Robust Crowdsourced Mapping for Landmarks-based Vehicle Localization

Alexis Stoven-Dubois

UNIVERSITÉ CLERMONT AUVERGNE
ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

THÈSE

Présentée par

ALEXIS STOVEN-DUBOIS
Master Robotique

pour obtenir le grade de

Docteur d'Université

Spécialité : **Électronique & Systèmes**

Robust Crowdsourced Mapping for Landmarks-based Vehicle Localization

Soutenue publiquement le 02/03/2022 devant le jury composé de :

VINCENT **FRÉMONT**	Rapporteur	Professeur à l'École Centrale de Nantes
VÉRONIQUE **CHERFAOUI**	Rapporteur	Professeure à l'Université Technologique de Compiègne
FAWZI **NASHASHIBI**	Examinateur	Directeur de Recherche à l'INRIA
AZIZ **DZIRI**	Encadrant de thèse	Chercheur à VEDECOM
BERTRAND **LEROY**	Encadrant de thèse	Chercheur à VEDECOM
ROLAND **CHAPUIS**	Directeur de thèse	Professeur à l'Université Clermont Auvergne

ACKNOWLEDGMENTS

During the duration of my thesis, I received a great deal of assistance and support. First, I would like to thank my thesis supervisor, Roland Chapuis, Professor at the Clermont Auvergne University. His scientific knowledge and practical experience, as well as his willingness to partake in frequent discussions, have encouraged me throughout this journey. Next, I would like to express sincere gratitude to my two thesis mentors, Aziz Dziri, Researcher at Vedecom, and Bertrand Leroy, Researcher at Vedecom. Their dedication to provide constant support, along with their research experience and technical insight, have allowed me to continuously progress within my work. I would also like to thank Kuntima Kiala Miguel, Engineer at Vedecom, who helped me gather and organize the data that I used for my experiments. Finally, I would like to thank my parents, Monique Stoven and Gérard Dubois, as their uninterrupted support was revealed to be essential towards the completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADAS Advanced Driver Assistance Systems

AR Auto-Regressive

CAM Cooperative Awareness Messages

CC Correlated Constraint

CC+GSD Correlated Constraint and Graph Subdivisions

CIF Covariance Intersection Filter

CPM Cooperative Perception Messages

DC Decorrelated Constraints

DENM Decentralized Events Notification Messages

DGPS Differential GPS

DOF Degrees Of Freedom

DOP Dilution Of Precision

EGNOS European Geostationary Navigation Overlay Service

EIF Extended Information Filter

EKF Extended Kalman Filter

ETSI European Telecommunications Standards Institute

GNSS Global Navigation Satellite Systems

GPS Global Positioning System

IMMF Interacting Multiple Models Filter

IMU Inertial Measurement Unit

IVIM Infrastructure to Vehicle Information Messages

KF Kalman Filter

LBAS Location-Based Augmentation Systems

Lidar Light detection and ranging

MAPEM MAP Extended Messages

MCDM Multimedia Content Dissemination Messages

MCM Maneuver Coordination Messages

ODD Operational Design Domain

PDF Probability Density Function

PF Particle Filter

Radar Radio detection and ranging

RLS Recursive Least-Squares

RTK-GNSS Real Time Kinematic-GNSS

SBAS Satellites-Based Augmentation Systems

SCIF Split Covariance Intersection Filter

SLAM Simultaneous Localization and Mapping

SPATEM Signal Phase And Timing Messages

UKF Unscented Kalman Filter

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

VANET Vehicle Ad-hoc Network

WAAS Wide Area Augmentation System

SUMMARY

The deployment of intelligent and connected vehicles, equipped with increasingly sophisticated equipment, and capable of sharing accurate positions and trajectories, is expected to lead to a substantial improvement of road safety and traffic efficiency. For this safety gain to become effective, vehicles will have to be accurately geo-positioned in a common reference, with an error up to a few decimeters [1]. To achieve this, they will be able to count on a variety of embedded sensors, such as GNSS (Global Navigation Satellite Systems) receivers, as well as additional proprioceptive and perception sensors. Nevertheless, in order to guarantee accurate positioning in all conditions, including in dense zones where GNSS signals can get degraded by multi-path effects, it is expected that vehicles will need to use precise maps of the environment to support their localization algorithms.

To build maps of the main highways, major automotive actors have made use of dedicated fleets of vehicles equipped with high-end sensors. Because of the associated high operational costs, they have been operating a limited number of vehicles, and remain unable to provide live updates of the maps and to register entire road networks. Crowdsourced mapping represents a cost-effective solution to this problem, and has been creating interest among automotive players. It consists in making use of measurements retrieved by multiple production vehicles equipped with standard sensors in order to build a map of landmarks. Nevertheless, while this approach appears promising, its real potential to build an accurate map of landmarks and maintain it up-to-date remains to be assessed in realistic, long-term scenarios.

In this thesis, in a first time, we propose a crowdsourced mapping solution based on triangulation optimization, and evaluate it using field-tests. The result analysis shows the potential of crowdsourced mapping to take advantage from measurements issued by multiple vehicles. On the other hand, it also indicates some critical limitations associated with triangulation optimization.

Therefore, in a second time, we propose another crowdsourced mapping solution based on graph optimization, and we introduce different approaches to include and update the map within the optimization, which correspond to different trade-offs between the map quality and computational scalability. Simulation experiments are conducted in order to compare the different approaches. The results enable to identify the most efficient one, and to assert that it provides a scalable solution for crowdsourced mapping.

The robustness of this solution to various types of noises, such as auto-correlated and biased noises, is then evaluated using extended simulation tests. The results analysis show its ability to build an accurate map of landmarks in various noises conditions, making use of measurements retrieved by multiple vehicles. Subsequently, field-tests are performed to confirm the results obtained in simulation, and draw conclusions both from a theoretical and practical viewpoint. Finally, the capacity of our crowdsourced mapping solution to increase the localization capabilities of vehicles is evaluated in simulation. The results show the effectiveness of the proposed approach to improve positioning performances in various conditions, while also pointing out the importance of providing a map with a sufficient density of landmarks.

RÉSUMÉ

Le déploiement de véhicules intelligents et connectés, dotés de capteurs de plus en plus sophistiqués, et capables de partager des positions et des trajectoires précises, permettra d'améliorer considérablement la sécurité routière et l'efficacité du trafic. Pour que ce gain de sécurité devienne effectif, les véhicules devront être géo-positionnés dans un référentiel commun avec précision, avec une erreur d'au plus quelques décimètres [1]. Pour y parvenir, ils pourront compter sur une variété de capteurs embarqués, tels que des récepteurs GNSS (Global Navigation Satellite Systems), ainsi que des capteurs proprioceptifs et des capteurs de perception. Toutefois, afin de garantir un positionnement précis dans toutes les conditions, y compris dans les zones denses où les signaux GNSS peuvent être dégradés par des effets de trajets multiples, les véhicules devront utiliser des cartes précises de l'environnement pour soutenir leurs algorithmes de localisation.

Afin d'établir de telles cartes pour les principales autoroutes, les principaux acteurs automobiles ont eu recours à des flottes de véhicules spécialisés équipés de capteurs haut de gamme. Cependant, en raison des coûts opérationnels élevés qui y sont associés, ils n'ont exploité qu'un nombre limité de véhicules et ne sont pas en mesure de fournir des mises à jour en direct des cartes, ni de cartographier des réseaux routiers entiers. La cartographie crowdsourcée représente une solution rentable à ce problème et suscite aujourd'hui l'intérêt des acteurs du secteur automobile. Cette technique consiste à exploiter les mesures récupérées par de multiples véhicules de production équipés de capteurs standard, afin de construire une carte contenant des points de repère. Néanmoins, même si cette approche semble prometteuse, sa capacité réelle à construire une carte précise et à la maintenir à jour a besoin d'être évaluée dans des scénarios réalistes et long-terme.

Dans cette thèse, nous proposons d'abord une solution de cartographie crowdsourcée basée sur une optimisation par triangulation, et l'évaluons à l'aide de tests de terrain. L'analyse des résultats montre le potentiel de cette approche à tirer profit des mesures

émises par plusieurs véhicules. Elle permet aussi d'identifier certaines limitations critiques associées à l'optimisation par triangulation.

Pour remédier à cela, nous proposons ensuite une autre solution de cartographie crowdsourcée basée sur l'optimisation de graphe, et nous introduisons différentes approches pour inclure et mettre à jour la carte dans l'optimisation, qui correspondent à différents compromis entre la qualité de la carte et la scalabilité. Des expériences de simulation sont menées afin de comparer ces approches. Les résultats permettent d'identifier la plus efficace, ainsi que de vérifier qu'elle représente une solution scalable de cartographie crowdsourcée.

La robustesse de cette approche à divers types de bruits, tels que les bruits auto-corrélés et biaisés, est ensuite évaluée à l'aide de tests de simulation étendus. L'analyse des résultats montre sa capacité à construire une carte précise dans diverses conditions de bruits, en utilisant des mesures récupérées par plusieurs véhicules. Ensuite, des tests de terrain sont effectués afin de confirmer les résultats obtenus en simulation, et de tirer des conclusions tant d'un point de vue théorique que pratique. Enfin, la capacité de notre solution de cartographie crowdsourcée à améliorer les capacités de localisation des véhicules est évaluée en simulation. Les résultats montrent l'efficacité de l'approche proposée dans diverses conditions, tout en soulignant l'importance de fournir une carte avec une densité suffisante de points de repère.

CHAPTER 1
INTRODUCTION

During the last 25 years, road safety has motivated car makers and tier-one suppliers to design and market increasingly sophisticated equipment, in order to enhance both the safety of car drivers and passengers, as well as surrounding vehicles and pedestrians. The development of automotive vehicles has boosted this effort and led to important innovations in the domain of ADAS (Advanced Driver Assistance Systems) such as lane departure warning, adaptive cruise control, advanced emergency braking. At the same time, public authorities are currently pushing for the deployment of connected vehicles, one of the main benefits of which is to prevent crashes by continuously broadcasting vehicles positions and trajectories. In this context, a key issue consists in accurately geolocalizing the vehicles, i.e. localizing them in a common reference. To achieve this goal, great efforts are currently being made by numerous automotive players to provide vehicles with a high-precision map allowing for precise positioning.

1.1 Vehicles Localization and High-Precision Maps

GNSS, such as GPS (Global Positioning System), GLONASS (Russian equivalent), Baidu (Chinese equivalent) or Galileo (European equivalent), have represented the conventional solution used to geo-position vehicles. However, they are not able to provide an accuracy with errors up to a few decimeters, which is estimated necessary to enhance vehicles safety through the sharing of accurate positions and trajectories [1]. In fact, most production vehicles are expected to be equipped with standard GNSS receivers, which have a typical accuracy of a few meters in open conditions [2], and up to a few dozens of meters in dense environments [3]. Even the more costly RTK-GNSS (Real Time Kinematic-GNSS) can have their signals disturbed in dense zones, leading to unavailability and inaccuracy issues [4].

