strategy developed in Subsection 4.4.1.

A first drive is used to estimate the reliability of the lane markings. During this drive the lane markings are assumed to be accurate (all p_m are at 1). Once the reliability of the observed markings has been obtained, the localization is performed on a second drive using the computed reliability factors. The tuning parameter α is chosen equal to 0.3 meter.

5.3.2.1. Reliability Computation

Figure 5.3 shows the map of the estimated reliability of lane markings around the erroneous section of the map and. The central lane marking known to be inaccurate is correctly identified as unreliable (red), while most of the other lane markings are found reliable. Figure 5.4 shows the distribution of the reliability factor p_i. Several correctly referenced lane markings are also found unreliable. The markings mainly correspond to markings around intersections. This is most likely due to the camera measurement being unreliable in curves rather than the map being unreliable. In this work, it is assumed that the error is due to the map rather than the measurements. To unambiguously identify the source of the error more redundancy is required in the sensors used which is not available at this stage.

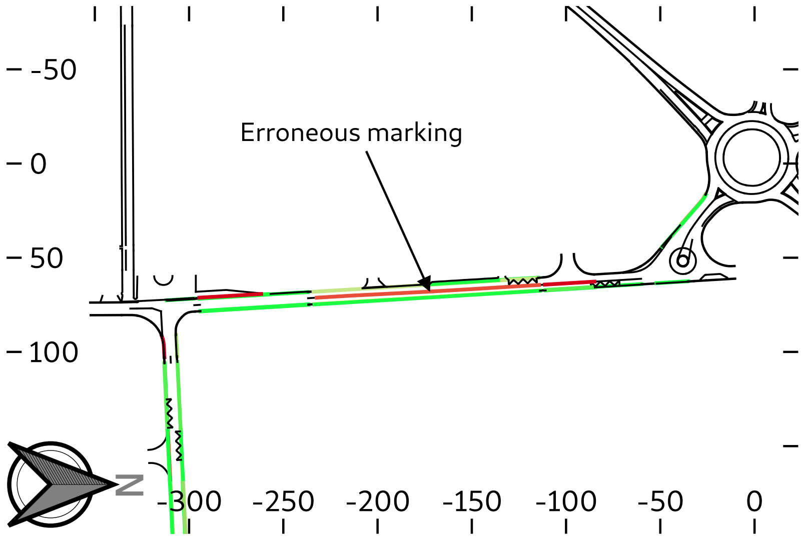


Figure 5.3.: Representation of marking reliability (black uncategorized, green most reliable, red least reliable).

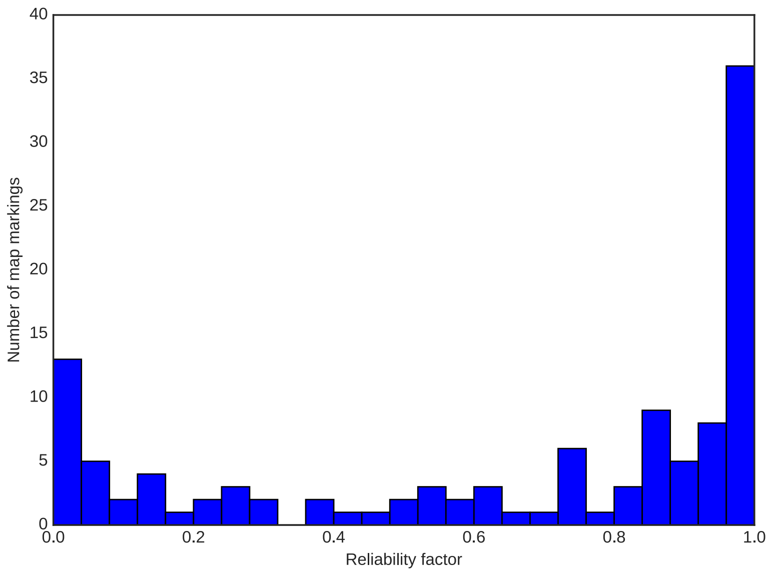


Figure 5.4.: Distribution of the reliability factor p_i.

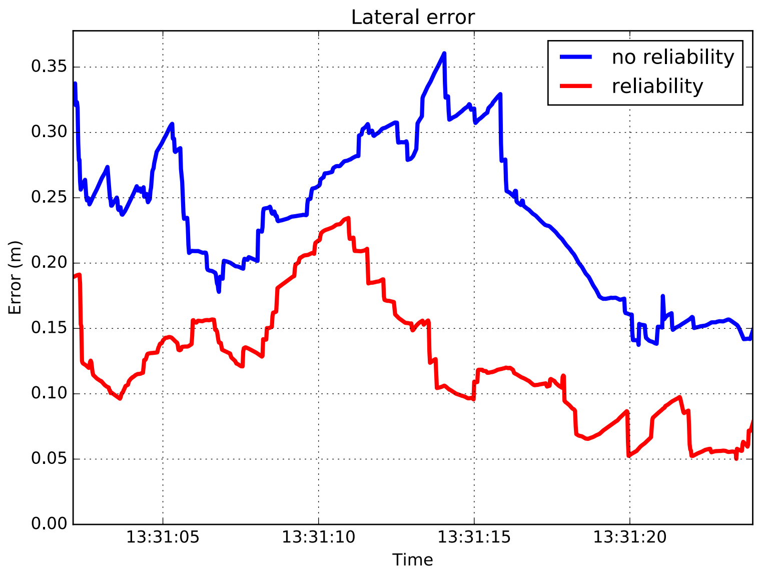


Figure 5.5.: Lateral localization error on the road section with the erroneous lane marking. No accounting for the lane marking reliability (blue) the localization error is higher than with the proposed method (red)

5.3.2.2. Effect on Localization Accuracy

After having used a first drive to compute the reliability of lane markings, a second drive is used to evaluate the localization. Two estimations are performed, an estimation without accounting for lane marking reliability and an estimation weighting the lane markings uncertainty based on the reliability factor. Because the method is implemented within the ROS framework, small message transmission delays caused by ROS lead the estimation to be slightly different when the same program is run multiple times. The program is therefore run five times for each configuration and the results are averaged to limit the effect of the ROS framework on the results.

Figure 5.5 shows the lateral localization error when the erroneous lane marking is being observed. In this section multiple lane markings are observed. Thus, even when not accounting for the erroneous lane marking, the localization error is not entirely biased by the 0.5 meter error of the central lane marking. But accounting for the unreliable lane marking further improves the localization estimate leading to errors below 20 cm in most cases.

These results have been presented at the Intelligent Vehicles Symposium (IV) [Welte et al., 2019b] along with the one-dimensional matching strategy of the previous chapter.

5.4. Sequential Residuals Update for Maps Error Characterization

Although the aforementioned method has shown to improve localization, the way it mitigates observation is purely empirical. The function used to convert residuals into sensible weights does not derive from theoretical considerations. Weighting observation is not a simple task. However eliminating map features can be done with a probabilistic justification.

The feature error detection described here considers residuals as observations of the feature errors. Residuals are supposed to follow Gaussian distributions centered on the origin. However, this is only true if the mapped feature is correctly referenced. If it is wrong, the residuals will be centered on an unknown non-zero value.

With that in mind, the erroneous features will generally have higher residuals than correct features for the same uncertainty. This can be leveraged by eliminating features whose residuals have Mahalanobis distances too high. Indeed by performing a χ^2 test on the residuals of a feature, it is possible to detect erroneous features. While correct features will be rejected with the selected risk factor α, erroneous features will be rejected at a much higher rate.

5.4.1. Method

5.4.1.1. Computing Residuals

To evaluate map features, residuals are used. Map features can be observed multiple times. For a single trajectory, multiple residuals are computed. After having smoothed the state estimates, the residuals of each observation are computed as in the previous method. Unlike the previous method, here the uncertainties of the residual are considered. The residuals and their covariance matrix [Gibbs, 2013] are computed as,

$$\boldsymbol{y}_{k|N} = \boldsymbol{z}_k^j - h_i\left(\hat{\boldsymbol{x}}_{k|N}\right) \tag{5.5}$$

$$\boldsymbol{S}_{k|N} = \boldsymbol{R}_k - \boldsymbol{H}_k \boldsymbol{P}_{k|N} \boldsymbol{H}_k^\top \tag{5.6}$$

with $\boldsymbol{R}_k$ being the covariance matrix of the observation, $\boldsymbol{H}_k$ the Jacobian of the observation model and $P_{k|N}$ the covariance matrix of the smoothed state estimate.

It is important to note some observation models are not suited to be used with this method. This method combines multiple residuals from multiple observations and multiple drives. This can be done only if the residuals can be compared. In Chapter 4 the observation model chosen for road sign measurement was chosen such that the observation (and thus the residuals) are expressed in the global reference frame. With this observation model, all residuals can be interpreted as the road sign error in the global frame. An observation model where the observation is expressed in each road sign local reference frame (with axes oriented along and orthogonal to the road sign) would also work as this frame does not change from an observation to the next. However, an observation model with the observation being the coordinates of the road sign expressed in the sensor frame would not work. Indeed with such a model, the residuals would change depending on the point of view of the vehicle. It is for that reason that this method is not applied to lane markings. The lane marking observation model as stated in Chapter 4 would result in residuals dependent on the point of view (e.g. positive residuals when observing a marking driving east to west but negative residuals when driving west to east).

At the end of a drive, a subset $\mathcal{M} = \{\boldsymbol{m}_i\}_{i\in[\![0,M]\!]}$ of map features have been observed. For every map feature $\boldsymbol{m}_i$, a number of residuals have been obtained $\{\boldsymbol{y}_{k_0}, \boldsymbol{y}_{k_1}, \dots, \boldsymbol{y}_{N_i}\}$ (where N_i is the number of residuals for feature $\boldsymbol{m}_i$). These residuals are correlated. Indeed, they are all computed using smoothed state estimates that are themselves correlated. Therefore, the residuals of features of the same trajectory are correlated.

5.4.1.2. Combining Residuals

To decide whether a feature should be used or not, an aggregate residual is used. Because the residuals are correlated standard fusion method cannot be used. Instead, the aggregate residual is obtained using a covariance intersection strategy. Covariance intersection enables fusion of multiple information with unknown correlation. It consists in a linear interpolation between the different sources of information weighted to minimize the resulting covariance matrix. In this application, a feature $\boldsymbol{m}_i$ as the N_i following residuals

$$\boldsymbol{y}_{k_0}, \boldsymbol{y}_{k_1}, \dots, \boldsymbol{y}_{N_i} \tag{5.7}$$

with corresponding covariance matrices

$$\boldsymbol{S}_{k_0}, \boldsymbol{S}_{k_1}, \dots, \boldsymbol{S}_{N_i} \tag{5.8}$$

The aggregate residuals are expressed as

$$\boldsymbol{S}_i^{-1} = \omega_{k_0}\boldsymbol{S}_{k_0}^{-1} + \omega_{k_1}\boldsymbol{S}_{k_1}^{-1}, \dots, +\omega_{N_i}\boldsymbol{S}_{N_i}^{-1} \tag{5.9}$$

$$\boldsymbol{y}_i = \boldsymbol{S}_i\left(\omega_{k_0}\boldsymbol{S}_{k_0}^{-1}\boldsymbol{y}_{k_0} + \omega_{k_1}\boldsymbol{S}_{k_1}^{-1}\boldsymbol{y}_{k_1}, \dots, +\omega_{N_i}\boldsymbol{S}_{N_i}^{-1}\boldsymbol{y}_{k_{N_i}}\right) \tag{5.10}$$

where $\sum_{k\in\{k_0,\dots,N_i\}} \omega_k = 1$

In the general case, the parameters ω_k have to be found using iterative estimation methods. For small number of dimensions (five dimensions when minimizing $det(\boldsymbol{S}_i)$

and four dimensions when minimizing $trace(\boldsymbol{S}_i)$) analytic methods exist [Reinhardt et al., 2012] to find the best parameter ω when intersecting two residuals $\boldsymbol{y}_{k_0}$ and $\boldsymbol{y}_{k_1}$. For that reason the covariance intersection is not performed globally using all residuals at once. Instead the residuals are intersected two by two until all residuals associated to a map feature have been processed.

Hence the smaller covariance intersection of Equation 5.12 is solved sequentially for each feature. The residual $\boldsymbol{y}$ is initialized to the first residual $\boldsymbol{y}_{k_0}$ along with its covariance matrix. The next residuals $\boldsymbol{y}_{k_1} \dots \boldsymbol{y}_{k_{N_i}}$ are sequentially used and update the aggregate residual.

$$\boldsymbol{y} \leftarrow \left(\omega \boldsymbol{S}^{-1} + (1-\omega)\boldsymbol{S}_k^{-1}\right)^{-1} \left(\omega \boldsymbol{S}^{-1}\boldsymbol{y} + (1-\omega)\boldsymbol{S}_k^{-1}\boldsymbol{y}_k\right) \tag{5.11}$$

$$\boldsymbol{S} \leftarrow \left(\omega \boldsymbol{S}^{-1} + (1-\omega)\boldsymbol{S}_k^{-1}\right)^{-1} \tag{5.12}$$

The algorithm used to compute the value of ω is presented in Appendix F.

5.4.1.3. Leveraging Multiple Trajectories

While residuals of the same trajectory are correlated, residuals from different trajectories are not. Indeed, the residuals from the same trajectory are correlated because all states are correlated through the evolution model. The states from different trajectories are not correlated in any way leading to uncorrelated residuals. Therefore, multiple trajectories can be used to further reduce the uncertainty of residuals.

A residual $\bar{\boldsymbol{y}}$ with corresponding covariance matrix $\bar{\boldsymbol{S}}$ known from previous drives can be updated using the new residual $\boldsymbol{y}$ (with associated covariance matrix $\boldsymbol{S}$) obtained as described in Subsubsection 5.4.1.2 using the following equations,

$$\bar{\boldsymbol{y}} \leftarrow \left(\bar{\boldsymbol{S}}^{-1} + \boldsymbol{S}^{-1}\right)^{-1} \left(\bar{\boldsymbol{S}}^{-1}\bar{\boldsymbol{y}} + \boldsymbol{S}^{-1}\boldsymbol{y}\right) \tag{5.13}$$

$$\bar{\boldsymbol{S}} \leftarrow \left(\bar{\boldsymbol{S}}^{-1} + \boldsymbol{S}^{-1}\right)^{-1} \tag{5.14}$$

The updated residual can finally be used to evaluate the feature.

5.4.1.4. Eliminating Inaccurate Features

As mentioned previously the residuals should be centered. Therefore, as new observations are obtained and new drives are performed, the residual should tend to 0 as the uncertainty of the residual decreases. However, erroneous features will result in residuals not converging toward 0 but covariance matrices still getting smaller. Hence, at some point, the residual will be too high with respect to accuracy that would be expected from the covariance matrix.

To detect these cases, a χ^2 test is used. After the residuals are computed, the inequality from Equation 5.15 is performed. Features failing this test are flagged as erroneous to be ignored in future measurements.

$$\bar{\boldsymbol{y}}^\top \bar{\boldsymbol{S}}^{-1} \bar{\boldsymbol{y}} < F_{\chi^2}^{-1}(1-\alpha, 2) \tag{5.15}$$

where F_{χ^2} is the cumulative distribution function of a χ^2 distribution.

5.4.2. Experimental Results

5.4.2.1. Experimental Setup

To evaluate the error detection, real drives are used. The drives were recorded using the experimental vehicle of the Heudiasyc laboratory. Lidar and camera information are used as in Chapter 4 for localization. Six drives are used which follow three trajectories. All trajectories start and end at the entrance to the laboratory site. Every trajectory also covers up to the main university build (eastern most point on the map). The trajectories are detailed more in Appendix A.

The map used in this experiment accurately references road sign positions. Ignoring the intersection that was completely redone, few errors have been found in the referencing on road signs. Therefore to evaluate the error detection system artificial errors have been added to some road signs. The map used references 2382 road signs (including lamp posts, traffic lights, bollard, etc.). From all drives performed using this map, only 432 different road signs have been observed. The first drive covers roads traveled by all trajectories. Errors have been added to 20 of the more than 200 road signs observed on this trajectory to insure that the erroneous road signs can be observed by almost all trajectories. The road signs for which an error is added have been randomly chosen. An error chosen using a uniform distribution between -1 and 1 is added on the road sign reference. Figure 5.6 shows the road signs observed during the first trajectory and those for which an error has been added (red).

The error detection is performed after each drive. In the current evaluation, the road signs detected as erroneous are not ignored in future localization. Hence, flagged road signs will be matched to measurements and re-evaluated in future drives. Therefore a road sign seeming unreliable can after new drives be considered reliable again.

5.4.2.2. Error Detection

From Figure 5.7 and Table 5.1, it can be seen that after a single drive few erroneous features have been found. However, as more drives are done the number of detected erroneous features increases.