To reach the targeted accuracy, another solution consists in making use of a high-precision map, providing knowledge about the road environment. The high-precision map would contain accurate positions of various road landmarks, such as traffic signs and road markings, and would allow to:

- Enhance the vehicles perception beyond their sensing capabilities, by providing them information about their environment.
- Serve as a common reference frame, enabling the vehicles to express their positions and trajectories relatively to the map, and thereby improve their positioning accuracy.
- Improve efficiency of the data-sharing, by making vehicles able to only update or download map information from a given geographical region.

When positioning a given vehicle, a data fusion process would merge various sensors observations, including GNSS measurements, and camera images matched with the high-precision map. Such data fusion process would typically require that the high-precision map contains not only accurate geo-positions of detected landmarks, but also their estimated accuracy [5]. Several companies are building high-precision maps, making use of dedicated fleets of vehicles equipped with high-end sensors. As of 2021, they are able to provide high-precision maps with a decimeter-level accuracy, but only for restricted areas such as major highways. Indeed, there is no viable business model allowing for the deployment of vehicles equipped with high-end sensors on the whole road network, especially if the map must be continuously updated for safety reasons.

1.2 Crowdsourced Mapping

In order to bypass the issues associated with building a high-precision map over large areas and keeping it up-to-date, the work of this thesis focuses on a crowdsourced mapping strategy that relies on production vehicles. This method consists in collecting measurements from multiple vehicles equipped with standard sensors, such as a standard GNSS receiver

and a monocular camera, in order to build and update a map collaboratively, by taking advantage from the frequent and redundant measurements issued by the vehicles.

Although crowdsourced mapping represents a cost-effective solution for building a high-precision map, it aggregates information from multiple vehicles on a long-time span, and thus poses a number of specific challenges. For instance, vehicles must possess image processing modules that are robust to various environmental conditions, in order to consistently detect and match landmarks with the map. While the use of high-level landmarks, such as traffic lights and road markings, facilitate this process [6], robust landmarks detection and recognition generally remains a challenge that is not completely solved.

Further, the crowdsourced mapping process must be robust to various types of noises affecting the measurements, in order to merge observations appropriately, and compute accurate map updates. In convenient configurations, such as when noises affecting the measurements are sampled from zero-mean distributions, simply collecting the measurements will eventually lead to accuracy improvements of the map. However, in real conditions, sensors measurements are often affected by auto-correlated or biased noises, which contradicts this assumption, and may affect the accuracy of the resulting map. To deal with this issue, the typical strategy consists in making use of a data fusion process that merges measurements from multiple sensors, taking into account their respective accuracy. Simultaneous Localization and Mapping (SLAM) methods, which aim to estimate at the same time the trajectory of a given agent and the positions of detected landmarks, have been widely used to process such data fusion. While SLAM methods have been based on both filtering techniques and graph optimization, the latter has demonstrated a better capacity to build accurate maps of landmarks, with more robustness to various types of noises configurations [7, 8]. For this reason, graph optimization represents a powerful tool for long-term map construction, and could be used by crowdsourced mapping to continuously build and update the high-precision map.

1.3 Contributions

In this thesis, we aim to develop and evaluate a crowdsourced mapping system based on graph optimization. Given an image processing module that provides robust detection and recognition of road-oriented landmarks such as traffic signs, we propose a crowdsourced mapping pipeline that takes advantage from measurements issued by multiple vehicles to build a map of geo-positioned landmarks. We evaluate this system by assessing both the map accuracy and potential scalability obtained throughout extensive simulation and field experiments. The contributions of this thesis are as follow:

- We propose and develop a crowdsourced mapping pipeline, in which measurements retrieved from multiple vehicles are merged to build and update a map of landmarks. Each time a vehicle uploads new measurements, centralized servers apply an optimization procedure to merge these measurements, and compute a new version of the map. To assess the potential of the collaborative approach, we first aim to verify that the map accuracy effectively improves as more vehicles upload their measurements. We build a solution based on a simple triangulation optimization, and evaluate it using field experiments. The results of this evaluation were presented in [9].

- We improve our crowdsourced mapping pipeline, by making use of graph optimization instead. Towards this goal, we propose and compare three different approaches to adapt graph optimization to the collaborative context. In the first approach, we include all map content within graph optimization. This approach increases the map accuracy at the potential cost of intensive computations. To alleviate calculations, we propose a second approach, in which we neglect cross-correlations between landmarks in the map. Finally, in another attempt to lower computations, we propose a third approach in which each graph optimization problem is subdivided into multiple subproblems that are solved sequentially. To identify the most efficient method, we compare these different approaches during simulation experiments. The results of this comparison

were presented in [10].

- We evaluate the performances that can be achieved with crowdsourced mapping in real conditions. Towards this goal, we drive our test-vehicle in a dense city center, for multiple passages along a loop of a few kilometers, in order to build a map of geo-positioned traffic signs. We evaluate the resulting map accuracy and computational scalability, and conclude on the potential of crowdsourced mapping. The results of this evaluation were presented in [3].

For a complete evaluation of our crowdsourced mapping solution, we extended simulation experiments with several types of noises, such as auto-correlated and biased noises, in order to evaluate their effective impact on the map accuracy. We compared the obtained results with the results from field-tests, and drew conclusions both from a theoretical and practical viewpoint. Furthermore, we evaluated in simulation the potential of our approach to improve the localization performances of the vehicles through the establishment of an accurate map of geo-positioned landmarks. These results and discussion are presented in the remainder of this document.

[9]: A. Stoven-Dubois et al., *"A Collaborative Framework for High-Definition Mapping,"* in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Oct.2019, pp. 1845–1850.

[10]: A. Stoven-Dubois et al., *"Graph Optimization Methods for Large-Scale Crowdsourced Mapping,"* in 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Jul. 2020, pp. 1–8.

[3]: A. Stoven-Dubois et al., *"Graph-based approach for crowdsourced mapping: Evaluation through field experiments,"* in 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2020, pp. 260–265.

1.4 Document Organization

This document is organized as follows: in chapter 2, we present the applicative context of intelligent and connected vehicles, while in chapter 3, we discuss existing localization and mapping methods. In chapter 4, we recall the basic formulation of graph optimization, and we introduce our crowdsourced mapping solution. Finally, chapter 5 contains the description and results from simulation and field experiments, during which we evaluate the performances of crowdsourced mapping.

CHAPTER 2
BACKGROUND ON INTELLIGENT VEHICLES

Currently, as of 2021, production vehicles are being equipped with multiple sensors, as will be the case for future autonomous vehicles. These sensors include proprioceptive sensors, such as motor encoders or IMU (Inertial Measurement Unit), and exteroceptive sensors, such as Radar sensors or monocular cameras. They enable equipped vehicles to position themselves in space, while detecting surrounding obstacles, and have led to the development of various active ADAS functions on current production vehicles. In parallel, production vehicles are being equipped with communication devices that will allow them to communicate positions with each other, as well as with the road infrastructure. All of these developments are expected to bring significant improvements for road safety, at the condition that vehicles can be localized accurately, and share meaningful position information. This motivates our work on crowdsourced mapping, in order to build a map that can be used to support the localization of vehicles.

In the present chapter, we provide some background about intelligent vehicles, in order to define the application scope of crowdsourced mapping. First, we discuss the different autonomy levels that can characterize driving vehicles. Next, we present the main sensors, both proprioceptive and exteroceptive, that can be used. Then, we talk about the different communication means expected for connected vehicles. Finally, we give a brief overview of existing map formats.

2.1 From ADAS to Autonomy

Road safety has been a major concern of car makers and customers for a long time, and this has led to the continuous development of sensors embedded in the vehicles. More recent sensors have provided measurements of better quality (both in terms of quantity and

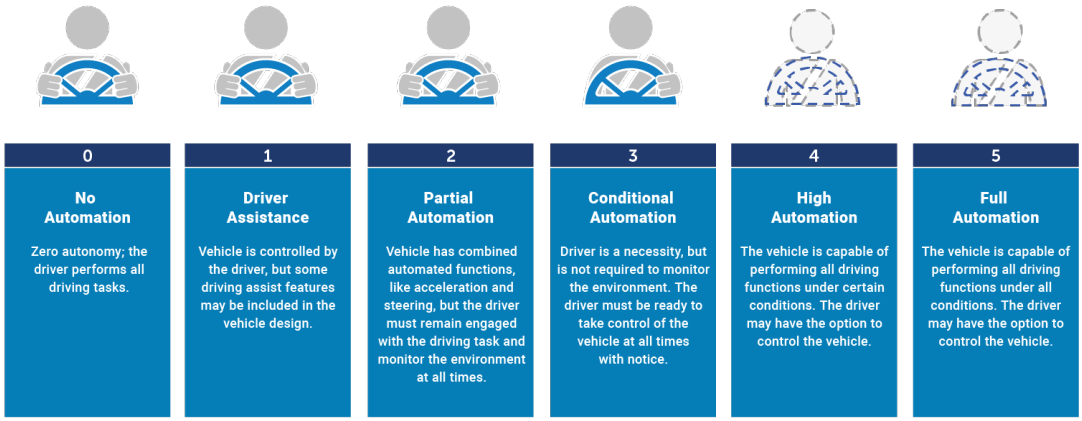

Figure 2.1: Vehicles autonomy levels defined by the Society of Automotive Engineers, taken from [11].

accuracy), and are generally more robust to various environmental conditions. However, they are also typically associated with more complex algorithms, which has motivated the embedding of additional computational resources within the vehicles.

These developments have increased, and will continue to increase, the autonomy capacities of driving vehicles, by allowing them to perceive their environment, detect potentially dangerous situations, and take subsequent driving actions. Following, several levels of autonomy have been envisioned, in order to characterize the autonomy capacities of different vehicles [11]. A summary of these autonomy levels is shown in Figure 2.1. Currently, most production vehicles are equipped with standard ADAS, which correspond to level 1 of autonomy. These ADAS include passive functions (e.g. lane departure warning, forward collision alert), and active functions (e.g. adaptive cruise control, anti-lock breaking system). Some high-end vehicles, such as those produced by Tesla, have reached level 2 of autonomy, by being capable of self-driving in restricted ODD (Operational Design Domain) under the supervision of a human driver [12].

Overall, whether we consider a standard vehicle of autonomy level 1, a future self-driving vehicle of autonomy level 5, or any vehicle in-between, the main autonomy operations processed by a given intelligent vehicle can be summarized in four main tasks: Perception,

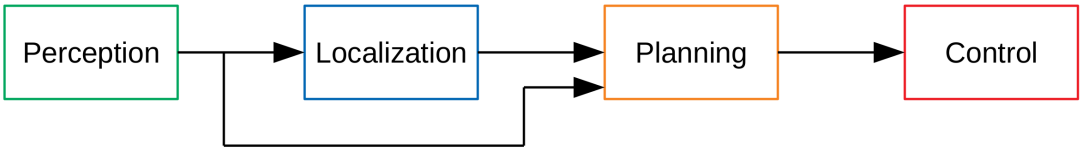

Figure 2.2: Broad pipeline of autonomy operations processed by an intelligent vehicle.

Localization, Planning and Control. These four tasks are illustrated in Figure 2.2, and are briefly discussed in the following.

2.1.1 Perception

The perception task aims at making the vehicle able to perceive and understand its surroundings. This is done through the detection of various items, such as road objects (e.g. traffic lights, traffic signs), static objects (e.g. stationed vehicles, buildings), moving objects (e.g. other driving vehicles, pedestrians). By detecting those, the vehicle can acknowledge its driving environment, and detect potentially dangerous situations, whether originating from its own behavior or from the behavior of other agents.

As inputs for the perception task, the vehicle can count on measurements provided by a range of different perception sensors. Production vehicles typically embed a few affordable sensors, such as ultrasound sensors, as well as monocular cameras [13]. Meanwhile, high-quality vehicles tend to be equipped with the same affordable sensors, along with more costly ones such as Radar and Lidar sensors [14].

2.1.2 Localization

The localization task aims at positioning the vehicle, either with respect to its environment (local positioning), or in a global reference (global positioning). As local positioning provides the vehicle with spatial knowledge about its immediate surroundings, it is a clear requirement for various driving tasks, such as collision avoidance and self-driving operations. In fact, it is estimated that an accuracy with errors up to a few decimeters is necessary to

ensure safe behavior whenever an intelligent vehicle takes driving decisions [1]. To achieve accurate local positioning, the vehicle can count on a variety of proprioceptive sensors (e.g. motor encoders, IMU), which measure its relative movements, as well as outputs from the perception task, that inform about the trajectory of nearby objects relatively to the vehicle.

Alternatively, global positioning (also called geolocalizing or geo-positioning) consists in positioning the vehicle in a global reference. This reference is common to all vehicles and road infrastructure, and enables to express all known positions and trajectories in a universal system. Especially, in the case where vehicles can communicate with each other, accurate global positioning (with errors up to a few decimeters) can bring substantial benefits to road safety and traffic efficiency, by allowing the different vehicles to communicate their positions and trajectories [15]. To achieve global positioning, the vehicle can traditionally use a GNSS receiver, which provides global position measurements.

2.1.3 Planning

The planning task aims at identifying optimal driving decisions that lead the vehicle to drive efficiently, while maintaining itself in the right lane and avoiding potential collisions. To achieve this, outputs from the perception and localization tasks are used, in order to inform the vehicle about its own situation, as well as the situation of surrounding objects.

On production vehicles equipped with standard ADAS, planning functions are generally restricted to simple procedures, such as Advanced Cruise Control, or Active Lane Keeping Assist. On more advanced vehicles, planning functions can include more complex operations, such as Autonomous Overtaking or Active Collision Avoidance, which require the vehicle to predict both its own trajectory, as well as the future behavior of nearby agents. In such cases, the accuracy of local positioning, which provides information about the trajectory of surrounding objects relatively to the vehicle, is of critical importance [1].

2.1.4 Control

Finally, the control task aims at transforming driving decisions provided by the planning task into effective driving actions. This is achieved by computing and actuating appropriate driving commands (e.g. acceleration, braking, driving wheel rotation) on the vehicle. Obviously, due to the vehicle model being typically imprecise, and due to the presence of various physical effects, including friction between the wheels and the ground, as well as noises affecting the actuating system, the actual vehicle movements are typically different from the intended ones. Therefore, the vehicle behavior must be constantly overseen and corrected with new actuation commands through a closed-loop approach.

2.2 Proprioceptive Sensors

In the following, we give a brief overview of the main types of sensors embedded in vehicles that provide perception and localization information. We start with proprioceptive sensors, which inform about the relative motion of vehicles, and play a crucial role towards their localization.

2.2.1 Motor Encoders

Motor encoders are sensors of mechanical motion that have been widely used in production vehicles, in order to measure movements (speed or angle) from a variety of moving items. In fact, motor encoders that measure the wheels velocities and the driving wheel angle are of particular importance when localizing a given vehicle. Indeed, they provide measurements related to the actual actuation of the vehicle, which can be then used inside a motion model to estimate and predict the vehicle displacements between successive instants. Multiple motion models have been applied in vehicular applications [16], and can be classified into two main categories:

- Linear models: CV (Constant Velocity), CA (Constant Acceleration)

- Curvilinear models: CTRV (Constant Turn Rate and Velocity), CTRA (Constant Turn Rate and Acceleration), CSAV (Constant Steering Angle and Velocity), CCA (Constant Curvature and Acceleration).

Obviously, the estimation of the vehicle displacements is always noisy up to a given level. This is mainly due to the presence of various physical effects undergone by the vehicle that can not be considered in the motion model [17], as well as to the presence of noises that affect the encoders measurements. To position a vehicle using only measurements from motor encoders, the conventional method, dead-reckoning, consists in fixing the initial position of the vehicle, and then integrating estimations of the vehicle displacements. Naturally, as multiple noisy estimations get integrated, their errors tend to add up, which leads the vehicle localization to diverge in the long-run, i.e. have an error that grows unbounded with time [18].

2.2.2 Inertial Measurement Unit

In order to measure the relative motion of a vehicle, another solution consists in making use of inertial sensors, which can be classified into three categories:

- Accelerometers, which measure accelerations.
- Gyroscopes, which measure angular velocities.
- Magnetometers, which measure the strength and direction of the surrounding magnetic field.

By associating 3 accelerometers and 3 gyroscopes, such that accelerations and angular velocities can be measured in all 3 dimensions, we obtain a 6-DOF (Degrees Of Freedom) IMU. And by also including a magnetometer to perceive the magnetic field, we get a 9-DOF IMU. Currently, production vehicles are mostly equipped with low-cost accelerometers and gyroscopes, which are used to detect strong vehicle movements and enable electronic

stability control [19]. Meanwhile, production vehicles generally do not embed IMU, as those remain costly sensors that are mainly used within high-quality vehicles.

To merge measurements provided by an IMU, and estimate the relative motion of a given equipped vehicle, an IMU model must be used. Similarly as for motor encoders, the IMU model can not account for all physical effects undergone by the vehicle, and noises affect IMU measurements, which makes the estimation of the vehicle displacements always noisy up to a given level. As the typical method used to localize a vehicle using only IMU measurements is dead-reckoning (as is the case with encoders measurements), the resulting localization tends to have an error that grows unbounded with time. Furthermore, due to a variety of physical effects (e.g. temperature, mechanical stress), the internal calibration of an IMU generally tends to diverge with time, even when the vehicle is stopped, which can lead to inaccurate measurements. Nevertheless, it is generally admitted that IMU-based localization is more accurate than encoders-based localization, although few research works have been realized to compare these two methods.

2.3 Exteroceptive Sensors

In addition to proprioceptive sensors, intelligent vehicles also embed exteroceptive sensors that enable them to perceive their environment and detect nearby objects, and which can be used for both localization and perception tasks, as explained in subsection 2.1.1 and subsection 2.1.2.

2.3.1 GNSS Receivers

A GNSS receiver computes precise time information and global position measurements using signals received from a given satellites network (e.g. GPS, GLONASS, Beidou, Galileo). To achieve this, it applies a trilateration-based technique that requires communication with at least 4 satellites to solve for 4 unknowns (3 unknowns for the 3-dimensional position, and 1 unknown for the time). Nevertheless, while travelling from satellites to a given receiver,

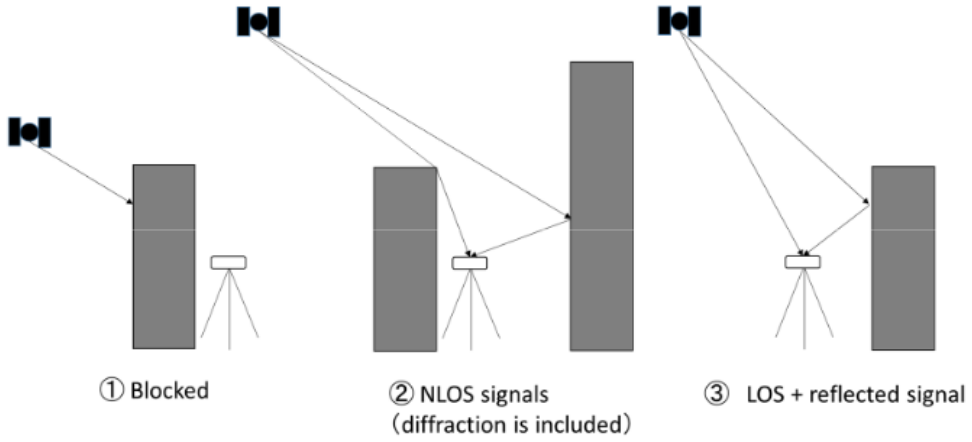

Figure 2.3: Different effects of GNSS multipath, taken from [20].

GNSS signals can get degraded by various environmental effects. In open environments, the main source of degradation consists in the delaying of GNSS signals when travelling the ionosphere and troposphere. In dense environments, another source of degradation comes into play, and consists in the blocking, diffraction and reflection of GNSS signals on various buildings and structures [20]. This phenomenon is known as multi-path, and is illustrated in Figure 2.3. In the case where GNSS signals emitted by a satellite get blocked, they are not used by the GNSS receiver. Therefore, such effect does not impact the accuracy of position measurements, as long as there remains enough other satellites to communicate with. Oppositely, when GNSS signals get diffracted or reflected, they are still considered by the GNSS receiver to compute the position measurements, which can lead to large position errors.

In fact, errors associated with position measurements provided by a standard GNSS receiver are usually comprised between $4\ m$ and $5\ m$ in open environments [2], and can reach dozens of meters in dense environments [3]. To improve the positioning accuracy, several enhanced systems have been designed:

- Multi-frequency systems, which attempt to correct GNSS errors due to atmospheric layers by making each satellite emit two or more signals at different frequencies [21].

- SBAS (Satellites-Based Augmentation Systems), such as EGNOS (European Geostationary Navigation Overlay Service) and WAAS (Wide Area Augmentation System) [22]. These systems make use of ground stations to avoid the degradation of signals due to atmospheric layers, and compute and emit corrections directly from satellites.

- LBAS (Location-Based Augmentation Systems), such as DGPS (Differential GPS) and RTK-GPS [22]. These systems also make use of ground stations for the same purpose, but oppositely to SBAS, they emit corrections from the ground stations, and make GNSS receivers communicate constantly with them.

- Multi-constellation systems, which communicate with more satellites by considering several satellites networks simultaneously, in order to make GNSS measurements more robust to situations in which signals from only a few satellites get degraded [21].

Generally, these systems achieve better positioning accuracy than standard GNSS receivers in all types of environments. Nevertheless, they still get affected by multi-path issues, which worsen their performances in dense environments. For instance, RTK-GPS receivers, which are usually considered as the most accurate systems, reach up to a centimeter-level accuracy in open environments [23], but can also suffer from severe inaccuracy and unavailability issues in dense environments [4].