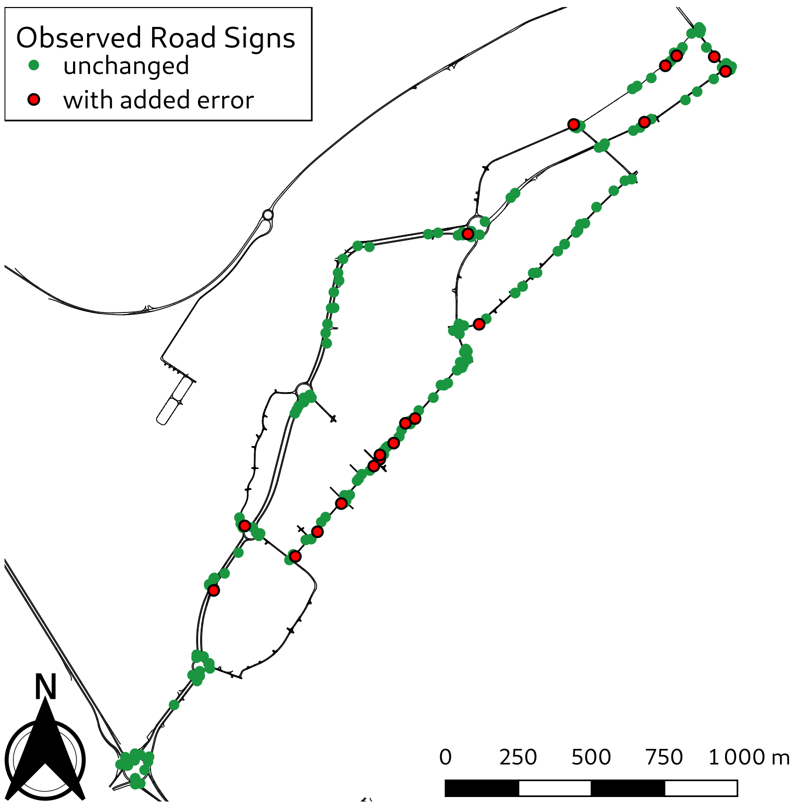
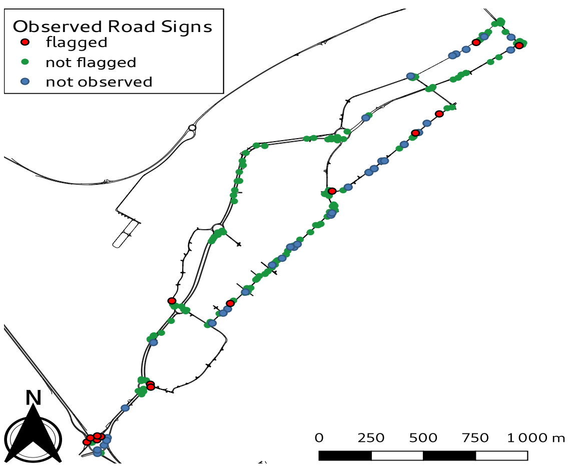


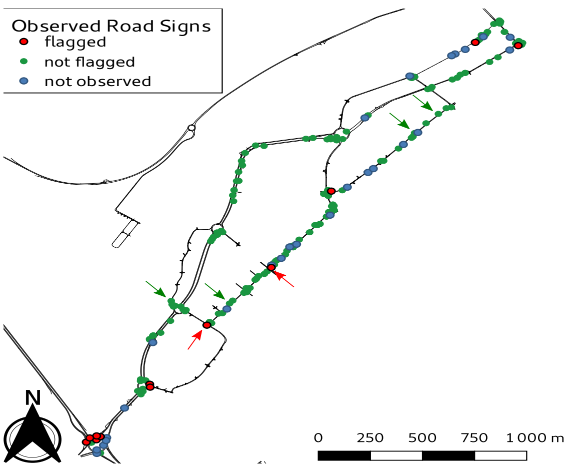
Figure 5.6.: Observed Road Signs during one drive in Compiègne (France). Road signs for which a simulated error has been added are shown in red.

Table 5.1.: Evolution of the categorization of road signs. The flagged road signs have been observed detected as faulty. The first three lines correspond to road signs with no added error (green dots in Figure 5.6) the last three lines are the 20 road signs with added errors.

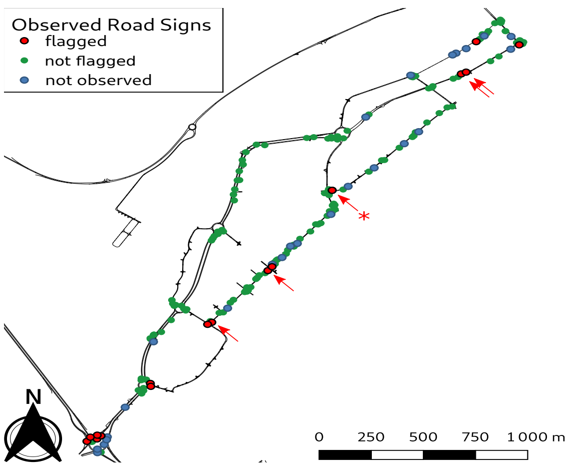
		first	second	third	fourth	fifth	sixth
Not observed	(ok)	28	16	15	14	14	11
Not flagged	(ok)	147	162	161	161	161	159
Flagged	(ok)	12	9	13	12	12	17
Not observed	(faulty)	10	10	10	9	9	9
Not flagged	(faulty)	8	7	4	4	4	3
Flagged	(faulty)	2	3	6	7	7	8



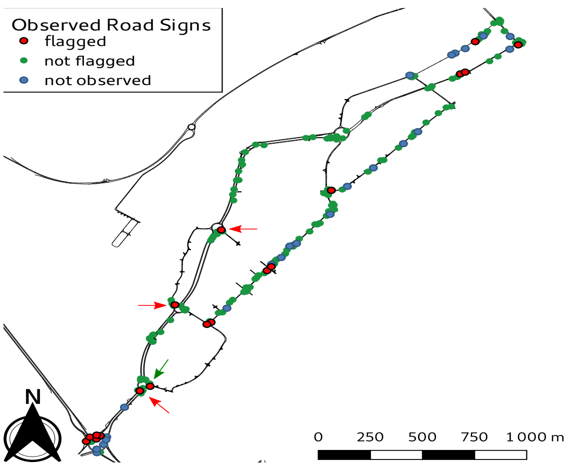
(a)



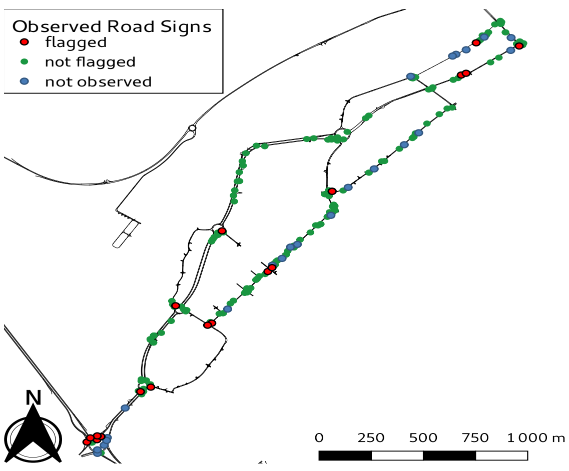
(b)



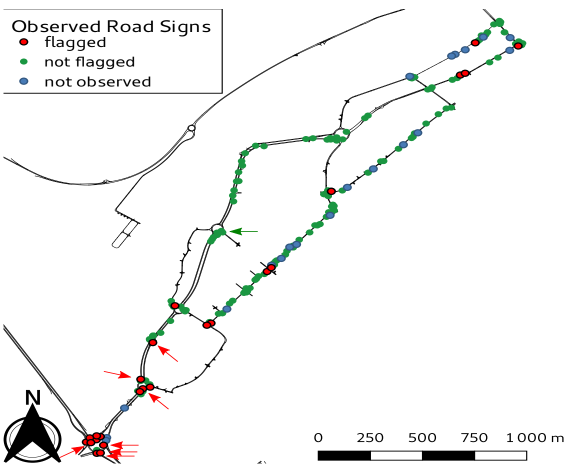
(c)



(d)



(e)



(f)

Figure 5.7.: Evolution of erroneous road signs detection trajectory after trajectory. (a) was obtained after a first drive, (b) after a second drive, etc. The arrows show changes from the previous drive (the star indicates two road signs overlapping).

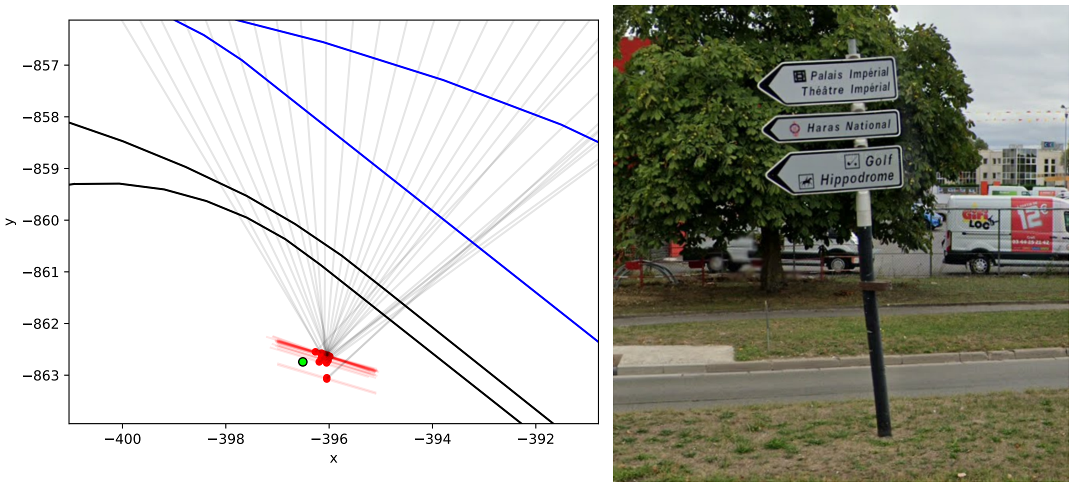


Figure 5.8.: Real erroneous road sign. (a) shows the mapped road sign (green) compared to the measurements (red), the line of sight directions are shown as gray lines (the lane markings are shown in black and the lane centers in blue). (b) photograph of the road sign

The method produces some false positives. One of the false positives, the south-western most road sign, is actually a real erroneous sign Figure 5.8. The road sign pole is tilted and the signs are not centered on the pole resulting in a combined detection error of about 0.5 m. The other false positive are found mostly near erroneous signs. This is a limitation of the method as it relies on the state estimates to evaluate road signs. Erroneous road signs will affect the state estimates negatively leading to other, accurate road signs, to manifest higher residuals than they otherwise would. This problem arises because with the current availability of measurement to the localization system, single measurements have a significant importance. As different measurements and more redundancy are added to the localization system, the effect of single measurements will be lessened and the error detection will be more robust.

The error added to the road signs being chosen uniformly between -1 and 1 meter in both dimensions, some road signs have small errors. On Figure 5.9 it is shown that road signs with errors higher than 0.5 m are either not observed (not matched) or flagged. The three faulty road signs that remain not flagged are those with smaller errors. Errors too small would require more drives to be detected. Also, the number of road signs not observed when the errors are added, even though they were observed before, shows that the matching is able to discard some erroneous road signs during localization.

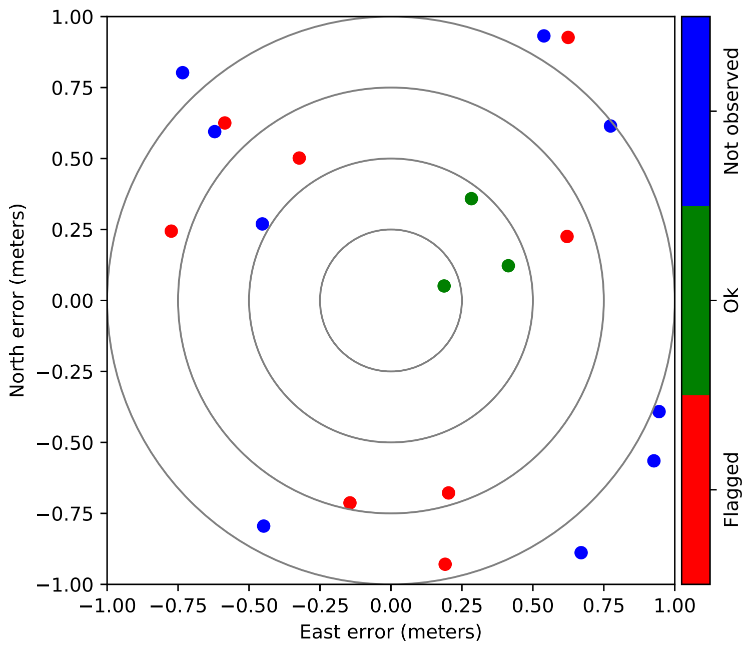


Figure 5.9.: Distribution of the error added to road signs. The color shows the classes obtained after the sixth drive.

5.4.3. Discussion on Limitation and Perspectives

5.4.3.1. Reliance on Localization Accuracy

Because the error detection system relies on the estimates from the localization to evaluate features, it will not be able to detect some map errors. Indeed, if a feature is providing the only accurate observation on a stretch of road, the localization solution will be entirely driven by this feature. The residuals of the feature would be small because no other observation is available in the vicinity. In those cases, the error of the feature will be undetectable using the proposed approach.

These two problems will become less and less likely to occur as more observations are provided to the estimation process and redundancy of the observation sources increases.

5.4.3.2. Map Correction

Removing features from the estimation process is not guaranteed to improve future localization accuracy. Indeed, even though the feature references might have some error it might still be smaller than the state estimate error. In those cases, the estimation might benefit from using the measurement.

One way to keep using the feature even though it has been flagged would be to correct the feature reference using the estimated error of this reference. However this approach needs to be carefully thought out as it could worsen the map. Map errors are not the only systematic errors that can affect localization. If another source is affecting the state estimation, the map features might be seen as unreliable. Correcting them in those cases would only reinforce the confidence of the localization in the other source.

Another way to deal with the problem would be to dynamically decide whether to use map features or not depending on the current state estimation accuracy. When the state estimation accuracy is good, the localization can afford to discard some observations that might be unreliable. When state estimation becomes less accurate, it might be preferable to use features with some referencing error over not using any observations. Such strategies have been used successfully with map features and odometry systems where map features are used to update the estimates only when the they bring enough information to improve the state [Delobel, 2018].

5.4.3.3. Risk of Over Cautiousness

In this work any error, no matter how small, is considered as an error that would justify discarding a road sign from the localization. Hence, it is assumed that the map claims perfect accuracy and that anything but perfect accuracy should be considered as erroneous. In practice, no map provider makes such claims and any map is known to have unavoidable errors. While the uncertainty attributable to the map can largely be ignored in the localization process as sensor uncertainty is significantly greater, it cannot be ignored for a process that accumulates the information of multiple trajectories. Indeed, the fusion performed by Equation 5.14 is only valid if the errors between two trajectories are uncorrelated. This assumption breaks down not only when the feature has an irregular referencing error but also if the feature has an acceptable mapping error.

This problem clearly appears in the result Table 5.1. In every drive, some actually correct road signs are classified as incorrect. If nothing is done, at some point all road signs will be classified as inaccurate.

In the current method, the uncertainty matrix of the residuals will, after many drives, tends toward perfect accuracy while the actual residual will never be able to reach such accuracy because the map never claimed that it could. Solving this problem could be done by setting a lower bound to the matrix $\boldsymbol{S}$ or including in Equation 5.14 the level of error correlation that can be expected given accuracy claimed by the map.

5.5. Conclusion

In this chapter, methods to detect map errors have been introduced. Studying post smoothing residuals of map features has been found to enable the detection of error in the references of map features.

Using lane marking residuals, a real map error has been detected. Using the weighting strategy proposed in this work, the localization accuracy was improved in the affected area.

Residuals have also been used to detect simulated errors on road signs. By merging information from the same drive using Covariance Intersection and from multiple drives, erroneous road signs can be detected. The first results obtained using this method shows that after six drives, the most prominent errors can be detected leaving road signs with small errors. The effects eliminating erroneous features will have on localization will be studied in future work.

From this work it is clear that detecting erroneous features without ground truth positioning is challenging. Moreover evaluating methods trying to achieve this requires multiple drives through the same areas for smaller errors to be detectable. Therefore, the recording of more drives and more diverse drives will be essential to any future work on that subject.

6. Conclusion

Contents

6.1. Synthesis

This thesis addresses the problem of localization using spatial memory (using maps), and temporal memory (through buffers). The goal is to improve localization using the information stored in the buffers and the map. Several key aspects of the localization process have been studied to achieve it. These aspects spread over three times scales: a high frequency estimation using Dead Reckoning (DR) (Chapter 3), a low frequency matching system (Chapter 4) and a post processing layer (Chapter 5).

A framework for performing the localization task has been proposed. The localization is built around a *Filtering Layer* which insures that the estimation is provided at high frequency and takes care of sensor delays. The intent behind having an estimation layer that only takes observations as inputs and outputs an estimate without performing more complex processing has mainly been to guaranty the time constraint the estimation has to abide. The more resource intensive tasks such as matching and calibration are performed in parallel so as not to burden the state estimation. Their effects on the estimation will only be reflected by the updates they make on the observations. This architecture therefore enables any number of parallel processes to run to perform tasks that improve localization.

One of the most critical aspects of localization is the quality of the dead-reckoning system. Early in this thesis, it was identified as a critical component as it is the system that enables localization to be performed at high frequency. It also has a great influence on the quality of the post smoothing trajectories which are used extensively in this work. In this thesis, the calibration is achieved using the estimated trajectory. This choice is made as a commercial system cannot be expected to rely on anything but the vehicle on-board sensors. Since the smoothed trajectory uses all known observations to estimate all states of the system, the choice was made to use it for calibration. Using estimated states for calibration can lead to observably problems on some parameters (e.g. wheel radii) if the localization system does not have sufficiently accurate observations to correctly estimate the vehicle state. However, with this strategy, the more observations are used, the more accurate the estimations will be, leading to a more accurate estimation of the calibration parameters.