So far (in 2021), production vehicles are generally equipped with standard GNSS receivers, due to their affordability. Meanwhile, they are usually not equipped with RTK-GPS receivers, as those require communication with a base station that often requires costly subscriptions to some services (e.g. Sogelink, Geotopo in France). Other enhanced systems, such as multi-constellation and multi-frequency solutions, constitute affordable options that can be envisioned to be widely embedded in production vehicles in the near future. Nevertheless, while such systems will certainly help to support the localization of future intelligent vehicles, they do not address some issues that prevent vehicles from being positioned with the targeted accuracy (with errors up to a few decimeters), especially in

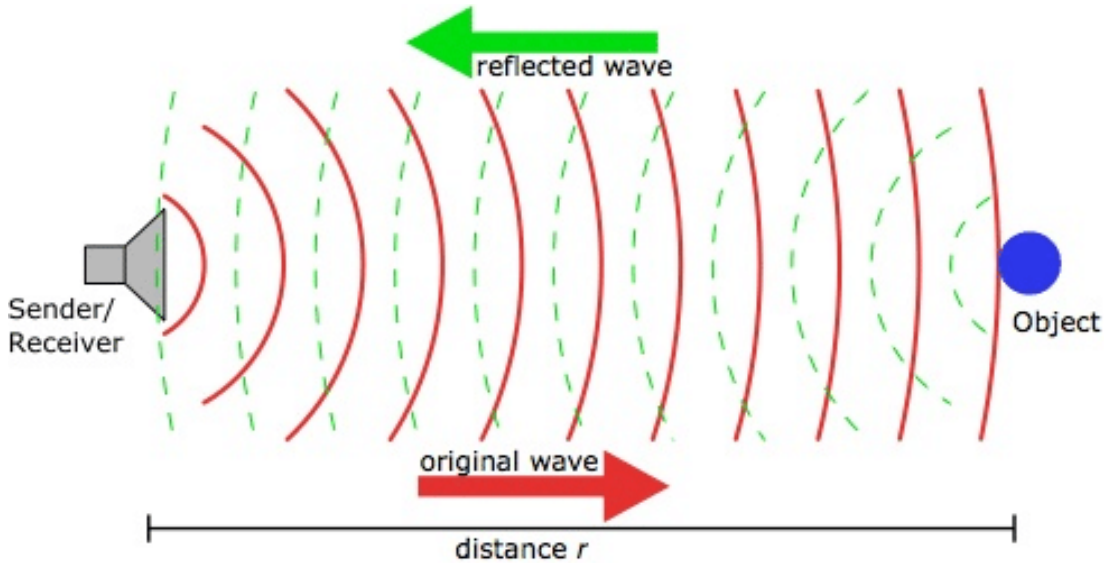

Figure 2.4: Functioning principle of an ultrasonic sensor, taken from [24]. The sensor emits ultrasound signals, which are reflected on the object.

dense environments where GNSS signals get degraded through multi-path effects.

2.3.2 Ultrasonic Sensor

An ultrasonic sensor is an active perception sensor that provides range measurements about surrounding obstacles. It functions by emitting ultrasound signals, and waiting for their echo after they get reflected on a nearby object, as illustrated in Figure 2.4. By using the time between emission and reception of the signals, it is able to compute distance measurements to the given object [24]. Furthermore, by making use of the Doppler effect, some ultrasonic sensors are also capable of estimating the velocity of the detected object. Generally, ultrasound signals can get severely degraded by travelled distance, due to various environmental conditions such as temperature, wind or humidity. Furthermore, they can also get affected by the presence of soft materials, which have peculiar reflection properties. Nevertheless, they remain robust to rough illumination and weather conditions when the travelled distance is short (up to a few meters).

Overall, standard ultrasonic sensors have been affordable for many years, and are now

part of the standard equipment of production vehicles. They have enabled some ADAS functions that require short-range detection measurements and that are now widely employed in production vehicles, such as blind spot detection and park assist [25].

2.3.3 Radar Sensor

A Radar (Radio detection and ranging) sensor is an active perception sensor that functions similarly as an ultrasonic sensor, but using radio waves instead of ultrasound signals, in order to provide range and velocity measurements. Radio waves are generally more robust than ultrasound signals to various environmental conditions, and the typical maximum range of Radar sensors can reach a few dozens of meters for standard models, and several hundreds of meters for high-quality models. Nevertheless, in dense environments, such as in a city center or within a traffic jam, radio waves often get affected by rebound effects, which can lead Radar sensors to provide measurements with a low accuracy and resolution at long distances.

Although Radar sensors are generally too expensive for standard vehicles, they have been widely employed in high-quality ones. By providing long-range measurements, they have enabled equipped vehicles to detect obstacles in advance, and have led to the development of various ADAS functions, such as adaptive cruise control and forward collision warning [25].

2.3.4 Lidar Sensor

A Lidar (Light detection and ranging) sensor is another type of active perception sensor that operates similarly as ultrasonic and Radar sensors, but making use of light waves instead, in order to provide range and velocity measurements. Generally, Lidar sensors are more sensible to environmental conditions than Radar sensors, and can be unable to function under strong weather occurrences such as fog or snow. They also include moving mechanical parts that typically make them fragile and subject to vibrations undergone by the equipped vehicles. Nevertheless, they typically provide more accurate measurements than Radar

sensors, and with a longer maximum range that can reach a few hundreds of meters.

Currently, Lidar sensors remain expensive systems that are mainly used in research vehicles, and not in production ones. They provide long-range and accurate measurements, and have enabled the development of advanced ADAS, such as collision avoidance systems [26]. The upcoming arrival of solid-state Lidar sensors, which do not have any moving mechanical part and are expected to be sold at a much lower price, might address the cost issue and allow for the wide deployment of Lidar sensors. Nevertheless, this remains an open question, as some car makers (e.g. Volvo, Google) consider that Lidar sensors will soon become part of the standard equipment, while other companies (e.g. Tesla) expect the opposite.

2.3.5 Monocular Camera

A monocular camera is a passive perception sensor that provides visual images of the surrounding environment. Most cameras are able to detect lights rays in the visible spectrum, and produce images of best quality during the day or within lit environments. Meanwhile, other types of cameras can detect other ranges of wavelengths, such as infrared cameras that provide rich images during the night or within dark environments. To transform light rays into images, cameras make use of a converging lens whose shape dictates both the angle of view and the capacity of the camera to magnify distant objects. Thus, while telephoto lenses typically have a restricted angle of view, wide-angle lenses produce shrink in the images. Between these two extremes, medium lenses that equip most monocular cameras typically have a fair opening angle of view (about $120\ °$), and produce small distortion effects on the border of images.

Monocular cameras do not provide any range or velocity information, and therefore must be used in conjunction with image processing algorithms, in order to extract useful information from the images. For illustration purposes, we show in Figure 2.5 an image captured by the monocular camera installed on our test-vehicle, along with the detection of

Figure 2.5: Detection of a traffic sign within an image issued by the monocular camera installed on our test-vehicle.

a traffic sign depicted as a cyan square. While image processing algorithms require available computational resources on the vehicles, and can be sensitive to various illumination and weather conditions, cameras remain very affordable sensors that can provide rich visual information about the surrounding environment. They are now part of the standard equipment of production vehicles, and have enabled many ADAS functions such as lane keeping assist and collision warning systems [25].

Furthermore, while monocular cameras do not provide range measurements, they can be associated in groups of two (stereo-camera), or more than two (multi-camera systems), in order to reconstruct the 3D environment through image processing algorithms, and produce range information [27]. Such systems remain rarely used in standard vehicles, but some companies, such as Tesla, have been employing them in high-quality vehicles. To retrieve range information, another solution consists in making use of a depth camera, which associates a monocular camera with a depth sensor functioning similarly as a Lidar sensor, by emitting and receiving infrared signals. Nevertheless, depth cameras typically have a limited maximum range, and have been rarely used within vehicular applications.

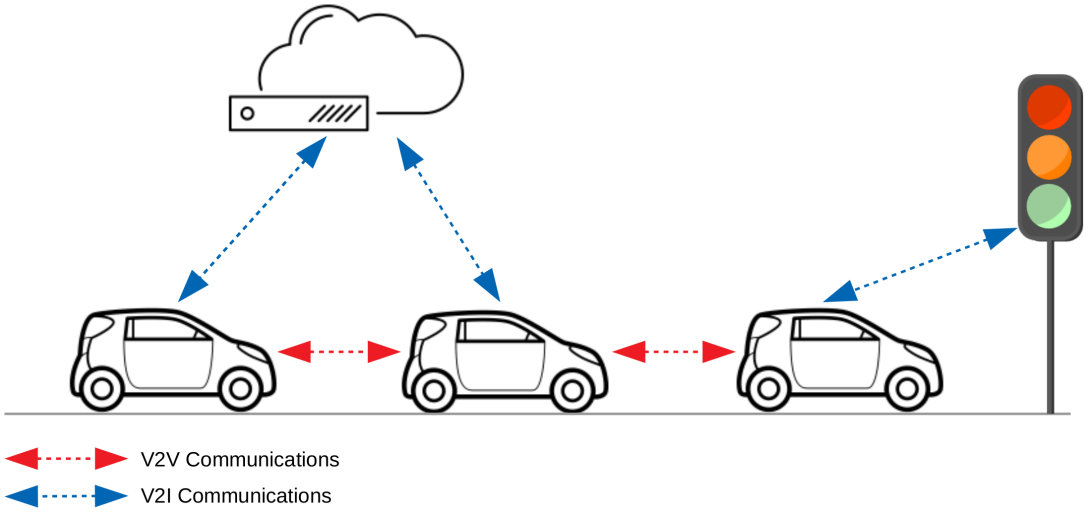

Figure 2.6: Communications between connected vehicles (V2V) and with the road infrastructure (V2I).

2.4 Connected Vehicles

In parallel to the development of vehicles equipped with more sensors and capable of more autonomy, efforts have also been directed towards the enhancement of road connectivity. This has been initiated by the embedding of various communication devices (e.g. Wifi modules, cellular antennas) within the vehicles, as well as within the road infrastructure. Thanks to these devices, it is expected that vehicles will soon be able to communicate in real time with other vehicles in V2V (Vehicle-to-Vehicle), and with the infrastructure in V2I (Vehicle-to-Infrastructure), as depicted in Figure 2.6. The subsequent benefits can be classified under three main aspects:

- Improvement of road safety through inter-vehicles communications (e.g. collision avoidance, lane change warning).
- Enhancement of traffic efficiency (e.g. traffic lights control, collective planning).
- Extension of in-vehicle services (e.g. anti-theft tracking, internet access).

Hereafter, we give a brief description of existing messages formats, and we follow with

the description of the different types of communications (V2V and V2I).

2.4.1 Communication Formats

To allow the emergence of connected vehicles, two different communication technologies are being considered:

- Cellular communications, making use of the incoming 5G network.
- Wifi communications, using the ITS-G5 protocol defined in Europe [28] and the 802.11p protocol defined in the USA [29].

Independently from the technology that will be used, different messages formats have already been defined by ETSI (European Telecommunications Standards Institute) [30], in order to standardize future connectivity solutions:

- CAM (Cooperative Awareness Messages) contain localization information (e.g. position, trajectory).
- CPM (Cooperative Perception Messages) contain perception information (e.g. pedestrian detection).
- MCM (Maneuver Coordination Messages) contain driving information (e.g. lane change warning, overtake warning).
- MAPEM (MAP Extended Messages) contain geographic information (e.g. road curve).
- SPATEM (Signal Phase And Timing Messages) contain information related to intersections (e.g. traffic light state).
- IVIM (Infrastructure to Vehicle Information Messages) contain infrastructure information (e.g. road signs).

- MCDM (Multimedia Content Dissemination Messages) contain multimedia content.
- DENM (Decentralized Events Notification Messages) are dedicated to emergency cases (e.g. informing about a road accident).

2.4.2 V2V Connectivity

Inter-vehicle communications, also called V2V communications, represent the exchange of information between different vehicles. This can be achieved through direct connection between nearby vehicles, through indirect links between distant vehicles using a VANET (Vehicle Ad-hoc Network), or through internet connection using Wifi or cellular network. By communicating with each other, vehicles could share precious information, and allow for substantial improvements of road safety [15]. For instance, they could exchange perception and localization information, such that each individual vehicle could improve its own understanding of the environment far beyond the scope of its own embedded sensors. They could also share driving information, such as lane departure warning or collision detection warning, in order to allow for collaborative planning between nearby vehicles. Therefore, while several challenges remain for V2V communications, such as dealing with severe ambient noise, high vehicle mobility, and multiple vehicles attempting to connect at the same time [15], intensive efforts have been initiated to tackle these challenges in the short future.

2.4.3 V2I Connectivity

Vehicle-to-infrastructure communications, also called V2I communications, represent the sharing of information between vehicles and the road infrastructure, which is expected to progressively integrate road objects equipped with communication devices, such as traffic lights embedding a cellular antenna. Thus, vehicles could communicate with the infrastructure either through direct connection with nearby road objects, or through internet connection. They could share useful information with the infrastructure but also with other

distant vehicles, using the infrastructure as a relay. The vehicles could share information such as driving decisions and traffic situation (e.g. blocked road, traffic accident), allowing the infrastructure to react and apply collective fleet management in order to improve the overall traffic efficiency. The infrastructure could also integrate perception sensors, and provide vehicles with information that could reach beyond their own sensing capabilities, in order to improve road safety. For instance, a camera could be installed at the top of a traffic light, and benefit from a privileged angle of view to detect dangerous situations. Similarly as for V2V communications, challenges for V2I connectivity include dealing with the road environment conditions, and with multiple concurrent communications [15]. Again, thorough efforts have been initiated to allow V2I connectivity in the short future.

2.5 Maps

In order to provide vehicles with more information that can be perceived by their sensors, and to support their perception and localization algorithms, many applications have considered the use of maps of the environment. With the emergence of enhanced connectivity, such maps also become an efficient tool to share various information between vehicles and the infrastructure, by providing a common positioning reference, as well as a standardized format for storing data. Nowadays, multiple automotive players consider the availability of a highly accurate map as an essential step towards the deployment of autonomous and connected vehicles [31]. Such a map would be organized into different map levels, which would contain different types of information, including both static data (e.g. road shape, traffic signs) and dynamic data (e.g. vehicles positions, lane change warnings). This map organization would allow efficient information sharing, as each vehicle would be able to access the map only for a given type of information, and in a given geographical region.

In the following, we describe the main types of map formats that have been used in vehicular applications, in order to model the road environment and enhance vehicles capacities.

2.5.1 Topological Map

A topological map is a map containing a set of nodes that encode available information at specific positions in the environment, and a set of vertices that model relations between the nodes [32]. Topological maps typically lack scale and direction, and instead model the environment through a given topology, by associating vertices with given cost measures. In vehicular applications, topological maps are mainly dedicated to path-planning, which aims to compute the targeted trajectory for a vehicle to reach its destination. Therefore, they generally model the road network by encoding distance measures within the vertices, while the information contained in the nodes depends on the targeted application. For instance, popular topological maps include road maps such as Google Maps and OpenStreetMap, which encode road information within the nodes, and street-view maps such as Google Street View, which encode rich visual information within the nodes.

2.5.2 Grid-based Map

A grid-based map is a continuous representation of the environment that divides the space into squares (for 2D maps) or cubes (for 3D maps) of pre-defined dimensions. Each square or cube is a cell that can encode any type of information. In vehicular applications, grid-based maps have been widely used to support the navigation of vehicles and perform collision avoidance, by informing about which part of the environment is likely occupied by obstacles, and which part is free. To achieve this, they encode within each cell the likeliness to find an obstacle there, either using a binary format or discrete probabilities, and must be constantly updated using new sensors measurements and obstacles detections. In such maps, which are also called occupancy grids [33], an important challenge resides in defining appropriately the size of cells, as those must not be too large, to avoid over-occupying space whenever an obstacle is detected, and not too small, to limit computations and allow for real-time map updates.

2.5.3 Landmarks Map

A landmarks map is a discrete representation of the environment that consists of a set of positioned points called landmarks, which can represent any given type of object detected in the environment, such as high-level items (e.g. traffic signs, buildings) [34], and low-level items (e.g. interest points) [35]. Generally, such map also contains the estimated accuracy of landmarks positions, which is typically in the form of a covariance matrix. Landmarks maps have been widely used to enhance vehicles capacities, by providing a variety of information related to the surrounding environment and nearby objects. They have enabled vehicles to compare and match their sensors measurements with the map content, and thus enhance their perception and localization capabilities. To properly merge the information brought by sensors measurements with the map, data fusion techniques typically make use of the estimated accuracy of the map [5]. Therefore, in order to be useful, landmarks maps need to be accurate, i.e. contain precise landmarks positions, but also consistent, i.e. provide a valid estimated accuracy.

2.6 Conclusion

Road safety has for long been a major concern in the automotive industry, and has motivated car makers to develop various ADAS functions for production vehicles. This has been supported by the embedding of proprioceptive and exteroceptive sensors that have enabled both passive and active ADAS functions, such as lane departure warning and adaptive cruise control, to be part of the standard equipment. Even now, thorough efforts are still being done to develop more ADAS functions, and to make production vehicles evolve towards more autonomy.

In parallel to such developments, production vehicles are being equipped with communication devices that will soon enable them to communicate with each other and the road infrastructure. This is expected to have a substantial impact on road safety, by allowing the

vehicles to share in real-time useful driving information (e.g. positions, obstacles detections, driving decisions). Nevertheless, in order to exchange useful position data, production vehicles will need to be accurately geolocalized (i.e. with errors up to a few decimeters).

To geolocalize production vehicles, GNSS receivers have been of particular interest, as they allow to directly retrieve geo-position measurements. However, neither standard GNSS receivers nor their improved versions have been able to guarantee the targeted accuracy in dense environments, where GNSS signals can get severely impacted by multi-path effects. Therefore, accurate geolocalization of connected and intelligent vehicles in all environments remains an open challenge.

CHAPTER 3
RELATED WORK

To localize themselves with precision and accuracy, intelligent vehicles can count on measurements issued by a variety of embedded proprioceptive and exteroceptive sensors. To complete such measurements, it is also expected that they will make use of pre-built maps that contain prior knowledge about the road environment. This strategy requires the prior establishment of a highly-accurate map, which is maintained up-to-date, and spans over entire road networks. To build such a map, a crowdsourcing approach making use of observations issued by multiple vehicles appears as a cost-effective strategy.

In this chapter, we review localization and mapping techniques in the context of intelligent and connected vehicles, with the purpose of positioning our contributions relatively to existing methods. This chapter is organized as follows. In section 3.1, we discuss localization methods for intelligent vehicles. And in section 3.2, we present map-building strategies that enable to build maps of the road environment.

3.1 Localization

Localization consists in estimating the position and trajectory of a vehicle relatively to a given reference, which can be either a global or a local one. It plays an essential role for intelligent vehicles, by allowing them to assess their situation, navigate and avoid accidents.

Initial works focused on estimating the lateral position of driving vehicles on the road. These methods have relied on the detection of road markings [36], and have enabled multiple ADAS functions, such as lane departure warning systems [37]. Nevertheless, by only providing lateral positions, they could not position the vehicles in space nor estimate their trajectories. Subsequent works aimed to achieve global positioning, also called geolocalization or geo-positioning, of the vehicles. This consists in estimating the position of each vehicle

in a global reference, such as an Earth-fixed frame. If a map of the environment is available, the vehicles can position themselves within the map, and gain additional knowledge about their surroundings [31]. Furthermore, by positioning themselves within a common reference, they can potentially communicate their positions and trajectories with each other and the road infrastructure, which can bring substantial benefits towards road safety and traffic efficiency [15]. Nonetheless, in order to provide meaningful information, and thereby effectively improve driving conditions, it has been evaluated that intelligent vehicles must be geolocalized with errors up to a few decimeters [1].

In this section, we discuss localization methods that have been used to geo-position driving vehicles. In subsection 3.1.1, we present methods based on the use of measurements issued by the sensors embedded in vehicles. In subsection 3.1.2, we discuss the use of SLAM techniques. And in subsection 3.1.3, we present localization methods that benefit from a previously-established map.

3.1.1 Multi-Sensor Localization

In the following, we discuss localization methods that enable to geo-position a vehicle, making use of measurements obtained from embedded proprioceptive and exteroceptive sensors.

GNSS Positioning To geolocalize a vehicle, GNSS receivers appear as the most obvious solution, as they provide direct geo-position information. As discussed more in details previously (see subsection 2.3.1), there exist multiple types of GNSS receivers, which correspond to different levels of performances and prices. The optimal configuration of GNSS receivers consists of open environments, where GNSS signals can travel without encountering obstacles. In such conditions, standard GNSS receivers can reach an accuracy of a few meters [2], while more advanced GNSS receivers, such as RTK-GNSS receivers or PPP-GNSS receivers, can reach a centimeter-level accuracy [38]. Nevertheless, in dense

environments, such as urban city centers, the quality of GNSS signals can get severely affected by multi-path effects for long periods of time, resulting in unavailability and inaccuracy issues. In such conditions, even advanced models of GNSS receivers can reach errors of a few meters [39], while standard GNSS receivers can reach errors of dozens of meters [3]. To deal with this, different strategies have attempted to address multi-path effects. In [40], a map of the environment was used to identify and reject unreliable GNSS signals. In [41], the auto-correlated noises affecting GNSS signals were modeled using an Auto-Regressive (AR) process. Both of these methods could retrieve GNSS positions with errors up to a few meters, making use of standard GNSS receivers within dense environments. Although this represents an improvement of positioning accuracy, this is still below the targeted accuracy for intelligent vehicles. Furthermore, making use only of a GNSS receiver would generally be insufficient, as GNSS signals can become unavailable.

Adding Proprioceptive Sensors To deal with the degradation of GNSS signals, some works have attempted to fuse them, generally using a filtering technique, with displacement measurements obtained from proprioceptive sensors. In [42], a vehicle is geo-positioned by fusing measurements from a GNSS receiver, an IMU and motor encoders through a single KF (Kalman Filter). In [43], the same type of measurements are merged through either an EKF (Extended Kalman Filter) or an EIF (Extended Information Filter), while in [44], they are merged applying either an EKF or an IMMF (Interacting Multiple Models Filter), which fuses outputs from several KF run in parallel. In fact, making use of the IMMF, [44] could geo-position a vehicle with errors up to a few decimeters in case of smooth motion (constant speed), and with a meter-level accuracy in case of rough motion (strong acceleration or turning).