To approach accuracy levels required for autonomous driving, observations of map features are used in the estimation. The problem of matching measurements to map features is crucial as wrong matches may have disastrous effects on localization accuracy and consistency. Using lane markings only and then road signs, a matching technique relying on an adjustment step was proposed. Global optimization methods can deal with ambiguous matches using multi-modal distributions. Correct matches become clearer and unambiguous as estimation progresses. The goal of the proposed adjustment step is to leverage a similar capacity in a filtering framework. Performing the adjustment on a trajectory rather than individual states is an approximation that enables the computation to be performed in a reasonable time frame. This approach also enables to set stricter matching thresholds enabling to limit wrong matches further. Since this task is performed in parallel to the estimation, matches are not obtained in real-time but with a delay. Because of this delay, the last few states are never estimated with measurements needing to be matched. Hence, this strategy sacrifices some real-time accuracy to improve the system safety.

Map-aided localization is only as good as the maps used. Repeatedly, errors were found in the maps used in this work. Even with a strict matching strategy erroneous features are bound to be matched with some measurements. The detection of map errors was therefore studied. Again, smoothed state estimates were used to perform this task. Errors in the reference of the map features manifest on the residuals of observation of the feature. The proposed method of weighting the accuracy of a feature observation based on a reliability factor computed using the residuals has been shown to improve the accuracy of localization when some lane markings are erroneous. Although errors on lane markings result in lateral error, it is not as critical as an error introduced on the yaw estimate. Road sign measurements can introduce such an error. Again, using the residuals has been found to enable the detection of erroneous road signs. The proposed method also uses multiple trajectory which has been found to be essential to the detection of such errors. As for the calibration, the error detection is only as good as the smoothed state estimates are accurate and map errors are bound to degrade the quality of this estimation. Hence, having a lot of observations and redundant information will be critical to further improve the system.

Throughout this thesis, the added value of that buffers and maps for localization has been clear. Kalman Smoothing has been used as the backbone of the calibration, matching and error detection. It enables to obtain the entire trajectory estimated with the information from all observations available. Moreover, it is performed following traditional filtering, which is widely used for real-time high-frequency estimation tasks. Although all methods presented in this thesis have been tested on real experimental data, the importance of more extensive testing has been made clear in particular for map error detection.

6.2. Perspectives

This work has highlighted the interest of using a spatio-temporal approach for map-aided localization, but there are still many points to explore.

Calibration experiments with additional sensors The dead-reckoning model and calibration method presented in this thesis was developed at the beginning of this work to form the basis of the localization system. Since then additional observations have been added to the localization system. The new observations result in better state estimates which should lead to better estimation of the calibration parameters. Also the size of the buffer needed to estimate the parameters should decrease as a small accurate buffer should yield better results than a longer less accurate one. Hence, it would be interesting to study how the quality of the calibration changes with more accurate state estimates. The low cost GNSS receiver used in this work does not enable accurate estimation of the vehicle position leading to estimates of the vehicle speed almost entirely dictated by the DR sensors. The wheel circumferences are therefore hard to estimate. More accurate sensors might enable to better estimate the vehicle speed leading to a better estimation of wheel circumferences.

Increase the quantity and the variety of the observations Throughout this thesis, several exteroceptive sensor measurements have been added incrementally to improve the localization accuracy. Adding lane marking measurements have proven effective to achieve small lateral errors, while road signs improve both the position estimate and the yaw estimate. From the experiments performed in Compiègne and Rambouillet, it is clear that a large number of measurements is essential to accurate localization. This effect also explains the better accuracy that can sometimes be reached using dense point cloud maps for localization since the sheer number of observed points inevitably lead to better accuracy and more robustness to small map errors. The benefit of having more observations have been noticed in this work between the two cities. In Compiègne, fewer road signs are observed and more parked cars are detected (license plates are retroreflective). The localization accuracy in Compiègne is therefore much worse than in Rambouillet. Thus, adding new measurements to the localization process is essential. Street lights are numerous in Compiègne and would be interesting to use. Beyond increasing the number of observations, it is also important to increase the redundancy of observations. In this work map errors are detected based on observed residuals. It is assumed that the residuals are caused by map errors rather than systematic errors in the observation of a particular feature. For instance, some road signs might have their pole correctly referenced at ground level, but a tilted pole results in the actual sign to be detected slightly next to the referenced position of the pole. Detecting the same feature through different means should partly limit this effect.

Exploit a better GNSS technology with accurate pseudoranges The GNSS receiver used in this work used multiple constellations (GPS and Glonass) but did not use Galileo. This new GNSS, still in its deployment phase to this day, shows very interesting accuracy characteristics. In particular, with civil codes on two different carriers, ionospheric errors can be eliminated. Moreover, with PPP corrections, an accuracy of around 20 cm is expected to be achieved in the coming years. If we anticipate this type of GNSS systems in the near future, we will either be able to use very accurate calculated positions (loose coupling approaches) or it will be possible to use very accurate pseudorange measurements in our proposed framework (tight coupling approaches). In the latter case, it will be necessary to integrate an unknown linked to the receiver clock. In both cases, the proposed approach can be adapted very well.

Validate the localization system integrity The consistency and integrity of the localization systems are essential for autonomous vehicles. While the current system is consistent for a risk of 5% (except in cases of faulty matching, see Subsection 4.5.3), autonomous vehicle applications require much smaller risk levels. The consistency of the system has not yet been studied for smaller risks. In fact, since the GNSS pose estimate is biased, we suspect that the consistency of the estimation is not satisfied for smaller risks. Once better GNSS technology is added to the localization system, its consistency should be studied for the low risk levels needed for autonomous driving applications.

Compare the matching strategy with Combined Constraint Data Association (CCDA) The matching strategy has been compared to traditional nearest neighbor methods and Munkres matching. Another interesting matching strategy is Combined Constraint Data Association (CCDA) which looks at the global compatibility of associations. In this work, although observation buffers were used to increase the number of observations used for matching, the number of individual features observed was rather small (about one road sign every five seconds) making the CCDA method impractical. With more observations, this matching method could be used and should be compared to the presented approach (although a distance between the two-dimensional road signs and the one-dimensional lane markings would have to be defined).

Adapt association probabilities using feature classification In Chapter 4, the likelihood functions used to adjust the trajectory, considered all possible matches. The *a priori* probabilities of an observation being matched to a feature were chosen the same for all features. This was done since the sensors used in the experiments did not provide a reliable classification of the observations. Using a more reliable classification, the *a priori* association probabilities could be adapted based on the observed classes. If a classifier was found to classify stop signs correctly in 90% of cases and misclassify them as yield signs in 10% of cases, this information could be used as *a priori* association probabilities in the matching process. Thus leading to better matching and better localization.

Effect of Road Sign Error Detection on Localization Accuracy In the latest results presented in this thesis, it was shown that using post smoothing residuals, map errors can be detected. The main purpose of detecting such errors is to improve localization for future drives. The effect of removing faulty features (and some correct ones) on the accuracy and consistency of the localization has yet to be studied.

Error detection does not affect localization in a straight forward way. On the plus side, removing faulty features should improve localization. However, correct features (false positives) are also bound to be removed. Because of this, the number of observations used overall is expected to diminish which might lead to worse accuracy in some areas. As discussed in Subsection 5.4.3, the scale of the errors might also have to be considered in the estimation process. It might be better for the accuracy of the localization to use a feature with a small error that no feature at all. Some errors might be acceptable if the state estimate is not accurate enough.

Application of Road Sign Error Detection to multiple vehicles In Chapter 5, a road sign error detection strategy was presented. Here, the method was integrated as part of the localization system. Nothing prevents this method to be applied remotely by sending computed residuals to a cloud service managing the map. Also, the method was applied on data recorded using a single vehicle performing multiple drives. It could also be applied using different vehicles, all contributing to improving a single map.

Adding Missing features to Improve Localization The errors studied in Chapter 5 concerned incorrectly referenced features. These errors are important as they can be small enough for the matching system to match the feature but still cause localization errors. Other types of errors can also be interesting to look at for localization. Adding features that were missing will increase the number of observations thus leading to better accuracy. This needs to be done carefully as once a feature is added it will reinforce the confidence of the localization in a trajectory. Hence, features added based on a biased state estimate (e.g. caused by incorrect matching) will lead future trajectories to also be biased. This point only reinforces the need for more observations and more redundancy.

Matching is not equally difficult for all observations and features In this work, features were treated equally in the matching process. However, some features should be considered easier to match. Markings separating two lanes are far from any other feature and could be treated differently. Also some road signs (e.g. right arrows on the inside of large roundabouts) are fairly isolated and could be matched even with large innovation. Such features could also be matched using faster techniques to enable their use in the estimation to be close to real-time. The features easy to match could be learned for each sensor using several trajectories and looking at isolated observations matched to isolated features.

A. Experimental Setup

All methods presented in this thesis have been tested using experimental data. This data was acquired using two experimental vehicles each with their own specificity. Two different experimental areas were used: Compiègne and Rambouillet. Three different dataset series were recorded. This Appendix compiles the information on the experimental vehicles, the test sites, and the datasets used for the work.

A.1. Vehicles

The Heudiasyc laboratory has several experimental vehicles that can be used to test autonomous driving features. In recent years, experimentations have almost exclusively been done with three of them. The three vehicles are all Renault ZOE.

Two vehicles called ZOE gray (see Figure A.1 (a)) and ZOE white, were purchased by the laboratory in 2013 and 2015 using funds from the Equipex ROBOTEX (ANR-10-EQPX-44-01). Those vehicles are first generation Renault ZOEs which have been modified to enable autonomous control.

In 2019 a third vehicle (see Figure A.1 (b)), a third-generation Renault ZOE, was also purchased. This vehicle is not modified for autonomous use. It is therefore only dedicated to sensor acquisitions.

The three vehicles are set up to have the closest sensor setup possible. However, there are some notable differences between the first vehicles and the most recent one.

A.1.1. Reference frames

To express the sensor measurements and the vehicle poses, several reference frames are used in this work. Figure A.2 shows each frame and the next paragraphs detail their definitions.

The first frame $\mathcal{R}_0$ is a global reference frame in which the global vehicle pose is measured. It is an East-North-Up frame meaning that its axes point toward the east, north, and up. The frame is centered on a point O chosen arbitrarily in the world.

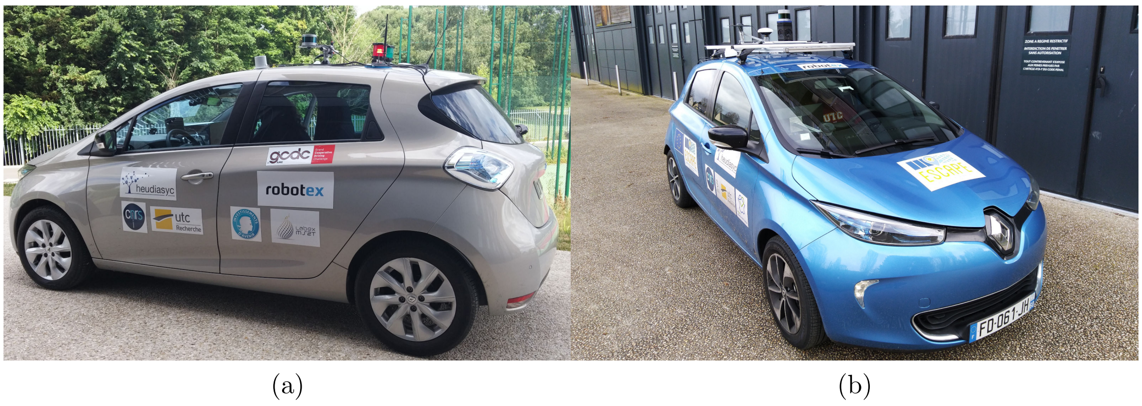


Figure A.1.: Experiment vehicles used to record the datasets. (a) is a first generation Renault ZOE, (b) is a third generation Renault ZOE.

Table A.1.: Available sensors on the two experimental vehicles.

Vehicle	1st generation	3rd generation	frequency
Gyrometer	✓	✓	100 Hz
Speed Measurement	✓	✓	100 Hz
Steering Angle	✓	✗	100 Hz
Wheel Ticks	✓	✗	50 Hz
Wheel Speeds	✗	✓	50 Hz
ublox M8T	✓	✓	2 Hz
VLP-32C	✓	✓	10 Hz
Mobileye camera	✓	✓	37 Hz

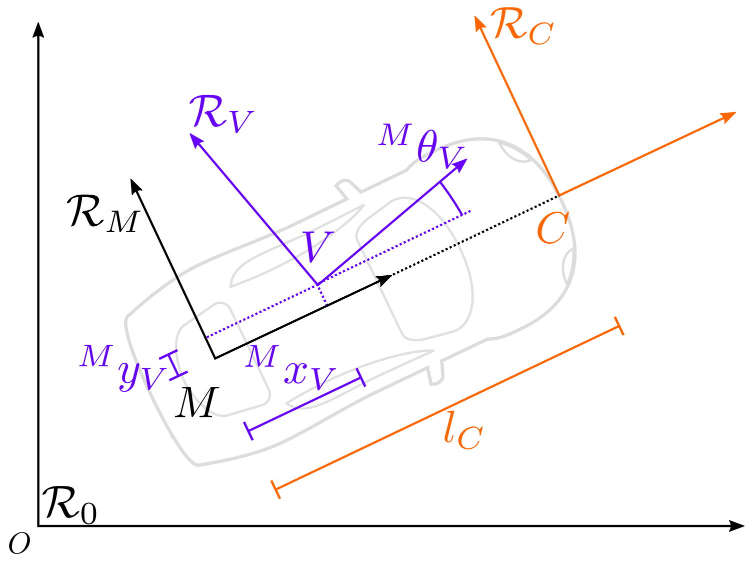


Figure A.2.: Graphic of the frames used to describe information

Both the estimated vehicle pose and the ground truth are obtained and compared in this frame.

For the localization task, the pose (position and orientation) of the vehicle are being estimated. As such, a vehicle frame $\mathcal{R}_M$ needs to be defined. The vehicle frame (also called Mobile frame) is defined to be centered on the point M in the middle of the vehicle rear wheels axle. The frame is oriented such that it has the same Up direction as the global frame $\mathcal{R}_0$, and its first axis points toward the front of the vehicle, leaving the second axis pointing toward the left. This frame can be used to evaluate localization accuracy, as the localization requirements can be defined more finely in this frame. Whereas in $\mathcal{R}_0$ the requirements for localization accuracy depend completely on the situation, in $\mathcal{R}_M$ different requirements can be set on the different axes. Indeed, localization accuracy on the first axis (also called longitudinal or along track axis) is often not critical. However, on the second axis (also called transverse or cross-track axis), the localization accuracy is critical as an error in this direction could result in the vehicle leaving the lane boundaries.

The vehicle is also equipped with two perception sensors, a Velodyne lidar and a Mobileye camera that each provides measurements in their own frames (respectively $\mathcal{R}_V$ and $\mathcal{R}_C$). The velodyne frame is centered on the center of measurement V of the sensor and its orientation is dictated by the sensor internal orientation. The accurate referencing of this frame with respect to the mobile frame has to be done by an extrinsic calibration method (described in Subsection A.1.7). The Mobileye camera, being a smart camera, provides ready-to-use measurements that are not referenced in the camera frame but rather referenced in a frame similar to the mobile frame. This frame is oriented exactly as the mobile frame but is instead centered on the point C at the vehicle front bumper. To provide measurements in this frame the camera was calibrated by Mobileye engineers after it was installed on the vehicle.

A.1.2. Ground Truth System

All vehicles are equipped with a Novatel SPAN-CPT GNSS receiver. This system uses a GNSS receiver with Real-Time Kinematic (RTK) capability loosely coupled with a high-accuracy Inertial Measurement Unit (IMU). This system provides the ground truth state used to evaluate the localization.

RTK positioning relies on a static receiver at a known position to estimate the errors affecting satellite signals. These errors can then be compensated by the vehicle receiver to obtain an accurate positioning. Several methods have been used to produce RTK solutions. The first dataset recorded in 2018 used a static receiver on top of one of the university buildings. This method enables RTK positioning in Compiègne. The RTK corrections being only valid close to the static receiver, this solution was not applicable for datasets recorded in Rambouillet (over 100 km from

Compiègne). For this dataset, a Network RTK (NRTK) system was used. Traditional RTK use a single static receiver to provide local corrections. NRTK uses a network of static receivers covering the entire country. Using this network, corrections anywhere in the covered area can be computed. In the experiments, a SatInfo NRTK module (provided by Renault) was used. To further improve the accuracy of the ground truth, the latest dataset was also post-processed with "Inertial Explorer" from the Novatel company (this is called Post-Processed Kinematics - PPK). Real time estimation is limited by the accuracy of the ephemerides and atmospheric models available. Computing the ground truth by post processing enables to use more accurate ephemerides and models. Also, the post-processed data is smoothed leading to better overall accuracy.

A.1.3. CAN Bus

All vehicles provide access to their Controller Area Network (CAN) bus. The vehicle internal sensors are therefore accessible. Through this bus, the gyrometer, steering wheel angle, wheel speeds/ticks, and vehicle speed can be obtained. The two generations of vehicles have some differences. As explained in Chapter 3 and summarized in Table A.1 the steering wheel angle and the wheel ticks are not available on the most recent vehicle.

A.1.4. u-blox Receiver

All three vehicles are equipped with the same low-cost GNSS receiver: a u-blox M8T. This receiver provides estimates of the vehicle pose.

This receiver is a single frequency receiver that can use up to two GNSS constellations. In the experiments, the receiver was set up to use GPS and GLONASS since the Galileo constellation was not yet complete. More recent tests suggest that the GALILEO measurements are much better than other constellations. Future datasets will therefore prefer Galileo over GLONASS.