Overall, merging GNSS signals with displacement measurements enables to improve the positioning accuracy, and allows to deal with situations in which GNSS signals become unavailable, by performing dead-reckoning with the displacement measurements [42].

Nevertheless, dead-reckoning tends to have errors that grow unbounded with time, due to the fact that errors associated with displacement measurements get integrated [45]. Therefore, while this strategy can be efficient against temporary outages of GNSS signals, it is not capable to answer the challenges met by GNSS signals within dense environments, where inaccuracy and unavailability issues can last for long periods of time.

Adding Perception Sensors In a second approach, other works aimed to deal with the degradation of GNSS signals by making use of measurements issued by perception sensors. They did so by computing displacement measurements from perception observations (e.g. camera images), and then fusing these measurements similarly as if they were obtained from proprioceptive sensors. In both [46] and [47], a vehicle equipped with a stereo-camera and a GNSS receiver is geolocalized with errors up to a few meters. To achieve this, low-level visual features are detected within camera images, and used to compute displacement measurements of the vehicle. Then, these displacement measurements are merged with GNSS signals through an EKF.

Overall, this strategy enables to improve the positioning accuracy, as well as to perform dead-reckoning whenever GNSS signals become unavailable. Nevertheless, its performances heavily rely on the algorithms that are used to process perception measurements, such as the detection and matching algorithms used by [46] and [47] to process camera images. This can make the positioning accuracy sensitive to environmental effects, such as illumination conditions when using camera images. In fact, the obtained performances do not seem to challenge the performances obtained when directly using proprioceptive sensors instead. Furthermore, dead-reckoning based on perception sensors suffers from the same shortcoming as dead-reckoning based on proprioceptive sensors, which is to have errors that grow unbounded with time. Thus, this strategy is also unable to deal with the long-term degradation of GNSS signals present in dense environments.

In order to effectively address this issue, other methods have attempted to make use of

maps of the road environment. They aimed to geo-position vehicles by matching observations issued by their perception sensors with the information contained in the maps. These methods are generally based on the use of SLAM, and are introduced in the following.

3.1.2 Simultaneous Localization and Mapping

The unavailability of GNSS signals has for long been a challenge for the robotics community, due to the fact that many robotics applications take place either indoor, or in dense environments outdoor. In the 1980s, the robotics community has shifted its focus towards relative positioning [45], which consists in localizing a robot within its environment. The objective was to take advantage from the fact that a given environment contains many static objects, such as traffic signs or buildings in the case of a driving environment, which can be detected by the robot and used as localization anchors called landmarks. As a result, SLAM methods have been developed to build a map of the environment while simultaneously localizing a robot within this map. Starting from the 2000s, SLAM techniques have been adapted and used for vehicular applications [48]. Due to the fact that vehicles evolve in outdoor environments, and are generally equipped with GNSS receivers, maps have started to include geo-positioned data. Subsequently, relative positioning associated with SLAM methods has shifted towards global positioning, and has aimed to position a given vehicle relatively to a map that contains geolocalized data.

To solve the SLAM problem, multiple methods have been developed within the robotics field. In the following, we define the typical SLAM framework, introduce existing SLAM methods, and discuss their practical use for intelligent vehicles. When presenting a SLAM method in the general case, we refer to any given robot or vehicle as "agent".

SLAM Framework

The SLAM problem consists in estimating simultaneously the trajectory of a given agent (localization) and the positions of mapped objects (mapping). By processing localization

and mapping concurrently, the aim is that the localization can profit from the mapping output and inversely. To achieve this, measurements provided by embedded proprioceptive and exteroceptive sensors are used. To fuse the different measurements, which are typically of different nature, and affected by different noises levels, the SLAM problem is formalized as a probabilistic or optimization problem [49, 50]. To solve this problem, different steps are needed, in order to process the measurements, merge the information, and operate localization and mapping. SLAM methods can generally be split between three main operations: landmarks detection, data association and estimation.

Landmarks detection aims to reliably detect the mapped landmarks within perception measurements (e.g. Lidar scans, camera images). The landmarks can potentially take any form, depending on the desired map content. Low-level landmarks are generally associated with specific types of measurements (e.g. pointclouds obtained from LiDAR scans [51], image features extracted from camera images [52]). Meanwhile, high-level landmarks, which correspond to real objects such as traffic signs and buildings, can be found in applications with various types of sensors [53, 54].

Data association consists in identifying which landmarks detections relate to the same landmarks, and to match these detections with landmarks in the map. This step includes tracking, which associates detections between subsequent measurements, and loop recognition, which links detections obtained at separate moments (meaning the agent has returned to a place it has visited previously). Generally, tracking is operated by matching detections between successive measurements, potentially making use of a motion model for the agent [52]. Meanwhile, loop recognition algorithms typically depend on the type of landmarks that are used (e.g. Iterative Closest Point to match pointclouds [51], place recognition algorithms to match image features [55]).

Estimation aims to solve the SLAM problem, i.e. estimate the agent trajectory and landmarks positions. To achieve this, it makes use of landmarks detections and corresponding associations, as well as measurements obtained from proprioceptive sensors. Two different

strategies exist for the formulation of this problem: full SLAM and online SLAM [4]. In full SLAM, the objective is to estimate the whole agent trajectory at once, along with the positions of all detected landmarks [50]. Thus, full SLAM makes use of all retrieved measurements at once, and leads to an optimal accuracy, but also to a potentially high computational complexity. To achieve real-time performances, online SLAM was proposed, in which the estimation step is divided in two operations: local estimation and loop closure [55]. The local estimation is activated at a high-frequency and estimates only the last positions of the agent, as well as the positions of the lastly detected landmarks. Meanwhile, whenever a loop is recognized through data association, a loop closure step is run as a background process, in which a substantial part of the agent trajectory and landmarks positions are estimated and updated, in order to account for the loop recognition. To solve the SLAM problem, multiple techniques have been proposed for both full SLAM and online SLAM, and can be classified into two main categories: filtering-based methods and optimization-based methods. These two categories are discussed in the following.

Filter-based SLAM

SLAM methods based on the use of filters are probabilistic approaches that estimate a state, which comprises the agent trajectory and the landmarks positions, as well as the associated accuracy in the form of covariances. These methods follow a two-step iterative process that is activated periodically whenever new measurements are received. At each step, they only make use of the last measurements to update the estimation, and are thus mostly used to solve online SLAM.

The two-step process, independently of the used filter, runs as follows. During the prediction step, the filter state is predicted, making use of the preceding estimation, as well as a motion model associated with displacement measurements obtained from proprioceptive sensors (e.g. motor encoders, IMU). During the update step, this prediction is corrected to obtain the final estimation, making use of measurements and sensors models associated

with exteroceptive sensors (e.g. GNSS receiver, Lidar sensor, camera). Generally, SLAM methods based on filters originate from two main strategies: the KF (Kalman Filter) [56] and the PF (Particle Filter) [57], which are discussed below.

The Kalman Filter (KF) The KF was designed to handle linear systems affected by zero-mean, Gaussian noises, and KF-based SLAM has shown strong convergence properties in such conditions [56]. However, in practice, robots and vehicles are generally associated with non-linear motion models, and their embedded sensors also possess non-linear sensors models. Therefore, instead of applying the original KF, many SLAM methods have used improved versions of it: EKF (Extended Kalman Filter) [58, 59], UKF (Unscented Kalman Filter) [60, 48] and SCIF (Split Covariance Intersection Filter) [61].

The EKF contains a linearization step, in which non-linear motion and sensors models get linearized around the current estimation, in order to deal with non-linear models. It has been used in many outdoor SLAM applications, and has enabled vehicles equipped indifferently with a Lidar sensor [58] or a monocular camera [59] to geolocalize themselves while building a map of the environment. Nevertheless, while it has been demonstrated that the EKF is optimal as long as the linearization step is operated around the true value, this true value remains unknown in practice. Due to unknown correlations between the different measurements, the EKF tends to become over-confident in the long-run, especially with highly non-linear systems [60], and can lead to inconsistent estimations, i.e. estimations that do not contain the ground truth within the estimated uncertainty.

To deal with highly non-linear systems, the UKF was proposed as an improvement of the EKF [60]. It functions by using a number of different particles sampled around the current estimation, which enables to avoid the direct computation of Jacobian matrices during the linearization step. Although a high number of particles is generally needed to provide consistent estimations, [48] proposed to combine the use of UKF and EKF, in order to maintain real-time performances. Nevertheless, while the UKF helps to mitigate consistency

issues associated with highly non-linear systems, it does not completely address them, and it also tends to become over-confident with time.

The CIF (Covariance Intersection Filter) was introduced to avoid such over-confidence, by merging information conservatively, assuming dependency between all sensors measurements. Although this method enables to provide consistent estimations, such estimations are generally highly suboptimal, as many measurements can generally be assumed independent [62]. Therefore, the SCIF, which consists in combining the use of an EKF and a CIF to respectively process independent and dependent measurements, was introduced. In [61], a vehicle equipped with a Lidar sensor was accurately geo-positioned, making use of an SCIF. Nevertheless, although the SCIF appears as a good solution to avoid over-confidence without being highly suboptimal, it has been rarely used within vehicular applications. An explanation for this could be that identifying the parameters of the SCIF, which define the part of the measurements that are to be assumed independent or dependent, remains a complex and time-consuming task.

The Particle Filter (PF) The PF, oppositely to the KF and its derivatives, does not assume anything regarding the linearity of the system and the different types of noises. This enables to handle multimodal distributions, in which sensors measurements follow various non-linear models [57]. To achieve this, a number of different particles are randomly generated, and represent different hypotheses for the filter state. At each step, the PF associates a likelihood score to each particle. Subsequently, the most likely particles are maintained, while the least likely ones are eliminated. As time grows, it has been observed that one particle tends to dominate all other particles with its likelihood score. This makes the PF unable to consider multiple hypothesis, and thus endangers its capacity to model multimodal and non-linear systems. To address this issue, a resampling step is generally processed periodically, in order to generate new particles.

To provide accurate estimations of the agent trajectory and landmarks positions, the PF

would typically require a large number of particles, and induce a high computational cost. To deal with this, [63, 64] proposed to factor the SLAM problem into two subproblems, by estimating the agent trajectory and the landmarks positions using respectively a PF and multiple independent EKF. In [65], this strategy was evaluated during an outdoor experiment and achieved a better positioning accuracy than when using an EKF. Nonetheless, the factorization of the SLAM problem inevitably induces some approximations, which tend to make the process over-confident, and lead to inconsistent estimations in the long-run.

Optimization-based SLAM

While filter-based SLAM use only the last measurements provided by embedded sensors, methods based on optimization take advantage from all the measurements retrieved during a given time-window. To do so, they estimate the agent trajectory and the landmarks positions through the minimization of a cost function derived from sensors measurements. Overall, optimization-based methods have been used to solve both full SLAM, when the time-window contains all retrieved measurements, and online SLAM, when the time-window contains only a subset of the measurements. These methods generally follow two main strategies: bundle adjustment and graph optimization, which are discussed below.