A.1.5. Mobileye Camera

To obtain lane marking measurements, a Mobileye camera was used. This camera provides up to four lane marking measurements at 37 Hz. The errors of these measurements have been found to be correlated in time. For that reason, in the experiments, only a tenth of the measurements are used. The camera was provided by Renault as part of the ESCAPE project that studied localization using Precise Point Positioning (PPP) aided by lane marking measurements.

Although only a single camera is available, both the Rambouillet 2018 dataset (with the gray vehicle) and the Compiègne 2020 dataset (with the blue vehicle) have these measurements. The camera was moved from one vehicle to the other.

A.1.6. Velodyne VLP-32C

All three experimental vehicles are equipped with Velodyne VLP-32C lidar sensors. In 2018, the sensors were placed at the front of the vehicle (right at the edge between the windshield and the roof). Now, the lidars on all vehicles are placed centered on the vehicle slightly elevated above the roof.

The points provided by the sensors are corrected to compensate for the vehicle motion as described in Appendix D.

A.1.7. Extrinsic Calibration

Calibration of the sensors is essential to correctly model the link between the observations and the vehicle state. The dead-reckoning sensors only need distances that can be obtained directly from the vehicle technical specifications. The perception sensors and the GNSS antenna position still need to be identified.

The Mobileye camera returns measurements in a horizontal frame centered in the middle of the vehicle front bumper. To return measurements in this frame, the camera position in the vehicle needs to be known. This calibration was performed by a Mobileye engineer.

To obtain the lidar extrinsic calibration, a FARO Vantage laser tracker was used. It enables to measure the position of points of the vehicle with an accuracy below the millimeter. The position of the center of the rear axle can be found by measuring points on the rear wheels. The position of the screw with which the sensor is attached to the vehicle was also measured. The measurement system can then provide the transformation from the base frame to the sensor frame.

A.2. Maps

The experiments done in this work use two test cities.

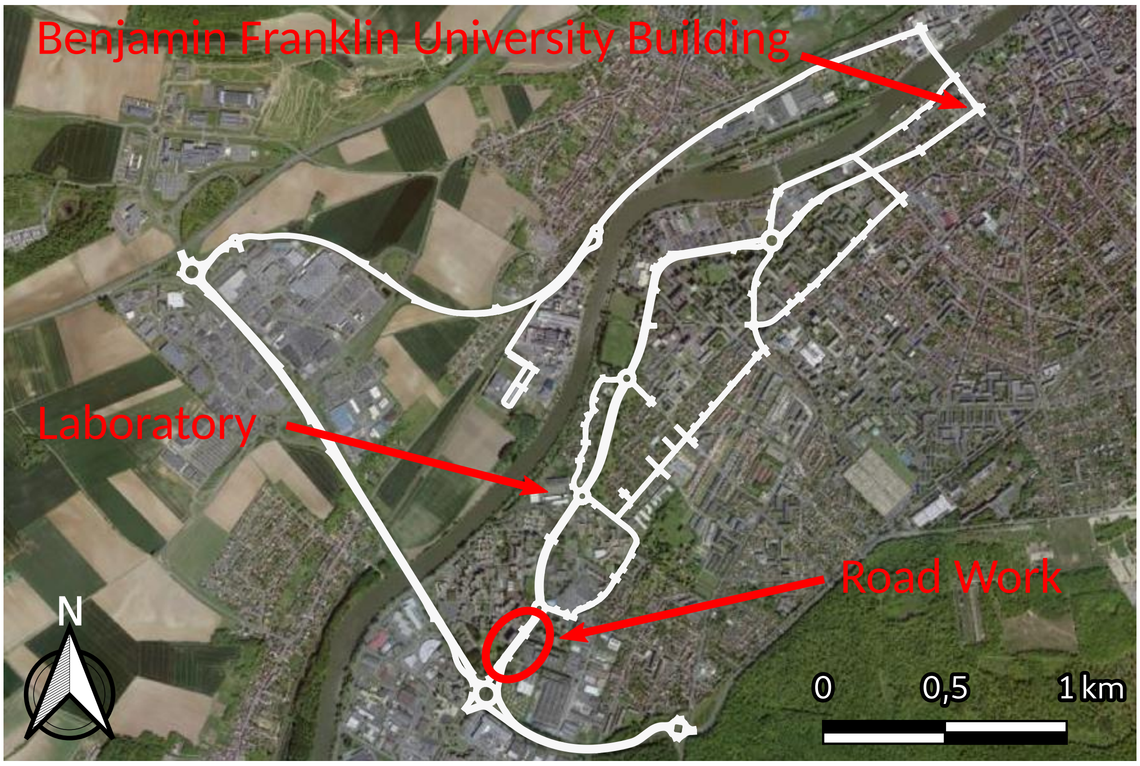


Figure A.3.: HD map of Compiègne

A.2.1. Compiègne

The laboratory being located in Compiègne most experiments are performed there. The map of Compiègne (see Figure A.3) is the largest High Definition map (HD map) available to the laboratory. The map contains 57 km of lanes (a road with two lanes is counted twice) and 164 km of lane markings (including curbs). It references 2382 road signs (including street lights, traffic lights, bollards).

A.2.2. Rambouillet

The laboratory takes part in the Tornado project which aims to demonstrate autonomous vehicles and intelligent infrastructure for mobility. The project is hosted by the city of Rambouillet and the experiments are done between the commercial area of the city and the train station of the neighboring town. Renault has provided the laboratory with an HD map of Rambouillet (see Figure A.4). This map is about half the size of Compiègne, with 28 km of lanes, 82 km of lane markings and 1127 road signs.

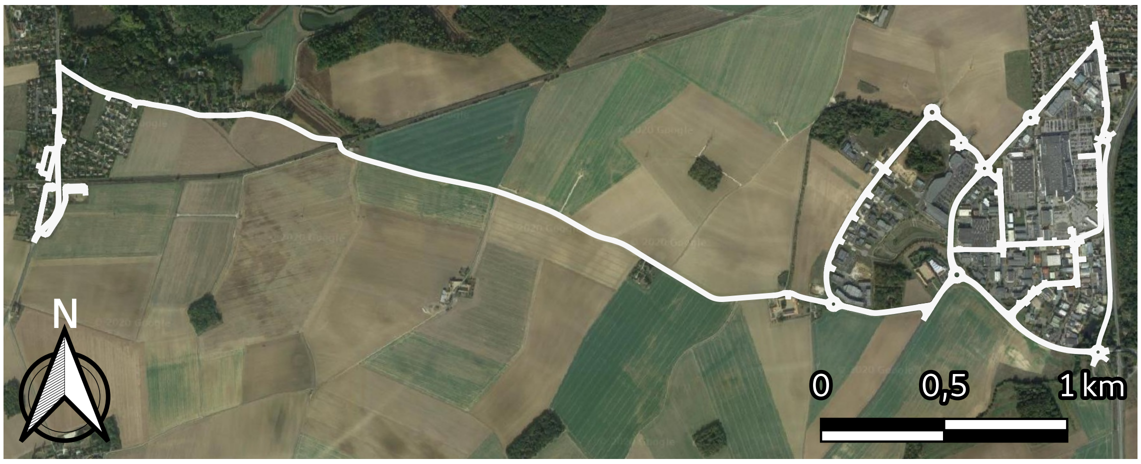


Figure A.4.: HD map of Rambouillet

Table A.2.: Length and duration of the 2018 datasets recorded in Compiègne.

Name	Length	Duration
First	4.02 km	9 min 36 s
Second	7.51 km	14 min 44 s

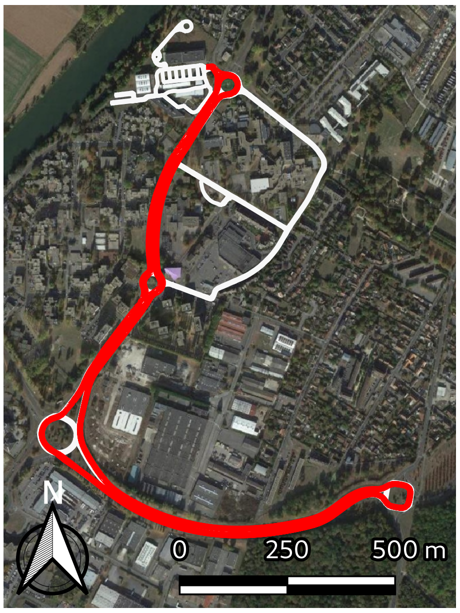
A.3. Datasets

During the course of this thesis, several datasets have been recorded. The datasets have been recorded with the help of Stéphane Bonnet, Antoine Lima, Stefano Masi, Thierry Monglon, Correntin Sanchez.

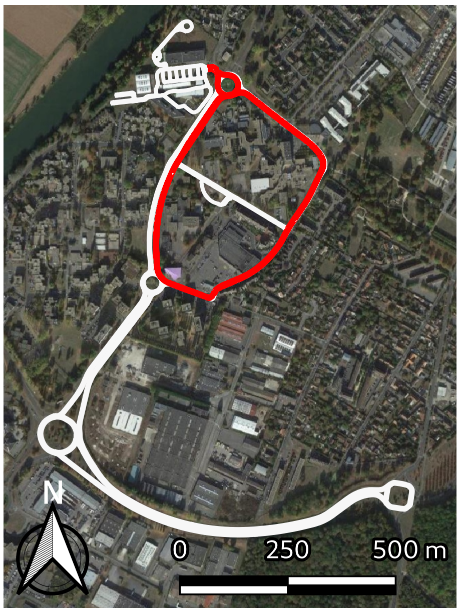
A.3.1. Compiègne April 2018

A first dataset was recorded with the gray Renault ZOE in 2018. At the time the Mobileye camera and Velodyne lidar were not available. Hence this dataset only contained ground truth positioning, a pose estimate from a u-blox M8T receiver and the vehicle internal sensors. The dataset was used to evaluate the dead reckoning model and the calibration technique presented in Chapter 3.

The dataset contains two recordings of two different trajectories. A first trajectory, shown in Figure A.5 (a), starts at the roundabout in front of the laboratory, loops around a square roundabout south-east of the map and returns back to the start. The second trajectory, shown in Figure A.5 (b), consists of three loops around a commercial center and a part of the university building. Characteristics of the trajectories are found in Table A.2.

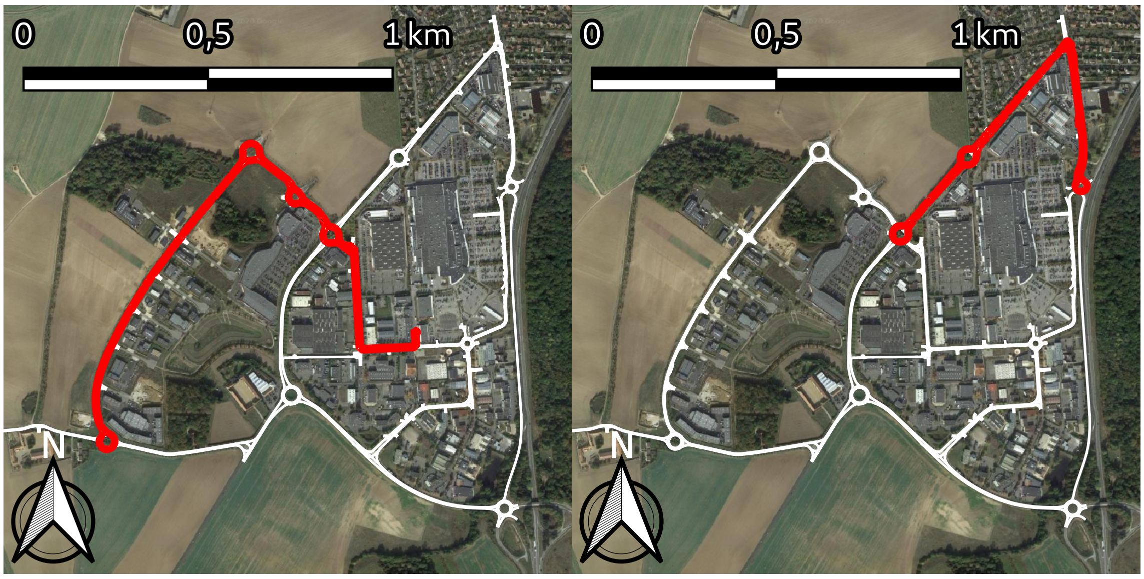


(a)



(b)

Figure A.5.: Trajectories of the dataset recorded in Compiègne in 2018. Both trajectories start at the entrance of the laboratory. The first trajectory goes up to the roundabout South-East of the map. The second trajectory loops three times around a commercial center and university buildings.



(a) (b)

Figure A.6.: Trajectories performed for the three recordings in Rambouillet 2018.

Table A.3.: Length and duration of the 2018 datasets recorded in Rambouillet.

Name	Length	Duration
Faulty 1	4.42 km	8 min 29 s
Faulty 2	5.10 km	9 min 47 s
Roundabouts	2.31 km	5 min 30 s

A.3.2. Rambouillet October 2018

Several datasets have also been recorded in Rambouillet. Unlike the previous datasets, the vehicles were now equipped with a Mobileye camera and a Velodyne VLP-32C lidar.

Three datasets recorded on two trajectories have been used in this work. The first two datasets (Faulty 1&2) followed the trajectory shown in Figure A.6 (a). This trajectory is interesting because it goes through an area of the map that has changed. Several lane markings are known to be incorrectly referenced and several road signs have also changed. The second trajectory used for the third dataset traverses four roundabouts close to each other in the north-east of the map. Table A.3 contains the length and duration of each dataset.

The first and third trajectory were used in Chapter 4 to evaluate the matching method. The first two trajectories were used in Chapter 5 to evaluate the marking reliability and the localization.

Table A.4.: Length and duration of the 2020 datasets recorded in Compiègne.

Name	Length	Duration
Between Roundabouts	9.50 km	17 min 30 s
Laboratory to BF 1	7.03 km	11 min 42 s
Laboratory to BF 2	7.03 km	12 min 54 s
Large Loop 1	12.85 km	14 min 43 s
Large Loop 2	12.85 km	14 min 37 s
Large Loop 3	12.84 km	14 min 17 s

A.3.3. Compiègne March 2020

In 2020, more extensive series of datasets were recorded in Compiègne. Following the newly received HD map of the city, several datasets covering most of the new map were recorded. Six datasets were recorded using three trajectories (see Figure A.7).

The first trajectory goes south to the largest roundabout, then north-east up to the Benjamin Franklin university building and back to the laboratory. To cover more ground, a different route was taken on the way back. This trajectory covers a difficult part of the map where an intersection was being changed during the recording. This area yields almost no lane marking or road sign measurements. The roundabout that follows is also difficult as few road signs are visible and one of them is not correctly referenced.

The next two datasets were recorded using a similar trajectory with the exception that it does not go through the difficult section of the map.

The last three datasets are the longest at almost 13 km in length each (see Table A.4 for exact figures). This trajectory forms a loop around the Oise river that traverses Compiègne. Like the first trajectory, it passes through the difficult part of the map. Additionally, it includes the southwest road on which the vehicle reaches up to 70 km/h and that contains very few road signs. The last three trajectories containing both the difficult map section and a long stretch of road with few longitudinal measurements, they were not used to evaluate the matching and localization of Chapter 4. However, all datasets were used in Chapter 5 to evaluate the fault detection method.

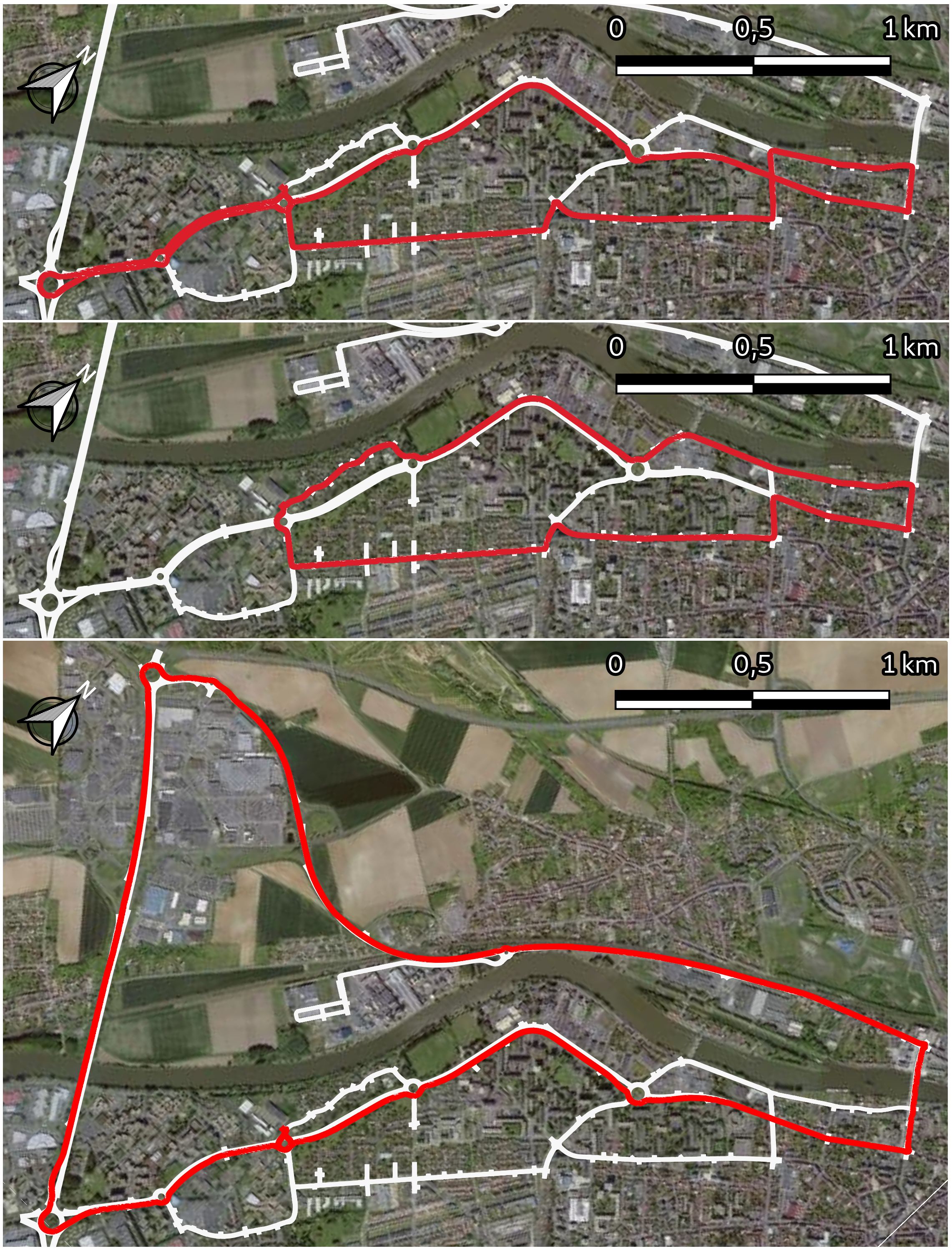


Figure A.7.: The three trajectories used for the datasets recorded in Compiègne in 2020.

B. Kalman Smoothing Derivation

This appendix is based on the book [Radix, 1984].

B.1. Prerequisites [Tong, 1990]

In the following sections, the probability density function of a multivariate Gaussian distribution with mean $\boldsymbol{a}$ and covariance matrix $\boldsymbol{A}$ is noted

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A}) \tag{B.1}$$

Joint Gaussian distribution from marginal and conditional distributions

Theorem 1. *If the marginal probability distribution of a random variable $\boldsymbol{x}$ is a Gaussian distribution, and the conditional probability distribution of a random variable $\boldsymbol{y}$ knowing $\boldsymbol{x}$ is also a Gaussian distribution with a mean being a linear function of $\boldsymbol{x}$ (i.e. $\boldsymbol{Jx}$)*

$$\begin{cases} p(\boldsymbol{x}) & = \mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A}) \\ p(\boldsymbol{y} \mid \boldsymbol{x}) & = \mathcal{N}(\boldsymbol{y}; \boldsymbol{Jx}, \boldsymbol{R}) \end{cases} \tag{B.2}$$

then, the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$ is a Gaussian distribution described by

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Ja} \end{bmatrix}, \begin{bmatrix} \boldsymbol{A} & \boldsymbol{AJ}^\top \\ \boldsymbol{JA} & \boldsymbol{JAJ}^\top + \boldsymbol{R} \end{bmatrix}\right) \tag{B.3}$$

Marginal and conditional distributions from the joint Gaussian distribution.

Theorem 2. *If the joint probability of $\boldsymbol{x}$ and $\boldsymbol{y}$ is described by a known Gaussian distribution*

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}; \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{B} \end{bmatrix}\right) \tag{B.4}$$

Then, the marginal distributions of $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are described by*

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A}) \tag{B.5}$$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{b}, \boldsymbol{B}) \tag{B.6}$$

And the conditional probabilities of $\boldsymbol{x}$ *knowing* $\boldsymbol{y}$*, and* $\boldsymbol{y}$ *knowing* $\boldsymbol{x}$ *are described by*

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{a} + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{b}), \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^{\top}\right) \tag{B.7}$$

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{b} + \boldsymbol{C}^{\top}\boldsymbol{A}^{-1}(\boldsymbol{x} - \boldsymbol{a}), \boldsymbol{B} - \boldsymbol{C}^{\top}\boldsymbol{A}^{-1}\boldsymbol{C}\right) \tag{B.8}$$

B.2. Mathematical Derivation

The objective of smoothing is to find the distribution of $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N})$ for $k \in [\![0; N]\!]$ [Radix, 1984]. It is important to note that $p(\boldsymbol{x}_N \mid \boldsymbol{z}_{1:N})$ is already known at epoch N using filtering only. Therefore, if we are able to find $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N})$ based on $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:N})$, we will be able to find the smoothed estimate at every epoch by applying a backward update.

The steps of the derivation are listed next. The key is to go from knowing a probability knowing $\boldsymbol{z}_{1:k}$ to knowing a probability knowing $\boldsymbol{z}_{1:N}$. This is achieved by finding $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k})$ and using the Markov assumption to obtain $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:N})$.

The steps that will be used to derive the smoothing equation are listed below, they are further detailed in the following paragraphs.

1. $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{z}_{1:k})$ and $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k})$ are known after the filtering
2. $p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:k})$ using Theorem 1
3. $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k})$ using Equation B.7 from Theorem 2
4. $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:N})$ using the Markov assumption
5. $p(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k \mid \boldsymbol{z}_{1:N})$ using Theorem 1
6. $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N})$ using Equation B.6 from Theorem 2

Output from the filtering

After the filtering is performed, the probabilities $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k})$ are known for $k \in [\![0; N]\!]$. Moreover, the evolution model provides a relation between $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_k$, we can therefore deduce $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{z}_{1:k}) = p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$ because of Markov assumption.

Hence, at this stage we know:

$$\begin{cases} p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k}\right) & = \mathcal{N}\left(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}\right) \\ p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{z}_{1:k}\right) & = \mathcal{N}\left(\boldsymbol{x}_{k+1}; \boldsymbol{A}_k\boldsymbol{x}_k, \boldsymbol{Q}_k\right) \end{cases} \tag{B.9}$$

Joint distribution knowing $\boldsymbol{z}_{1:k}$

Ideally, we would want to obtain $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k}\right)$ since using the Markov assumption, we would then be able to get $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:N}\right)$. In other words, finding $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k}\right)$ would enable us to get from an estimate knowing $\boldsymbol{z}_{1:k}$ to an estimate knowing $\boldsymbol{z}_{1:N}$.

One way to find the conditional probability $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k}\right)$ is to use the joint probability $p\left(\boldsymbol{x}_k, \boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:k}\right)$. We can see that using Equation B.9, the Theorem 1 directly apply. We therefore obtain:

$$p\left(\boldsymbol{x}_k, \boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:k}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_{k+1} \end{bmatrix}; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k} \\ \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{k|k} & \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top \\ \boldsymbol{A}_k\boldsymbol{P}_{k|k} & \boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top + \boldsymbol{Q}_k \end{bmatrix}\right) \tag{B.10}$$

or using intermediate variable from the Kalman filter

$$p\left(\boldsymbol{x}_k, \boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:k}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_{k+1} \end{bmatrix}; \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k} \\ \hat{\boldsymbol{x}}_{k+1|k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{k|k} & \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top \\ \boldsymbol{A}_k\boldsymbol{P}_{k|k} & \boldsymbol{P}_{k+1|k} \end{bmatrix}\right) \tag{B.11}$$

Conditional distribution knowing $\boldsymbol{z}_{1:k}$

As mentioned previously, finding the conditional distribution $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k}\right)$ is the key to obtain a probability knowing $\boldsymbol{z}_{1:N}$. Using Equation B.7 we have:

$$p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:k}\right) = \mathcal{N}\left(\boldsymbol{x}_k; \boldsymbol{m}, \boldsymbol{P}_{\boldsymbol{m}}\right) \tag{B.12}$$

with

$$\boldsymbol{m} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top + \boldsymbol{Q}_k\right)^{-1}\left(\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right) \tag{B.13}$$

$$\boldsymbol{P}_{\boldsymbol{m}} = \boldsymbol{P}_{k|k} - \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top + \boldsymbol{Q}_k\right)^{-1}\left(\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\right)^\top \tag{B.14}$$

Here the notation can be simplified by introducing the variable $\boldsymbol{J}_k = \boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top + \boldsymbol{Q}_k\right)^{-1}$. The previous expressions become:

$$\boldsymbol{m} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{J}_k\left(\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right) \tag{B.15}$$

$$\boldsymbol{P}_{\boldsymbol{m}} = \boldsymbol{P}_{k|k} - \boldsymbol{J}_k\left(\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\right)^\top \tag{B.16}$$

Furthermore, it would be useful in the following development to have $\boldsymbol{P}_m$ expressed in the form $\boldsymbol{P}_m = \boldsymbol{A} \pm \boldsymbol{J}_k\boldsymbol{B}\boldsymbol{J}_k^\top$. We use the fact that

$\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\left(\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)^{-1}\right)^\top=\boldsymbol{I}$ (because of the properties of symmetric invertible matrices) and write:

$$\begin{aligned}
\boldsymbol{P}_{\boldsymbol{m}} &= \boldsymbol{P}_{k|k}-\boldsymbol{J}_k\,\overbrace{\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\left(\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)^{-1}\right)^\top}^{\boldsymbol{I}}\left(\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\right)^\top && \text{(B.17)}\\
&= \boldsymbol{P}_{k|k}-\boldsymbol{J}_k\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\left(\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)^{-1}\right)^\top\left(\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\right)^\top && \text{(B.18)}\\
&= \boldsymbol{P}_{k|k}-\boldsymbol{J}_k\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\left(\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)^{-1}\right)^\top && \text{(B.19)}\\
&= \boldsymbol{P}_{k|k}-\boldsymbol{J}_k\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\boldsymbol{J}_k^\top && \text{(B.20)}
\end{aligned}$$

Conditional distribution knowing $\boldsymbol{z}_{1:N}$

At this stage, we are finally able to write the probability of $\boldsymbol{x}_k$ knowing $\boldsymbol{z}_{1:N}$ (and $\boldsymbol{x}_{k+1}$). Using the Markov assumption again we obtain,

$$\begin{aligned}
p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1},\boldsymbol{z}_{1:N}\right) &= p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1},\boldsymbol{z}_{1:k}\right) && \text{(B.21)}\\
&= \mathcal{N}\left(\boldsymbol{x}_k;\boldsymbol{m},\boldsymbol{P}_{\boldsymbol{m}}\right) && \text{(B.22)}
\end{aligned}$$

with

$$\begin{aligned}
\boldsymbol{m} &= \hat{\boldsymbol{x}}_{k|k}+\boldsymbol{J}_k\left(\boldsymbol{x}_{k+1}-\boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right) && \text{(B.23)}\\
\boldsymbol{P}_{\boldsymbol{m}} &= \boldsymbol{P}_{k|k}-\boldsymbol{J}_k\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top+\boldsymbol{Q}_k\right)\boldsymbol{J}_k^\top && \text{(B.24)}
\end{aligned}$$

Joint distribution knowing $\boldsymbol{z}_{1:N}$

We now know $p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1},\boldsymbol{z}_{1:N}\right)$ from the aforementioned reasoning. And we can assume that $p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:N}\right)$ is known since we are working backward starting from the only "smoothed" state that we know: $\hat{\boldsymbol{x}}_{N|N}$. Hence, we have the following information:

$$\begin{cases}
p\left(\boldsymbol{x}_k \mid \boldsymbol{x}_{k+1},\boldsymbol{z}_{1:N}\right) &= \mathcal{N}\left(\boldsymbol{x}_k;\boldsymbol{m},\boldsymbol{P}_{\boldsymbol{m}}\right)\\
p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:N}\right) &= \mathcal{N}\left(\boldsymbol{x}_{k+1};\hat{\boldsymbol{x}}_{k+1|N},\boldsymbol{P}_{k+1|N}\right)
\end{cases} \qquad \text{(B.25)}$$

Therefore, we are in a case where we can apply Theorem 1 and get the joint distribution:

$$p\left(\boldsymbol{x}_k,\boldsymbol{x}_{k+1} \mid \boldsymbol{z}_{1:N}\right) = \qquad \text{(B.26)}$$
$$\mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_{k+1}\\ \boldsymbol{x}_k\end{bmatrix};\begin{bmatrix}\hat{\boldsymbol{x}}_{k+1|N}\\ \hat{\boldsymbol{x}}_{k|k}+\boldsymbol{J}_k\left(\hat{\boldsymbol{x}}_{k+1|N}-\boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right)\end{bmatrix},\begin{bmatrix}\boldsymbol{P}_{k+1|N} & \boldsymbol{P}_{k+1|N}\boldsymbol{J}_k^\top\\ \boldsymbol{J}_k\boldsymbol{P}_{k+1|N} & \boldsymbol{J}_k\boldsymbol{P}_{k+1|N}\boldsymbol{J}_k^\top+\boldsymbol{P}_{\boldsymbol{m}}\end{bmatrix}\right)$$

Marginal distribution knowing $\boldsymbol{z}_{1:N}$

Finally, from the previous joint distribution we can extract the marginal distribution $p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N}\right)$ using Equation B.6:

$$p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N}\right) = \tag{B.27}$$
$$\mathcal{N}\left(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{J}_k\left(\hat{\boldsymbol{x}}_{k+1|N} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right), \boldsymbol{J}_k\boldsymbol{P}_{k+1|N}\boldsymbol{J}_k^\top + \boldsymbol{P}_{k|k} - \boldsymbol{J}_k\left(\boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^\top + \boldsymbol{Q}_k\right)\boldsymbol{J}_k^\top\right)$$

Or written using the predicted state $\hat{\boldsymbol{x}}_{k+1|k}$ and covariance matrix $\boldsymbol{P}_{k+1|k}$,

$$p\left(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:N}\right) = \tag{B.28}$$
$$\mathcal{N}\left(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{J}_k\left(\hat{\boldsymbol{x}}_{k+1|N} - \hat{\boldsymbol{x}}_{k+1|k}\right), \boldsymbol{P}_{k|k} + \boldsymbol{J}_k\left(\boldsymbol{P}_{k+1|N} - \boldsymbol{P}_{k+1|k}\right)\boldsymbol{J}_k^\top\right)$$

C. Dead-Reckoning sensor Modeling Analysis

The accuracy of the dead-reckoning depends on the accuracy of sensor models. The models presented in Chapter 3 have been found to fail in some instances. This Appendix details the limitations of the dead reckoning sensor models and the reasons why more accurate models have not been used.

C.1. Speed Observation

The speed measurement is available on all vehicles used for the experiments. It is used for the estimation only for the first generation Renault ZOE for which wheel speed measurements were not available. This is done such that the speed estimation is smoothed which cannot be achieved using the wheel ticks.

Although the exact way the vehicle computes that speed is unknown, it is fair to assume that it is computed using measurements of the wheel speeds. It was assumed that to a scaling factor this measurement directly described the speed of the vehicle.

In Figure C.1, the speed measurement error $(v_k^{CAN} - \check{v}_k)$ is shown with respect to the yaw rate measurement. The lateral acceleration is also shown by the color of the points. In this figure, it can be seen that the speed measurement error has a dependency to the vehicle yaw rate or lateral acceleration. One explanation of this effect is tire deformation. At high lateral accelerations, the tire slips more leading to a vehicle speed slightly slower than what would be expected given the wheel speeds.

In this work, this effect was not corrected. Figure C.1 was produced using data recorded with purposefully high rates of rotation. This was done to study this effect but is not representative of normal driving situations.

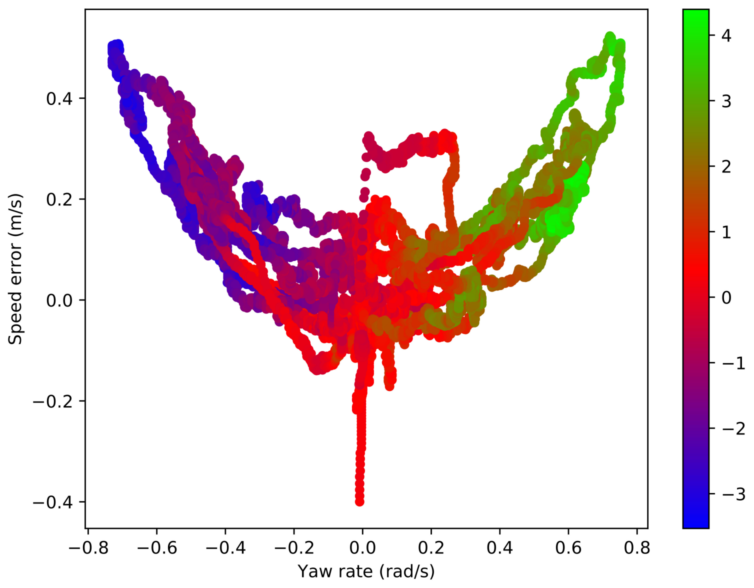


Figure C.1.: Error in the speed observation provided by the vehicle CAN bus with respect to the yaw rate. The color of the points shows the lateral acceleration of the vehicle measured at the center of the rear axle.

C.2. Steering Wheel Angle Observation

In Chapter 3, the relation between the virtual front wheel (which relates to the vehicle speed and yaw rate) and the measured steering wheel angle was studied. It was found that the relation is linear with a factor a_δ linking the two. The high angles often correspond to low vehicle speeds. Hence, the modeling was mainly driven by low speed values.

On Figure C.2, the effect of the speed on the relationship is shown. The linear model is recomputed using only data from specific speeds. For low speeds (< 5 m/s), the relation between the virtual front wheel angle and steering wheel angle is the same as what was found in Chapter 3. However, at higher speeds, the value found for a_δ changes.

As for the yaw rate influence on the speed error, this effect can be explained by tire deformations. The virtual front wheel angle cannot be measured (because it does not really exist). Hence to obtain the angle, the ground truth speed and yaw rate were used. The assumption was made that the wheel was not skidding so the wheel angle would be directed by the speed vector at the wheel position. At higher speeds, this assumption is not guaranteed.