Bundle Adjustment Bundle adjustment is a vision technique that makes use of camera detections to build a map of landmarks. It functions by using many such detections to establish constraints between the camera trajectory and the positions of detected landmarks. Assuming that camera detections are affected by zero-mean noise, the associated constraints are used to form a non-linear least-squares problem that is solved through global optimization. Throughout this process, accuracy information can be computed in the form of covariances, at the condition that noises follow a Gaussian behavior [66]. Obviously, the global optimization can be computationally extensive, and bundle adjustment has been mostly dedicated to solve full SLAM.

Nevertheless, some attempts have been made to perform online SLAM by limiting the number of variables used in the optimization. In [67], an iterative process is proposed, in which the optimization only updates variables associated with new measurements. In [68] and [69], a local optimization scheme is introduced, where the optimization considers only a sliding-window of the last camera keyframes. All of these methods enabled to reduce the computational cost of bundle adjustment, with [69] even achieving real-time performances, and paved the way for the development of subsequent optimization methods.

However, while bundle adjustment enables to take advantage from a large number of camera detections, it does not easily integrate other types of measurements. This remains problematic towards its use for vehicular applications, since intelligent vehicles are expected to be equipped with a variety of proprioceptive and exteroceptive sensors. To deal with this, some works proposed to modify the optimization process, in order to include additional information, such as GNSS and inertial measurements [70]. Notably, their formulation ends up looking very similar to the formulation of graph optimization, which is introduced hereafter.

Graph Optimization Graph optimization was proposed as a generalization of the probabilistic formulation of SLAM [50]. It is an optimization method that can integrate any type of measurements during a given time-window. It does so by representing the SLAM problem as a graph, in which the agent trajectory and landmarks positions are nodes, and measurements are links that constrain the nodes. By assuming all measurements to be affected by zero-mean, Gaussian noises, and making use of an optimization process, it is able to estimate the nodes states, i.e. the agent trajectory and landmarks positions.

Over the years, multiple SLAM methods based on graph optimization have been proposed. They generally aim to decrease the computational complexity, in order to achieve real-time requirements. In [71], the graph model is represented as a tree structure, and optimized using a gradient descent. In [72], the structure of the graph is taken into account

to speed up the optimization. In [5], an incremental version of graph optimization, which is able to maintain scalable computations, is proposed. In [73], graph optimization is dedicated to the localization of the vehicle, and is only applied on a sliding-window of the last measurements.

Overall, graph optimization is a probabilistic method that can take advantage from different types of measurements, making it more flexible than bundle adjustment. Furthermore, it is also an optimization method that can profit from all measurements retrieved during a given time-window, leading to a better accuracy than filter-based methods. In fact, [7] compared the use of graph optimization and the EKF, and showed that graph optimization generally provides better estimations. In [8], a comparison of the use of graph optimization and the PF also shows that graph optimization tends to provide more accurate results.

However, during graph optimization, in order to keep computations at a reasonable level, the noises affecting measurements are assumed to follow zero-mean, Gaussian distributions. In practice, noises may diverge from this hypothesis, leading graph optimization to become over-confident and provide inconsistent estimations in the long-run. Thus, although graph optimization appears as the best SLAM method, it remains insufficient, as other SLAM techniques, to guarantee long-term, accurate localization of a given vehicle.

3.1.3 Localization using Pre-Built Maps

To maintain accurate positioning in the long-run, some methods proposed to make use of maps of the environment. These maps are built in advance and used as prior knowledge within a SLAM process to geo-position vehicles. To do so, their content is matched with detections obtained from perception measurements issued by the vehicles. These maps can also be shared, in order to communicate various road information among vehicles and the road infrastructure. To build them, some methods aimed to take advantage from data that has already been gathered, including cadastral plans [74] and street views [75], while other techniques instead made use of dedicated fleets of vehicles to retrieve new data [76, 41].

All of these approaches can be classified into three categories, which are introduced below, depending on the type of maps that they are based on: road maps, street view maps and landmarks maps.

Road Maps The first range of methods aimed to take advantage from existing road maps, such as Google Maps or OpenStreetMap, to geolocalize vehicles. In these methods, the local trajectory of the vehicle is estimated, making use of measurements retrieved from embedded sensors. Then, this local trajectory is matched with the road map, assuming that the vehicle remains on the road at all times, in order to compute the global trajectory of the vehicle. In [77], visual odometry based on camera images provides the local trajectory, while a PF computes the global one. In [78], semantic classification based on Lidar scans identifies the shape of the road on which the vehicle is driving, enabling an optimization process to estimate the global trajectory. Overall, although these methods take advantage of widely available road maps, they remain sensitive to the quality of these maps. Furthermore, they can be sensitive to repetitive road patterns, and lose accuracy over long, straight roads. In fact, [78] achieves meter-level accuracy of localization, which remains below the targeted accuracy for intelligent vehicles, while [77] tends to provide less accurate results.

Street View Maps Other methods aimed to take advantage from street view databases, such as Google Street View. To geo-position a vehicle, they compare images retrieved by embedded cameras with images present in the given database. In [75], camera images are matched and localized using local bundle adjustment. In [79], continuous localization of camera images is achieved using a Bayesian tracking algorithm. In [80], a mobile robot is positioned by combining the use of camera images and inertial measurements through an optimization process. While the idea of using available street view images appears interesting, the associated methods remain complex and computationally demanding. Furthermore, these methods require rich visual environments, and are typically sensitive to environments that contain repetitive visual structures, such as suburbs or rural areas. In fact,

they typically provide positioning with errors that can reach several meters.

Landmarks Maps Another strategy consists in making use of a map of landmarks, which contains the geo-positions and accuracy of detected landmarks. Potentially, the landmarks can take any form, including low-level landmarks (e.g. dense pointclouds [81]), and high-level landmarks (e.g road markings [41]). The map is generally built through a SLAM process, making use of vehicles equipped with high-end sensors (e.g. high-quality Lidar sensors, Radar sensors, cameras). Nonetheless, some works aimed to use already available data, such as [74], which built a map of buildings from cadastral plans. Anyway, after the map is established, it can be used as prior knowledge for geolocalizing vehicles. To do so, landmarks are detected within perception measurements issued by a given vehicle, and matched with the landmarks present in the map. Subsequently, a process similar to SLAM, using both the sensors measurements and the map information, is able to estimate the vehicle trajectory.

The accuracy of such localization is subject to two different challenges. First, the landmarks must be reliably detected and matched, which has motivated the use in many applications of high-level landmarks, that are generally easily identifiable, and that can also be used for navigation purposes. Second, the map must have a high-precision, i.e. contain accurate landmarks geo-positions, as the map accuracy is correlated with the resulting localization accuracy. For these reasons, [76] and [34] used vehicles equipped with high-end sensors to build accurate maps containing respectively pole-shaped landmarks, and traffic signs and road markings. Making use of these maps, and applying respectively filter-based and optimization-based methods, [76] and [34] could geo-position vehicles equipped with standard sensors with a decimeter-level accuracy.

These works illustrated the potential of this approach to geo-position production vehicles with a high accuracy. In fact, major car manufacturers are now considering the use of maps of landmarks as an essential solution to achieve accurate geolocalization of the vehicles.

Nevertheless, while this approach appears promising, it requires the prior establishment of a map of landmarks that is both accurate, to provide precise position information, and up-to-date, to allow reliable detection and matching of the landmarks.

3.2 Map Construction

Making use of a map of landmarks appears as a promising solution to geo-position production vehicles with accuracy. Such a map should contain both the landmarks geo-positions and the associated accuracy, so as to allow for the proper merging of the information contained within the map, with the information provided by measurements issued by the sensors embedded in vehicles [5]. Furthermore, the map should be constantly updated, through the addition or the removal of new or outdated landmarks, in order to always remain up-to-date.

To build such a map, multiple methods have been proposed within the robotics field, and applied for vehicular applications. They generally make use of a SLAM process, in order to make localization and mapping profit from one another. While many robotics applications are restricted to short-term scenarios, vehicular applications require to build maps that can span over large areas, while remaining up-to-date. In order to address resulting scalability issues, collaborative strategies that make use of multiple vehicles have been proposed. In this section, we discuss these mapping methods, and introduce the different approaches. In subsection 3.2.1, we introduce collaborative strategies, and in subsection 3.2.2, we follow more specifically with crowdsourcing approaches.

3.2.1 Collaborative Mapping

To build maps that can span over large areas, various robotics applications have proposed to take advantage of multiple agents through collaborative SLAM methods. In [82], mobile robots equipped with monocular cameras are either grouped, to improve SLAM performances through overlapping views, or split, to explore different areas. In [83], a data-driven descriptor is introduced to extract meaningful features and perform collaborative SLAM

with Lidar-equipped vehicles.

To aggregate measurements from multiple agents and perform collaborative SLAM, existing methods can be classified into two main categories: centralized strategies and decentralized approaches. Within a centralized strategy [84, 85], agents first upload their measurements towards centralized servers. The servers then merge the observations received from multiple agents, making use of a SLAM process, in order to build and update the map. Such strategy appears straightforward, as all measurements are used at once during a single SLAM process to build a single map. Nevertheless, it does require to have available computational resources on centralized servers, as well as reliable communications between the agents and the servers [86]. To deal with communication issues, decentralized approaches, in which agents share measurements with each other, and perform SLAM using their own embedded processing units, have been proposed [87]. Such decentralized approaches remain generally complex to implement, due to the need to share measurements among participating agents without double-counting information when building the map. To deal with this, the typical strategy consists in avoiding to merge already fused information between the agents [87]. Furthermore, decentralized approaches can impose strong requirements on the limited computational resources available on agents. To address such issues, [88] proposed to limit inter-agent communications to one agent at a time. Nonetheless, while such strategies allow to decrease the computational burden and avoid double-counting information, they generally lead to suboptimal estimations.

To build high-precision maps of the road environment, major automotive manufacturers have applied collaborative SLAM methods using dedicated fleets of vehicles equipped with high-end sensors [31]. Due to the limited computational resources available on vehicles, as well as the targeted accuracy of the built maps, they generally apply a centralized strategy where massive cloud servers merge observations retrieved by continuously-driving vehicles. However, due to the high operational cost, they can only operate a limited number of vehicles, and have been unable to register more than the main highways, nor to provide

frequent updates to the maps [89].

3.2.2 Crowdsourced Mapping

Crowdsourced mapping defines a sub-category of collaborative mapping which emerged to take advantage from the increasing availability of smartphones (for indoor applications) and production vehicles (for outdoor applications) equipped with standard sensors. The crowdsourcing approach consists in making use of a vast amount of redundant measurements, in order to compensate for the use of standard sensors, and to build a map that is highly-accurate and always up-to-date. This strategy requires to aggregate measurements from many different agents, and is generally implemented through centralized approaches.

Initial works applied crowdsourced mapping for indoor applications, making use of inertial sensors embedded in smartphones. [90] applied a clustering algorithm on measured trajectories, while [91] solved the SLAM problem through graph optimization. These works showed the potential of crowdsourced mapping to build maps of indoor environments, making use of redundant measurements retrieved from standard sensors.