Despite this effect, the coefficient linking steering wheel angle and the virtual front wheel angle is still chosen constant. The coefficient only seems to change noticeably

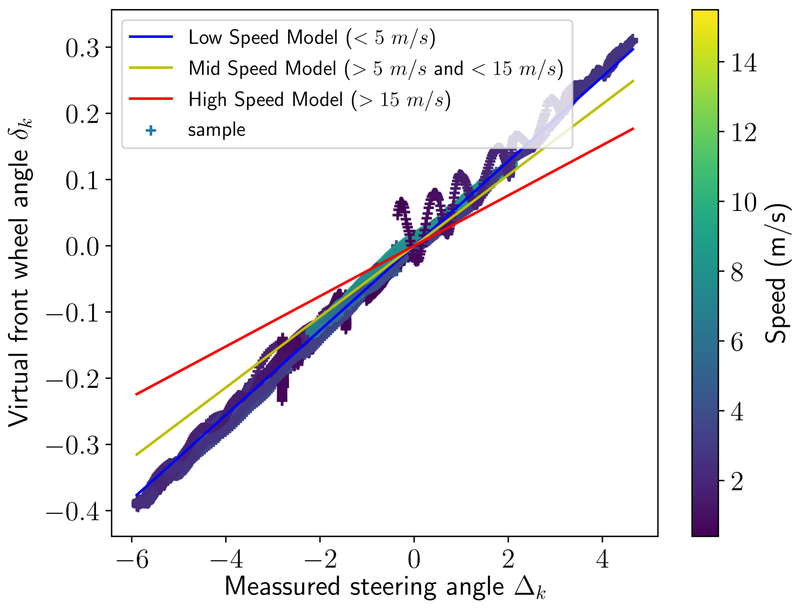


Figure C.2.: Estimation of the relation between the steering wheel angle observation and the virtual front wheel angle.

at speeds at which the vehicle does not turn significantly. Hence, this simplification does not have significant consequences on the estimation.

D. Correction of LiDAR Pointcloud using Vehicle Motion

D.1. Introduction

Rotating lidars such as Velodyne VLP32C provides information not directly as Cartesian point clouds but as a list of points in polar coordinates. Conversion from polar to Cartesian coordinates is not trivial in most cases [Merriaux et al., 2016]. Indeed, lidars typically do not register all points at once because of restriction on power emission. The 32 lasers from the VLP32C are fired by pairs (16 firing) over a 55.296 μss span. Hence, 32 points are measured every 55.296 μs, which amounts to 57856 points every 0.1 ms (scan length when the lidar is set at 600 RPM). Therefore each point in the scan is not measured at the same time. Hence when the sensor is moving, to obtain the true position of each point, the trivial conversion form polar to Cartesian coordinates is not sufficient. This is particularly important for vehicle applications where the speed often reaches over 10 m/s which would result in a 1 m error on point position in the Cartesian scan.

D.2. Cartesian Conversion

The sensor contains ranges ρ_t^i and azimuth angles α_t^i where t is the firing time and i is the laser id (corresponding to which ring the point belongs to). The trivial polar to Cartesian conversion can be done

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \rho_i^t \cdot \cos(e_i) \cdot \cos(\alpha_i^t + \delta_i) \\ \rho_i^t \cdot \cos(e_i) \cdot \cos(\alpha_i^t + \delta_i) \\ \rho_i^t \cdot \sin(e_i) \end{bmatrix} \tag{D.1}$$

where e_i is the laser elevation, δ_i is the azimuth offset that needs to be accounted for since the lasers are not aligned on the same azimuth but shifted on four different orientations.

D.3. Compensating Sensor Movement

The point cloud after conversion is provided with a time stamp corresponding to either the beginning of the firing t_0 sequence or the end t_1. If the point cloud is provided with the time stamp of the beginning of the firing sequence, the end of the point cloud will not be correctly referenced as the sensor might have moved.

Assuming a 2D movement defined by the sensor velocity v and its yaw rate ω, the pose of the sensor at time t in the sensor frame at t_0 is derived from the following equation:

$$\begin{bmatrix} x_s \\ y_s \\ \theta_s \end{bmatrix} = \begin{bmatrix} \Delta t \cdot v \cdot \cos(\Delta t \omega) \\ \Delta t \cdot v \cdot \sin(\Delta t \omega) \\ \Delta t \omega \end{bmatrix} \tag{D.2}$$

where $\Delta t = t - t_0$.

Using this pose, the transformation from the sensor frame at t to the sensor frame at t_0 can be defined. The polar coordinates can thus be converted using

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta_s & \sin\theta_s & 0 & x_s \\ -\sin\theta_s & \cos\theta_s & 0 & y_s \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \rho_i^t \cdot \cos(e_i) \cdot \sin(\alpha_i^t + \delta_i) \\ \rho_i^t \cdot \cos(e_i) \cdot \cos(\alpha_i^t + \delta_i) \\ \rho_i^t \cdot \sin(e_i) \\ 1 \end{bmatrix} \tag{D.3}$$

The resulting point cloud is provided with the timestamp t_0.

It is important to note that this correction assumes that the true position of each point is the same at time t and at time t_0 (or any reference time). With this assumption the points can be returned with any timestamp as long as the sensor movement has been considered. This assumption **does not hold** for moving objects. Indeed, for moving objects, the point true position changes with time and therefore needs to be accounted for when changing timestamp.

With this correction the lidar points do not form rings anymore as they are usually depicted. Indeed, if the laser path would actually form rings, it would require to jump from one ring of a scan to the next. This does not happen in practice. Instead, the laser forms a continuous path rotating around a possibly moving sensor. The path is closer to a cycloid than a ring (see Figure D.1).

Here the specific transformation is computed for each point. [Varga et al., 2017] have proposed approaches using matrix exponential and logarithmic functions. In those approaches a transformation for a specific delay Δ_0 is computed at the start. To obtain the transformation for a point with delay Δ_i the transformation is raised to the fractional power $\frac{\Delta_i}{\Delta_0}$.

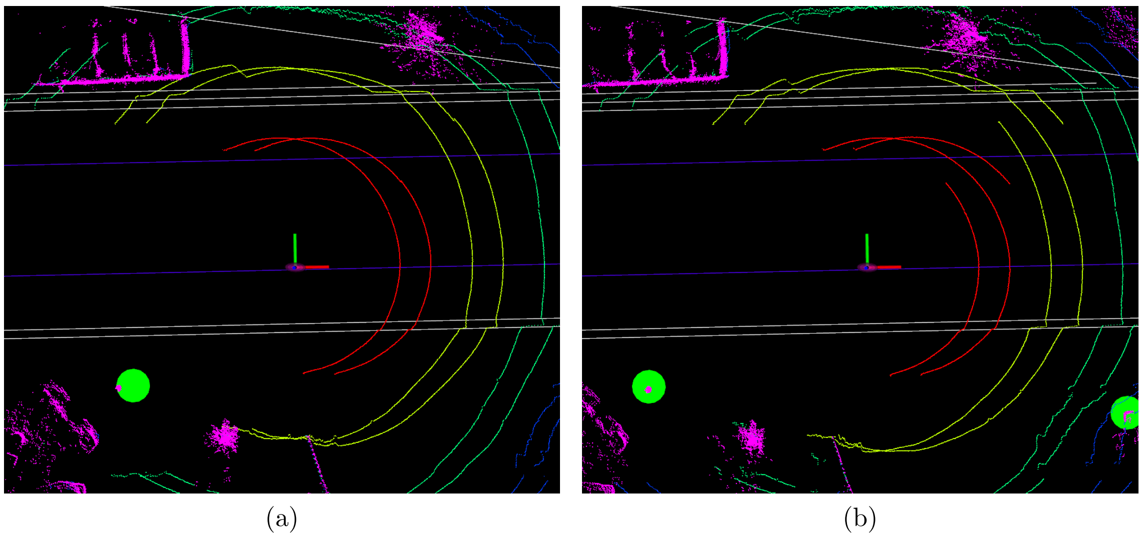


Figure D.1.: Two consecutive lidar scans without accounting for the sensor movement (a) and by correction the polar to Cartesian conversion using the sensor speed and yaw rate (b). The points of the same color have been registered by the same laser, in figure (a) the points form rings whereas in figure (b) the points form a continuous cycloid where the end of a scan directly follows the beginning of the next.

E. Crosswalk Measurements

Pedestrian crossings are an interesting type of marking. They are one of the few markings, along with stop and yield lines, that provide information in the road longitudinal direction. While the camera can detect such feature relatively easily because the camera measurements are dense, lidars have more difficulty. Indeed, lidars providing the most complete representation of the environment still do not provide a dense representation of the environment. In particular, rotating lidars are often dense in one direction (thousand points along a ring) and very sparse in another (at most 64/128 layers). Because of this, it is complicated to accurately detect features in this direction. Such lidars are often placed on top of the vehicle [Ghallabi et al., 2018, Levinson and Thrun, 2010, Takeda et al., 2019]. This placement results in point clouds with high angular resolution around the vehicle but with only a few rings (16 to 64) where points are measured. Using a single lidar scan, at most three rings will hit the crosswalks which can make them hard to identify.

The following sections describe the crosswalk observation model and the crosswalk detector developed during Antoine Lima's internship. This work has led to the publication of a conference paper [Lima et al., 2020].

E.1. Observation Model

The measurement $\begin{bmatrix} {}^{M}x_C & {}^{M}y_C \end{bmatrix}^\top$ of the position of a pedestrian crossing can only be accurate along one direction (the measurement is obtained in the sensor frame, here the observation model is presented using measurements in the vehicle mobile frame to make the equations lighter). This is due to two components.

First the map does not reference the center of the crosswalk but contains the segment of the road traversing the crosswalk. Hence, the point $\begin{bmatrix} {}^{0}x_R & {}^{0}y_R \end{bmatrix}^\top$ that can be extracted from the map will correspond to points that are centered on a lane. When multiple lane traverse the same crosswalk, multiple parallel segments will be referenced.

Secondly, the detector identifies crosswalks based on the strips it detects. Therefore the position of the detected point is sensitive to the detection (or not) of particular strips. If strips are missing the resulting point will shift in the direction of the pedestrian crossings.

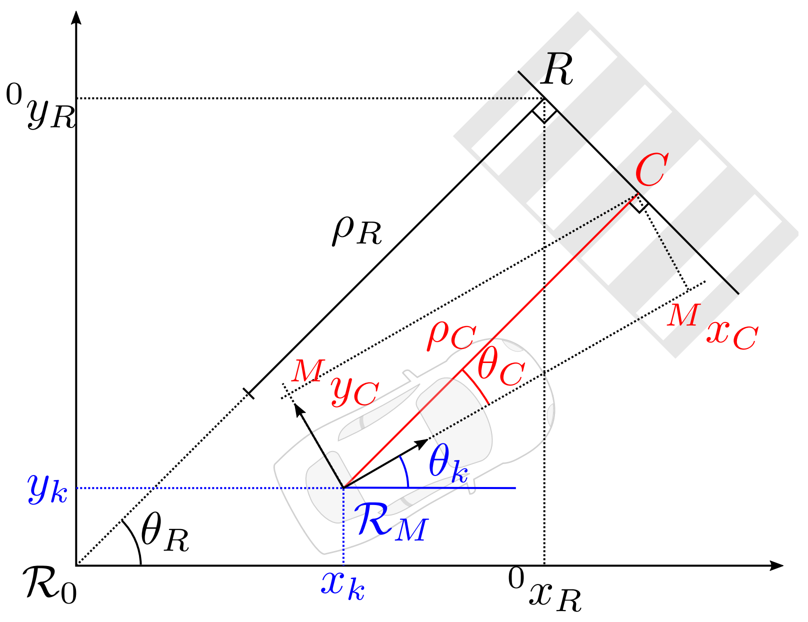


Figure E.1.: Schema of the observation model of a crosswalk.

For these reasons, the observation used for state estimation is not directly the two-dimensional point. One solution to deal with this problem could have been to keep the point as an observation but attribute it a large uncertainty in the direction of the pedestrian crossing. This solution is not perfect as some strips can be systematically missed by the detector (because of wear) thus leading to points systematically shifted in the same direction resulting in observations with correlated noises.

Instead, the detection is simplified in a single distance ρ_C from the vehicle to the crosswalk along the crosswalk longitudinal direction. This distance can be obtained because the orientation of the crosswalk can be measured (θ_C) and can be obtained from the map (θ_R). The observation used for the estimation is therefore,

$$\begin{bmatrix} \rho_C \\ \theta_C \end{bmatrix} = \begin{bmatrix} x_C \cos\theta_C + y_C \sin\theta_C \\ \theta_C \end{bmatrix} \tag{E.1}$$

The first term corresponds to the projection of the detected point on the crosswalk longitudinal direction. In other words, it corresponds to the distance along the crosswalk between the vehicle and the crosswalk. The same quantity can be obtained from the vehicle state by

$$h_i^{CW}(x) = \begin{bmatrix} \rho_R \\ \theta_R - \theta \end{bmatrix} \tag{E.2}$$

$$= \begin{bmatrix} (^0x_R - x_k)\cos\theta_R + (^0y_R - y_k)\sin\theta_R \\ \theta_R - \theta_k \end{bmatrix} \tag{E.3}$$

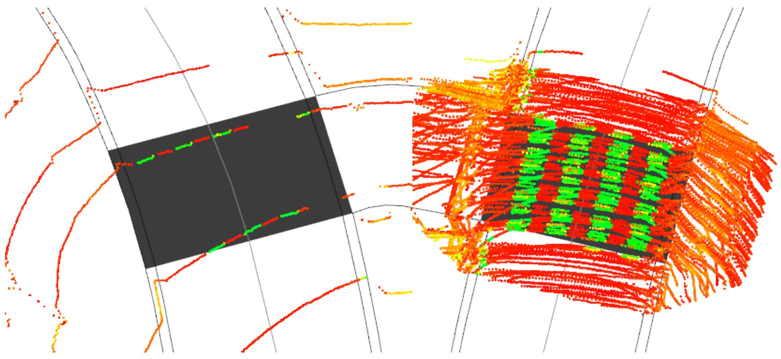


Figure E.2.: Comparison of the view of a single 16-layer lidar scan (a) and from an accumulation of 5 seconds of scans (b). Using a single scan, only two rings hit the crosswalk which can make it hard to detect. The crosswalk becomes clear when a buffer is used.

with corresponding Jacobian

$$H_i^{CW} = \begin{bmatrix} -\cos\theta_R & -\sin\theta_R & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{E.4}$$

The noise model of the detector can be defined by the two standard deviations σ_ρ and σ_{θ_C} forming the covariance matrix

$$R = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{E.5}$$

As for the other sensors, the numerical values for these standard deviations have been estimated by comparing the observation $\begin{bmatrix} \rho_C & \theta_C \end{bmatrix}$ with the output of the observation model $h_{CW}(\check{\boldsymbol{x}}_k)$ where $\check{\boldsymbol{x}}_k$ is the ground truth state.

E.2. Crosswalk Detector

The detection of pedestrian crossings is hard mainly due to the sparsity of the point cloud in the line-of-sight direction. This issue can be addressed using point cloud accumulation. Point cloud accumulation using grids has been used to successfully detect such feature [Yang et al., 2012]. Also dense mapping approaches use accumulation irrespective of the detectable features to perform localization. In this work, the accumulation is performed specifically to produce an observable and not as a direct step of the localization strategy.

Point cloud accumulation enables to obtain a denser point cloud in the vehicle travel direction. Depending on the size of the accumulation, it will have little effects on the vehicle sides, as shown on Figure E.2. Hence, pedestrian crossing located in front or behind the vehicle will become detectable using the accumulated point cloud.

The point cloud accumulation is performed using the vehicle kinematic state estimate $\begin{bmatrix} v_k & \dot{\theta}_k \end{bmatrix}^T$. Let $\mathcal{P}_k$ be the list of points obtained at time k (expressed in the mobile frame) and $\mathcal{P}_{k-S:k}$ the accumulated list of points from time $k-S$ to time k expressed in the mobile frame at time k. Each point is assumed to be described using its homogeneous coordinates $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^\top$. When a new point cloud $\mathcal{P}_{k+1}$ is obtained, the points from time $N-S$ are discarded from the buffer resulting in the point cloud $\mathcal{P}_{k-S+1:k}$ expressed in the mobile frame at time k. This point cloud needs to be propagated into the most recent mobile frame (at time $k+1$). This is done by using the transformation

$$^{k+1}\mathcal{P}_{k-S+1:k} = \begin{bmatrix} \cos\left(dt \cdot \dot{\theta}_{k+1}\right) & \sin\left(dt \cdot \dot{\theta}_{k+1}\right) & 0 & dt \cdot v_{k+1} \\ -\sin\left(dt \cdot \dot{\theta}_{k+1}\right) & \cos\left(dt \cdot \dot{\theta}_{k+1}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} {^k\mathcal{P}_{k-S+1:k}} \tag{E.6}$$

with $dt = t_{k+1} - t_k$.

The new accumulated point cloud is then obtained by adding the new points to $^{k+1}\mathcal{P}_{k-S+1:k}$

$$^{k+1}\mathcal{P}_{k-S+1:k+1} = {^{k+1}\mathcal{P}_{k-S+1:k}} \cup \mathcal{P}_{k+1} \tag{E.7}$$

The accumulated point cloud is estimated in a sliding window fashion every time a new point cloud is available.

Since modern lidars provide hundred thousands of points every second, the entire point cloud is not used for lidar detection. Instead, only an area of interest is processed. These areas of interest can be identified thanks to the map. The map contains polygons delimiting pedestrian crossing. The points near this polygon can thus be isolated to detect the crossing. This filtering can be performed using methods solving the point in polygon problem. With this method each polygon would need to be enlarged in order to account for the positioning error. Since this step is only used to reduce the computational complexity of the detection and is not required to be precise, a simpler method is used. If a point is in a polygon, the area of the triangles it forms with each segment of the polygon will equal the area of the entire polygon. However, if the point is outside of the polygon, some triangles will overlap resulting in a greater area than the polygon. Hence by checking the area of the triangles and the area of the polygon, one can verify if a point is within the polygon. This method also offers a simple way to dilate the polygon. Indeed, instead of requiring the two areas to be equal, by requiring the two surfaces to be close enough (as a proportion of the polygon surface) the points around the polygon might become acceptable. Hence the dilatation of the polygon can be controlled using a single, area proportion, factor.

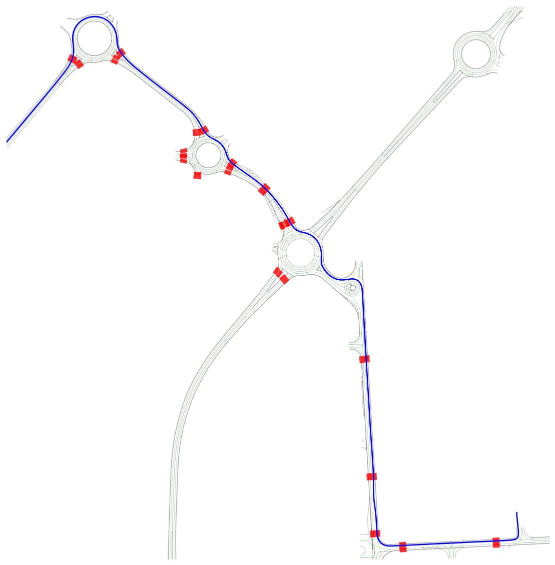


Figure E.3.: Trajectory (blue) used to evaluate the localization system using crosswalks. The crosswalks referenced in the map are shown in red.

Once the points are filtered around crossing candidates, the detection can be performed. Since the map only encodes the points where the road enters the crossing and exit the crossing, the method aims at detecting the middle of these points. To do so, the crosswalk strips are first isolated using Otsu thresholding [Otsu, 1979] on the intensity values as road paint is more reflective than asphalt. The resulting points are clustered and for each cluster (strip) its center and orientation are obtained using PCA. The final observation is then obtained by averaging the strips pose.

E.3. Experimental Results

To test the method, a trajectory in the city of Rambouillet was chosen (see Figure E.3). This trajectory was selected as it contained several crosswalks close to each other.

The vehicle used in these experiments is equipped with a Velodyne VLP-32C lidar and a ublox M8 GNSS receiver. In these experiments, lane markings and road sign measurements are not used such that the full influence of crosswalk measurements is not hidden by other measurements. The experiments are done with a 2 seconds buffer for the crosswalk detection.

Figure E.4 shows the differences in localization error between the localization with and without using crosswalk detections. When no crosswalk observations are available, both estimates tend to the same value as the ublox observations are driving

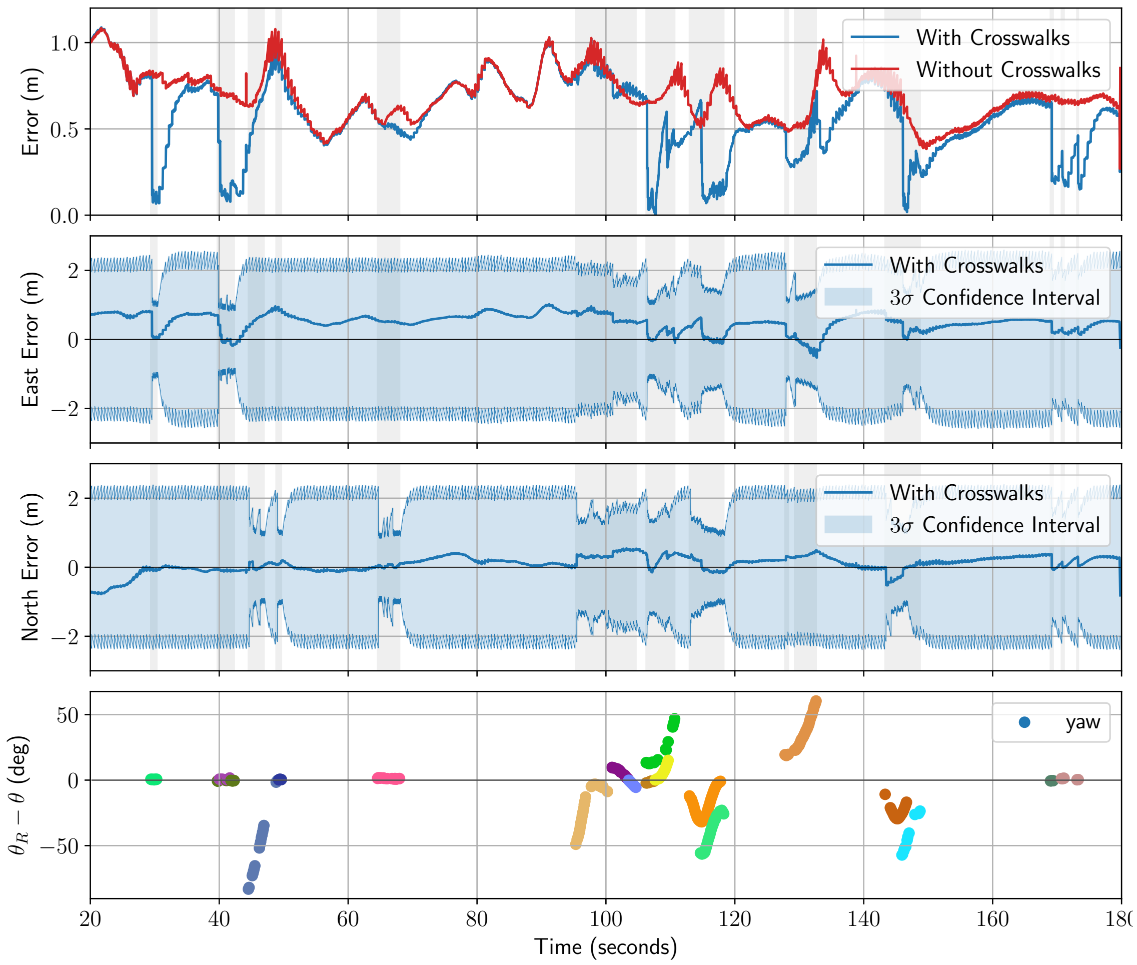


Figure E.4.: Localization error using crosswalk observations.

the estimated positions. Every time crosswalks are observed (light gray area in Figure E.4), the state uncertainty decreases along the crosswalk direction. In most cases, the error is also reduced. At $t = 30$ s, 40 s, 110 s, 145 s and 170 s, sharp decreases of the error are observed.

From these results, it can also be seen that crosswalks have difficulties improving the state estimates when they are viewed with a high relative angle ($\theta_R - \theta_k$). At $t = 45$ s, 95 s, 110 s and 130 s, the error is seen tending back toward the GNSS only error despite crosswalk observations.

The influence of the buffer length and of the dead reckoning accuracy on the accuracy of the observations has also been studied and can be found in [Lima et al., 2020].

Given that crosswalk detection can only be done using lidar when the road paint is fairly recent (crosswalks in Compiègne are much harder to detect) and correlation of observations (as crosswalks are detected using a rolling buffer, therefore, share points used for detection) still need to be studied. These observations are not yet integrated in the main localization system. The results shown in the main chapters of this thesis do not include these observations.

F. Covariance Intersection

F.1. Problem Statement

Covariance intersection enables the fusion of two sources of information with unknown correlation. Two observations $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ with corresponding covariance matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are combined into a single vector $\boldsymbol{y}$ and covariance matrix $\boldsymbol{S}$ using,

$$\boldsymbol{S} = \left(\omega \boldsymbol{S}_1^{-1} + (1-\omega)\boldsymbol{S}_2^{-1}\right)^{-1} \tag{F.1}$$

$$\boldsymbol{y} = \boldsymbol{S}\left(\omega \boldsymbol{S}_1^{-1}\boldsymbol{y}_1 + (1-\omega)\boldsymbol{S}_2^{-1}\boldsymbol{y}_2\right) \tag{F.2}$$

The factor $\omega \in [0,1]$ is chosen to minimize either $\det(\boldsymbol{S})$ or $\operatorname{trace}(\boldsymbol{S})$. In the general case, it has to be computed using iterative optimization techniques. For small state dimensions, analytic methods exist [Reinhardt et al., 2012]. In this thesis, covariance intersection is used with road sign residuals which are two-dimensional. A closed form solution is therefore useful to speed up the computations. We use here the determinant.

F.2. Covariance Matrix Transformation

In order to obtain an analytic expression of ω, an expression of $\det(\boldsymbol{S})$ is needed.

$$\det(\boldsymbol{S}) = \det\left(\left(\omega \boldsymbol{S}_1^{-1} + (1-\omega)\boldsymbol{S}_2^{-1}\right)^{-1}\right) \tag{F.3}$$

To easily compute the determinant of $\left(\omega \boldsymbol{S}_1^{-1} + (1-\omega)\boldsymbol{S}_2^{-1}\right)^{-1}$, a diagonalization of this expression is computed. This is achieved by first applying the transformation $\boldsymbol{T}_1 = \left(\boldsymbol{V}_1\sqrt{\boldsymbol{D}_1}\right)^{-1}$ where the diagonal matrix $\boldsymbol{D}_1$ and orthogonal matrix $\boldsymbol{V}_1$ are such that $\boldsymbol{S}_1 = \boldsymbol{V}_1\boldsymbol{D}_1\boldsymbol{V}_1^\top$. With this transformation,

$$\boldsymbol{T}_1\boldsymbol{S}_1\boldsymbol{T}_1^\top = \left(\boldsymbol{V}_1\sqrt{\boldsymbol{D}_1}\right)^{-1}\boldsymbol{S}_1\left(\boldsymbol{V}_1\sqrt{\boldsymbol{D}_1}\right)^{-\top} \tag{F.4}$$

$$= \sqrt{\boldsymbol{D}_1}^{-1}\boldsymbol{V}_1^{-1}\boldsymbol{S}_1\boldsymbol{V}_1^{-\top}\sqrt{\boldsymbol{D}_1}^{-\top} \tag{F.5}$$

$$= \boldsymbol{I} \tag{F.6}$$

$$\det\left(\boldsymbol{T}_1\boldsymbol{S}\boldsymbol{T}_1^\top\right) = \det\left(\boldsymbol{T}_1\left(\omega\boldsymbol{S}_1^{-1} + (1-\omega)\boldsymbol{S}_2^{-1}\right)^{-1}\boldsymbol{T}_1^\top\right) \tag{F.7}$$

$$= \det\left(\left(\omega\boldsymbol{T}_1^{-\top}\boldsymbol{S}_1^{-1}\boldsymbol{T}_1^{-1} + (1-\omega)\boldsymbol{T}_1^{-\top}\boldsymbol{S}_2^{-1}\boldsymbol{T}_1^{-1}\right)^{-1}\right) \tag{F.8}$$

$$= \det\left(\left(\omega\left(\boldsymbol{T}_1\boldsymbol{S}_1\boldsymbol{T}_1^\top\right)^{-1} + (1-\omega)\left(\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top\right)^{-1}\right)^{-1}\right) \tag{F.9}$$

$$= \det\left(\left(\omega\boldsymbol{I} + (1-\omega)\left(\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top\right)^{-1}\right)^{-1}\right) \tag{F.10}$$

A second transformation $\boldsymbol{T}_2$ is then used to diagonalize the matrix $\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top$, with $\boldsymbol{T}_2$ such that $\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top = \boldsymbol{T}_2^\top\boldsymbol{D}_2\boldsymbol{T}_2$ ($\boldsymbol{D}_2$ being a diagonal matrix).

$$\det\left(\boldsymbol{T}_2\boldsymbol{T}_1\boldsymbol{S}\boldsymbol{T}_1^\top\boldsymbol{T}_2^\top\right) = \det\left(\boldsymbol{T}_2\left(\omega\boldsymbol{I} + (1-\omega)\left(\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top\right)^{-1}\right)^{-1}\boldsymbol{T}_2^\top\right) \tag{F.11}$$

$$= \det\left(\left(\omega\boldsymbol{I} + (1-\omega)\left(\boldsymbol{T}_2\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top\boldsymbol{T}_2^\top\right)^{-1}\right)^{-1}\right) \tag{F.12}$$

$$= \det\left(\left(\omega\boldsymbol{I} + (1-\omega)\boldsymbol{D}_2^{-1}\right)^{-1}\right) \tag{F.13}$$

$$\tag{F.14}$$

The values of $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ do not depend on the value of ω. Moreover $\det(\boldsymbol{T}_1) > 0$ and $\det(\boldsymbol{T}_1) > 0$ because these matrices are built from orthogonal matrices and positive definite matrices. Therefore the minimization problem becomes,

$$\arg\min_\omega \det(\boldsymbol{S}) = \arg\min_\omega \frac{\det\left(\boldsymbol{T}_2\boldsymbol{T}_1\boldsymbol{S}\boldsymbol{T}_1^\top\boldsymbol{T}_2^\top\right)}{\det(\boldsymbol{T}_2\boldsymbol{T}_1)\det\left(\boldsymbol{T}_1^\top\boldsymbol{T}_2^\top\right)} \tag{F.15}$$

$$= \arg\min_\omega \det\left(\boldsymbol{T}_2\boldsymbol{T}_1\boldsymbol{S}\boldsymbol{T}_1^\top\boldsymbol{T}_2^\top\right) \tag{F.16}$$

$$= \arg\min_\omega \det\left(\left(\omega\boldsymbol{I} + (1-\omega)\boldsymbol{D}_2^{-1}\right)^{-1}\right) \tag{F.17}$$

$$= \arg\max_\omega \det\left(\omega\boldsymbol{I} + (1-\omega)\boldsymbol{D}_2^{-1}\right) \tag{F.18}$$

$$\tag{F.19}$$

Because $\boldsymbol{D}_2$ is diagonal, a simple expression of the determinant exists

$$\det\left(\omega\boldsymbol{I} + (1-\omega)\boldsymbol{D}_2^{-1}\right) = (\omega + \frac{1-\omega}{d_1})\cdot(\omega + \frac{1-\omega}{d_2}) \tag{F.20}$$

$$= \frac{(\omega(d_1-1)+1)\cdot(\omega(d_2-1)+1)}{d_1 d_2} \tag{F.21}$$

where d_1 and d_2 are the diagonal terms of the matrix $\boldsymbol{D}_2$.

When the previous expression is maximal, its first derivative is null. Therefore,

$$0 = \frac{d}{d\omega} \det\left(\omega \boldsymbol{I} + (1-\omega)\boldsymbol{D}_2^{-1}\right) \tag{F.22}$$

$$\Longleftrightarrow 0 = \frac{d}{d\omega}\left((\omega + \frac{1-\omega}{d_1}) \cdot (\omega + \frac{1-\omega}{d_2})\right) \tag{F.23}$$

$$\Longleftrightarrow 0 = \frac{d}{d\omega}\left((\omega d_1 + 1 - \omega) \cdot (\omega d_2 + 1 - \omega)\right) \tag{F.24}$$

$$\Longleftrightarrow 0 = 2(d_1-1)(d_2-1)\omega + (d_1+d_2-2) \tag{F.25}$$

$$\Longleftrightarrow \omega = \frac{d_1+d_2-2}{2(d_1-1)(d_2-1)} \tag{F.26}$$

F.3. Algorithm

Algorithm 1 Algorithm to compute ω for two-dimensional vectors.

```
 1: $[\boldsymbol{V}_1, \boldsymbol{D}_1] \leftarrow eigen(\boldsymbol{S}_1)$    ▷ Find first diagonalization
 2: $\boldsymbol{T}_1 = \left(\boldsymbol{V}_1\sqrt{\boldsymbol{D}_1}\right)^{-1}$    ▷ Compute the first transformation
 3: $[\boldsymbol{V}_2, \boldsymbol{D}_2] \leftarrow eigen(\boldsymbol{T}_1\boldsymbol{S}_2\boldsymbol{T}_1^\top)$    ▷ Compute second diagonalization
 4: $[d_1, d_2] \leftarrow diag(\boldsymbol{D}_2)$
 5: $\omega = -\frac{d_1+d_2-2}{(d_1-1)(d_2-1)}$
 6: if $\omega < 0$ then
 7:     $\omega \leftarrow 0$
 8: else if $\omega > 1$ then
 9:     $\omega \leftarrow 1$
10: end if
```

Bibliography

[Abdallah et al., 2008] Abdallah, F., Gning, A., and Bonnifait, P. (2008). Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44(3):807–815.

[Al Hage et al., 2020] Al Hage, J., Bonnifait, P., Xu, P., and Ibañez-Guzmán, J. (2020). Localization Integrity for Intelligent Vehicles through Fault Detection and Position Error Characterization. *IEEE Transactions on Intelligent Transportation Systems.*

[Anderson and Moore, 1979] Anderson, B. D. O. and Moore, J. B. (1979). *Optimal Filtering.* Prentice-Hall, Inc.

[Arasaratnam and Haykin, 2011] Arasaratnam, I. and Haykin, S. (2011). Cubature Kalman smoothers. *Automatica*, 47(10):2245–2250.

[Aubry et al., 2013] Aubry, C., Desmare, R., and Jaulin, L. (2013). Loop detection of mobile robots using interval analysis. *Automatica*, 49(2):463–470.

[Basnayake, 2009] Basnayake, C. (2009). A Novel Yaw Rate Sensor Bias Error Containment Method Using Existing Vehicle Sensors. *22nd International Meeting of the Satellite Division of The Institute of Navigation*, page 9.

[Berrio et al., 2019a] Berrio, J. S., Ward, J. R., Worrall, S., and Nebot, E. (2019a). Identifying Robust Landmarks in Feature-Based Maps. In *IEEE Intelligent Vehicles Symposium (IV)*, page 7, Paris, France.

[Berrio et al., 2019b] Berrio, J. S., Ward, J. R., Worrall, S., and Nebot, E. (2019b). Updating the Visibility of a Feature-Based Map for Long-Term Maintenance. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1173–1179, Paris, France.

[Bonnifait et al., 2001] Bonnifait, P., Bouron, P., Crubille, P., and Meizel, D. (2001). Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1597–1602, Seoul, South Korea. IEEE.

[Borenstein and Liqiang Feng, 1995] Borenstein, J. and Liqiang Feng (1995). Correction of systematic odometry errors in mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 569–574, Pittsburgh, PA, USA. IEEE Comput. Soc. Press.

[Brossard et al., 2020] Brossard, M., Bonnabel, S., and Barrau, A. (2020). Denoising IMU Gyroscopes with Deep Learning for Open-Loop Attitude Estimation. *IEEE Robotics and Automation Letters*.

[Broyden, 1970] Broyden, C. G. (1970). The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.

[Brunker et al., 2017] Brunker, A., Wohlgemuth, T., Frey, M., and Gauterin, F. (2017). GNSS-shortages-resistant and self-adaptive rear axle kinematic parameter estimator (SA-RAKPE). In *IEEE Intelligent Vehicles Symposium (IV)*, pages 456–461, Los Angeles, CA, USA. IEEE.

[Buckley, 1992] Buckley, D. (1992). *The GIS Primer: An Introduction to Geographic Information Systems*. Pacific Meridian Solutions, Incorporated.

[Bürki, Mathias et al., 2019] Bürki, Mathias, Schaupp, Lukas, Marcin, D., Dubé, Renault, Cadena, Cesar, Siegwart, Roland, and Nieto, Juan (2019). VIZARD: Reliable Visual Localization for Autonomous Vehicles in Urban Outdoor Environments. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1124–1130, Paris, France.

[Challa et al., 2002] Challa, S., Evans, R. J., Wang, X., and Legg, J. (2002). A fixed-lag smoothing solution to out-of-sequence information fusion problems. *Communications in Information and Systems*, 2(4):325–348.

[Chen et al., 2018] Chen, C., Lu, X., Markham, A., and Trigoni, N. (2018). Ionet: Learning to cure the curse of drift in inertial odometry. In *AAAI*.

[Chong et al., 2016] Chong, S., Rui, S., Jie, L., Xiaoming, Z., Jun, T., Yunbo, S., Jun, L., and Huiliang, C. (2016). Temperature drift modeling of mems gyroscope based on genetic-elman neural network. *Mechanical Systems and Signal Processing*, 72-73:897–905.

[Delobel, 2018] Delobel, L. (2018). *Agrégation d'information pour la localisation d'un robot mobile sur une carte imparfaite*. PhD thesis, Université Clermont-Auvergne.

[Doherty et al., 2019] Doherty, K., Baxter, D., Schneeweiss, E., and Leonard, J. (2019). Probabilistic Data Association via Mixture Models for Robust Semantic SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1098–1104.

[Domínguez Tijero et al., 2018] Domínguez Tijero, E., Martínez Fernández, L., Herrero Zarzosa, J. I., García, J., Ibanez-Guzman, J., Stawiarski, E., Xu, P., Avellone, G., Pisoni, F., Falletti, E., and Ortiz, M. (2018). High Accuracy Positioning Engine with an Integrity Layer for Safety Autonomous Vehicles. In *31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, pages 1566–1572, Miami, Florida.

[Drevelle and Bonnifait, 2013] Drevelle, V. and Bonnifait, P. (2013). Localization Confidence Domains via Set Inversion on Short-Term Trajectory. *IEEE Transactions on Robotics*, 29(5):1244–1256.

[Dube et al., 2017] Dube, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., and Cadena, C. (2017). SegMatch: Segment based place recognition in 3D point clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5272, Singapore. IEEE.

[Durrant-Whyte, 1988] Durrant-Whyte, H. (1988). Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, 4(1):23–31.

[EN 12899-1, 2007] EN 12899-1 (2007). Fixed, vertical road traffic signs.

[Fletcher, 1970] Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.

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

[Garcia et al., 2002] Garcia, R., Puig, J., Ridao, P., and Cufi, X. (2002). Augmented state Kalman filtering for AUV navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 4010–4015, Washington, DC, USA. IEEE.

[Ghallabi et al., 2019a] Ghallabi, F., Mittet, M.-A., El-Haj-Shhade, G., and Nashashibi, F. (2019a). LIDAR-Based High Reflective Landmarks (HRL)s For Vehicle Localization in an HD Map. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4412–4418, Auckland, New Zealand. IEEE.

[Ghallabi et al., 2018] Ghallabi, F., Nashashibi, F., El-Haj-Shhade, G., and Mittet, M.-A. (2018). LIDAR-Based Lane Marking Detection For Vehicle Positioning in an HD Map. In *21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2209–2214, Maui, HI. IEEE.

[Ghallabi et al., 2019b] Ghallabi, F., Shhade, G. E. H., Mittet, M.-A., and Nashashibi, F. (2019b). LIDAR-Based Road Signs Detection for Vehicle Localization in an HD Map. In *IEEE Intelligent Vehicles Symposium (IV)*, page 7, Paris, France.

[Gibbs, 2013] Gibbs, R. G. (2013). New Kalman filter and smoother consistency tests. *Automatica*, 49(10):3141–3144.

[Goldfarb, 1970] Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–23.

[Gong et al., 2013] Gong, X., Zhang, R., and Fang, J. (2013). Application of unscented R–T–S smoothing on INS/GPS integration system post processing for airborne earth observation. *Measurement*, 46(3):1074–1083.

[Gopalakrishnan et al., 2011] Gopalakrishnan, A., Kaisare, N. S., and Narasimhan, S. (2011). Incorporating delayed and infrequent measurements in Extended Kalman Filter based nonlinear state estimation. *Journal of Process Control*, 21(1):119–129.

[Harr and Schaefer, 2018] Harr, M. and Schaefer, C. (2018). Robust Dead Reckoning: Calibration, Covariance Estimation, Fusion and Integrity Monitoring. *CoRR*, abs/1801.02058.

[Hartmann et al., 2014] Hartmann, O., Gabb, M., Schweiger, R., and Dietmayer, K. (2014). Towards autonomous self-assessment of digital maps. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 89–95, MI, USA. IEEE.

[Hsiao and Kaess, 2019] Hsiao, M. and Kaess, M. (2019). MH-iSAM2: Multi-Hypothesis iSAM Using Bayes Tree and Hypo-Tree. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1274–1280, Montreal, QC, Canada.

[Hyyppa et al., 2009] Hyyppa, J., Jaakkola, A., Hyyppa, H., Kaartinen, H., Kukko, A., Holopainen, M., Zhu, L., Vastaranta, M., Kaasalainen, S., Krooks, A., Litkey, P., Lyytikainen-Saarenmaa, P., Matikainen, L., Ronnholm, P., Chen, R., Chen, Y., Kivilahti, A., and Kosonen, I. (2009). Map updating and change detection using vehicle-based laser scanning. In *Joint Urban Remote Sensing Event*.

[Jaulin, 2011] Jaulin, L. (2011). Range-Only SLAM With Occupancy Maps: A Set-Membership Approach. *IEEE Transactions on Robotics*, 27(5):1004–1010.

[Jiang et al., 2019] Jiang, S., Lu, D., and Cai, B. (2019). GNSS NLOS Signal Modeling and Quantification Method in Railway Urban Canyon Environment. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1140–1145, Paris, France.

[Jo et al., 2018] Jo, K., Kim, C., and Sunwoo, M. (2018). Simultaneous Localization and Map Change Update for the High Definition Map-Based Autonomous Driving Car. *Sensors*, 18(9):3145.

[Kaess et al., 2012] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235.

[Kaess et al., 2008] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.

[Kong et al., 2015] Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, Seoul, South Korea. IEEE.

[Lange et al., 2013] Lange, S., Sunderhauf, N., and Protzel, P. (2013). Incremental smoothing vs. filtering for sensor fusion on an indoor UAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1773–1778, Karlsruhe, Germany. IEEE.

[Lee and Woojin Chung, 2008] Lee, K. and Woojin Chung (2008). Calibration of kinematic parameters of a Car-Like Mobile Robot to improve odometry accuracy. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2546–2551, Pasadena, CA, USA. IEEE.

[Leonard and Durrant-Whyte, 1991] Leonard, J. and Durrant-Whyte, H. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS:IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, Osaka, Japan. IEEE.

[Leonard et al., 1992] Leonard, J. J., Durrant-Whyte, H. F., and Cox, I. J. (1992). Dynamic Map Building for an Autonomous Mobile Robot. *The International Journal of Robotics Research*, 11(4):286–298.

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

[Lima et al., 2020] Lima, A., Welte, A., Bonnifait, P., and Xu, P. (2020). Lidar observations by motion compensation and scan accumulation. In *16th International Conference on Control Automation Robotics & Vision (to be presented)*, Shenzhen, China. IEEE.

[Lundquist et al., 2014] Lundquist, C., Karlsson, R., Ozkan, E., and Gustafsson, F. (2014). Tire Radii Estimation Using a Marginalized Particle Filter. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):663–672.

[Magnusson, 2013] Magnusson, M. (2013). *The Three Dimensional Normal-Distributions Transform.Pdf.* PhD thesis, Örebro University, Örebro.

[Merlinge et al., 2016] Merlinge, N., Dahia, K., and Piet-Lahanier, H. (2016). A Box Regularized Particle Filter for terrain navigation with highly non-linear measurements. *IFAC-PapersOnLine*, 49(17):361–366.

[Merriaux et al., 2016] Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., and Savatier, X. (2016). Correction de nuages de points lidar embarqué sur véhicule pour la reconstruction d'environnement 3D vaste. In *Reconnaissance de Formes et Intelligence Artificielle (RFIA)*, Clermont-Ferrand, France.

[Montemerlo, 2003] Montemerlo, M. (2003). *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

[Munkres, 1957] Munkres, J. (1957). Munkres-variant-of-Hungarian-alg.pdf. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

[Nguyen et al., 2016] Nguyen, T. T., Spehr, J., Uhlemann, M., Zug, S., and Kruse, R. (2016). Learning of lane information reliability for intelligent vehicles. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 142–147, Baden-Baden, Germany. IEEE.

[Olson and Agarwal, 2013] Olson, E. and Agarwal, P. (2013). Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840.

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

[Pannen et al., 2019] Pannen, D., Liebner, M., and Burgard, W. (2019). HD Map Change Detection with a Boosted Particle Filter. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 7, Montreal, QC, Canada.

[Paul and Wan, 2008] Paul, A. S. and Wan, E. A. (2008). A new formulation for nonlinear forward-backward smoothing. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3621–3624, Las Vegas, NV, USA. IEEE.

[Radix, 1984] Radix, J. C. (1984). *Filtrage et Lissage Statistiques Optimaux Linéaires.* CEPADUES, Toulouse.

[Rajamani, 2006] Rajamani, R. (2006). *Vehicle Dynamics and Control.* Mechanical Engineering Series. Springer, New York, NY.

[Rauch et al., 1965] Rauch, H. E., Striebel, C. T., and Tung, F. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

[Reinhardt et al., 2012] Reinhardt, M., Noack, B., and Hanebeck, U. D. (2012). Closed-form optimization of covariance intersection for low-dimensional matrices. In *2012 15th International Conference on Information Fusion*, pages 1891–1896.

[Rohou et al., 2018] Rohou, S., Franek, P., Aubry, C., and Jaulin, L. (2018). Proving the existence of loops in robot trajectories. *The International Journal of Robotics Research*, 37(12):1500–1516.

[Rosen et al., 2013] Rosen, D. M., Kaess, M., and Leonard, J. J. (2013). Robust incremental online inference over sparse factor graphs: Beyond the Gaussian case. In *2013 IEEE International Conference on Robotics and Automation*, pages 1025–1032, Karlsruhe, Germany. IEEE.

[Rusu, 2010] Rusu, R. B. (2010). Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz*, 24(4):345–348.

[Schreiber et al., 2013] Schreiber, M., Knoppel, C., and Franke, U. (2013). LaneLoc: Lane marking based localization using highly accurate maps. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 449–454, Gold Coast City, Australia. IEEE.

[Shanno, 1970] Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111):647–647.

[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.

[Takeda et al., 2019] Takeda, Y., Tsuchiya, C., and Khiat, A. (2019). Ground Truth Generation for Quantitative Performance Evaluation of Localization Methods in Urban Areas. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1152–1158, Paris, France.

[Tao et al., 2017] Tao, Z., Bonnifait, P., Frémont, V., Ibanez-Guzman, J., and Bonnet, S. (2017). Road-Centered Map-Aided Localization for Driverless Cars Using Single-Frequency GNSS Receivers: Road-Centered Map-Aided Localization for Driverless Cars. *Journal of Field Robotics*, 34(5):1010–1033.

[Tijero et al., 2019] Tijero, E. D., Moreno, A. C., Calzón, M. F., García, J., Ibañez-Guzmán, J., Stawiarski, E., Xu, P., Avellone, G., Pisoni, F., Falletti, E., and Ortiz, M. (2019). Autonomous Vehicle High-Accuracy Position and Integrity Engine Performance Results. In *32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages 1234–1241, Miami, Florida.

[Tong, 1990] Tong, Y. L. (1990). *The Multivariate Normal Distribution.* Springer Series in Statistics. Springer, New York.

[Tran et al., 2017] Tran, T. A., Jauberthie, C., Gall, F. L., and Travé-Massuyès, L. (2017). Interval Kalman filter enhanced by positive definite upper bounds. *IFAC-PapersOnLine*, 50(1):1595–1600.

[Tsuchiya et al., 2019] Tsuchiya, C., Takeda, Y., and Khiat, A. (2019). A Self-Localization Method for Urban Environments using Vehicle-Body-Embedded Off-the-Shelf Sensors. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1159–1165, Paris, France.

[Varga et al., 2017] Varga, R., Costea, A., Florea, H., Giosan, I., and Nedevschi, S. (2017). Super-sensor for 360-degree environment perception: Point cloud segmentation using image features. In *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama. IEEE.

[Vivacqua et al., 2018] Vivacqua, R. P. D., Bertozzi, M., Cerri, P., Martins, F. N., and Vassallo, R. F. (2018). Self-Localization Based on Visual Lane Marking Maps: An Accurate Low-Cost Approach for Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):582–597.

[Wang et al., 2020] Wang, Z., Fang, J., Zhang, H., and Vlacic, L. (2020). Intelligent Vehicle Self-Localization Based on Double-Layer Features and Multilayer LIDAR. *IEEE Transactions on Intelligent Vehicles*.

[Wang and Lambert, 2018] Wang, Z. and Lambert, A. (2018). A Low-Cost Consistent Vehicle Localization Based on Interval Constraint Propagation. *Journal of Advanced Transportation*, 2018.

[Welte et al., 2019a] Welte, A., Xu, P., and Bonnifait, P. (2019a). Four-Wheeled Dead-Reckoning Model Calibration using RTS Smoothing. In *International Conference on Robotics and Automation (ICRA)*, pages 312–318, Montreal, QC, Canada. IEEE.

[Welte et al., 2019b] Welte, A., Xu, P., Bonnifait, P., and Zinoune, C. (2019b). Estimating the reliability of georeferenced lane markings for map-aided localization. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1225–1231, Paris, France. IEEE.

[Welte et al., 2020] Welte, A., Xu, P., Bonnifait, P., and Zinoune, C. (2020). Improved data association using buffered pose adjustment for map-aided localization. *IEEE Robotics and Automation Letters*, 5(4):6334–6341.

[Williams et al., 2014] Williams, S., Indelman, V., Kaess, M., Roberts, R., Leonard, J. J., and Dellaert, F. (2014). Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing. *The International Journal of Robotics Research*, 33(12):1544–1568.

[Wolcott and Eustice, 2015] Wolcott, R. W. and Eustice, R. M. (2015). Fast LIDAR localization using multiresolution Gaussian mixture maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2814–2821, Seattle, WA, USA. IEEE.

[Xiao et al., 2019] Xiao, Y., Ruan, X., Chai, J., Zhang, X., and Zhu, X. (2019). Online IMU Self-Calibration for Visual-Inertial Systems. *Sensors*, 19(7):1624.

[Yang et al., 2012] Yang, B., Fang, L., Li, Q., and Li, J. (2012). Automated Extraction of Road Markings from Mobile Lidar Point Clouds. *Photogrammetric Engineering & Remote Sensing*, 78(4):331–338.

[Zhao et al., 2016] Zhao, S., Chen, Y., and Farrell, J. A. (2016). High-Precision Vehicle Navigation in Urban Environments Using an MEM's IMU and Single-Frequency GPS Receiver. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2854–2867.

[Zinoune et al., 2012a] Zinoune, C., Bonnifait, P., and Ibanez-Guzman, J. (2012a). Detection of missing roundabouts in maps for Driving Assistance Systems. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 123–128, Alcal de Henares , Madrid, Spain. IEEE.

[Zinoune et al., 2012b] Zinoune, C., Bonnifait, P., and Ibanez-Guzman, J. (2012b). A sequential test for autonomous localisation of map errors for driving assistance systems. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1377–1382, Anchorage, AK, USA. IEEE.