Subsequent works aimed at building maps of the road environment, making use of production vehicles equipped with standard sensors. These works are summarized in Table 3.1. In [92], a map is built using only camera observations. This approach makes use of a global optimization process to correct the map in the case of a loop closure, and thus appears unscalable to a large amount of vehicles. Furthermore, the built map consists of low-level features associated with keyframes, which could complicate the data association step, and potentially lead to additional computations and unstability during the estimation process. In [93], [94] and [95], maps of high-level landmarks are built using camera observations to retrieve perception measurements. In all of these works, the mapping process is split in two steps. First, each individual vehicle builds a first estimation of the map, by making use of a local process. Then, centralized servers merge the different estimations through a global process. In fact, [93] and [94] use a global optimization for this task, which can potentially

	Perception sensor	Map accuracy	Map content	Number of passages	Mapping method
[93]	Monocular camera	Decimeter-level	Traffic signs, road lanes	25	Bundle adjustment
[92]	Monocular camera	Several meters	Keyframes, Point features	2	Bundle adjustment, Graph optimization
[96]	Lidar sensor, Monocular camera	Several decimeters	Street lights, Road lanes	11	Graph optimization RLS filter
[94]	Monocular camera	Few decimeters	Road boundaries, road markings	10	Graph optimization
[95]	Monocular camera	Unknown	Road markings	Unknown	Graph optimization

Table 3.1: Crowdsourced mapping methods developed for vehicular applications.

induce heavy computations, and be unscalable for a large number of vehicles. Notably, [93] and [94] respectively used 25 and 10 vehicle passages during their experiments, while [95] did not specify the number of used vehicle passages.

Meanwhile, an iterative method was proposed in [96], making use of vehicles equipped with both a Lidar sensor and a monocular camera. The built map consists of high-level landmarks, and is gradually updated as vehicles upload new measurements. This work showed the potential of crowdsourced mapping, by being able to build a map of landmarks while remaining computationally scalable. Nevertheless, the process remains suboptimal, as landmarks geo-positions and their accuracy are estimated using both graph optimization and an RLS (Recursive Least-Squares) filter. Furthermore, during their experiments, [96] used a limited amount of vehicles passages, 11 in fact. Thus, the real accuracy and scalability potential that can be achieved with crowdsourced mapping remains to be assessed in a realistic, long-term scenario.

3.3 Conclusion

Intelligent vehicles will need to be geolocalized with errors up to a few decimeters [1], in order to perform driving functions autonomously, and share meaningful positions and trajectories with each other. To achieve such accuracy, GNSS receivers are not sufficient, as they

suffer from various atmospheric and multi-path effects, which lead to severe unavailability and inaccuracy issues within dense environments. The addition of other types of sensors, including proprioceptive and perception sensors, helps to increase the localization accuracy, but does not allow to address inherent GNSS limitations in the long-run.

Instead, another strategy consists in positioning the vehicles relatively to a given map of the environment. A variety of SLAM methods have been proposed in this direction, and have allowed a given vehicle to build a map of its environment while localizing itself within this map. Among these methods, graph optimization has appeared as the technique achieving best localization and mapping performances [7, 8]. Nevertheless, such methods tend to become over-confident in the long-run, leading to inconsistent maps and thus inaccurate localization of the vehicles. Therefore, instead of making each vehicle build its own map, other works proposed to use maps of the environment that are established in advance and common to all vehicles. In fact, making use of a SLAM process along with a pre-built map of accurately positioned landmarks has enabled to geolocalize standard vehicles with the targeted accuracy [76, 34].

Nevertheless, this strategy requires the prior establishment of an accurate map of geo-positioned landmarks, which must span over large areas while always remaining up-to-date. In order to build such a map, major actors in the field have been applying collaborative SLAM methods with dedicated fleets of vehicles equipped with high-end sensors. However, they are now facing strong logistical limitations that prevent them from building maps that cover the entire road network, nor to maintain the existing maps up-to-date [89]. Following the generalization of production vehicles equipped with standard sensors, crowdsourcing methods have appeared as a cost-effective approach to build and update a high-precision map. Nonetheless, while initial works could show promising results, the real potential of crowdsourced mapping remains to be asserted in realistic, long-term scenarios.

CHAPTER 4

CROWDSOURCED MAPPING USING GRAPH OPTIMIZATION

Crowdsourced mapping represents a cost-effective solution to build a large-scale and up-to-date map of geo-positioned landmarks. It consists in retrieving measurements from production vehicles equipped with standard sensors, and building the map in a collaborative manner. A complete crowdsourcing system involves several different steps, in order to collect the raw measurements on the vehicles, pre-process and filter them, and build and update the map of landmarks.

In this thesis, we focus on the mapping part of such a system, which aims to build and update the map from given pre-processed measurements. In fact, wherever we mention the term: "crowdsourced mapping" in the following, we refer to the mapping part of the complete crowdsourcing system. First, in order to verify the effectiveness of the collaborative approach, we introduce a crowdsourced mapping pipeline based on a simple triangulation optimization. Then, we propose an improved solution based on graph methods such as those used in SLAM, which have shown better results than other filtering-based methods [7, 8].

To be of interest in real situations, crowdsourced mapping must be able to build an accurate map of geolocalized landmarks, while maintaining scalability, i.e. reasonable computational costs, such that the map can be frequently updated and span over large areas. Generally, the scalability of map-building methods and the accuracy of the resulting map are contradictory objectives. Therefore, in our solution based on graph optimization, we propose three different methods for updating the map, which correspond to different trade-offs between scalability and map accuracy. We compare and evaluate their respective performances, and conclude on the most efficient approach to be applied in real conditions.

The structure of this chapter is as follows. In section 4.1, we begin with a general overview of the crowdsourcing system and clarify the scope of our work. In section 4.2,

we introduce our solution based on triangulation optimization. In section 4.3, we recall the formulation of graph optimization, as a preamble before introducing our proposition. In section 4.4, we detail our solution based on graph optimization, and introduce the three different methods used to update the map.

4.1 Crowdsourced Mapping

In the following, we describe the main steps contained in a complete crowdsourcing system. We clarify our focus on the mapping part, and detail its content from an applicative standpoint.

4.1.1 Overview

The typical crowdsourced mapping system contains several different modules, as illustrated in Figure 4.1. On each vehicle, an acquisition module registers raw measurements from embedded sensors. A pre-processing module transforms these raw measurements into usable sensors observations, making use of several operations, such as time interpolation of the different measurements, detection of the landmarks within perception measurements, and data association to match the detected landmarks with landmarks in the map. Making use of these sensors observations, as well as the map, a localization module is able to geo-position the vehicle within the infrastructure. Obviously, these three operations have to be achieved directly on the vehicle and in real-time.

In the meanwhile, on centralized servers, the mapping module receives measurements from multiple vehicles, and uses them to build and update the map. Although this process does not need to be run in real-time, it does need to be scalable, as it must deal with a large quantity of crowdsourced data, in order to build an accurate and up-to-date map that covers large portions of the road network.

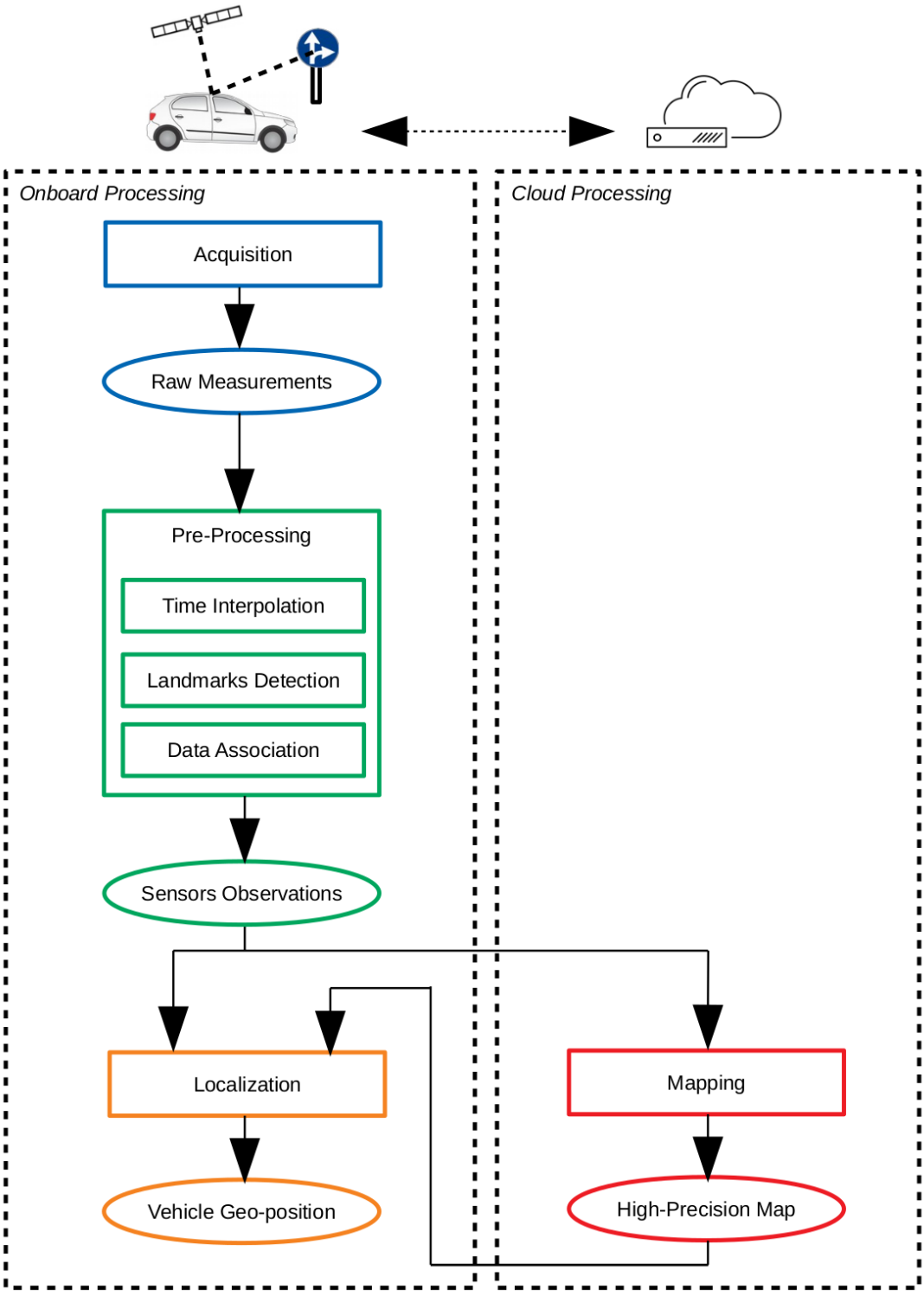

Figure 4.1: Overview of crowdsourced mapping, with modules split between onboard processing and cloud processing.

4.1.2 Mapping Strategies

To implement the mapping module, two different architectures can be used. In a first approach, each vehicle estimates the landmarks geo-positions by itself, making use of its own sensors observations. Then, the role of the servers consists in merging the different geo-positions estimations received from multiple vehicles. This solution requires few communication bandwidth between the vehicles and the servers, but it can impose high computations on the processing units embedded in the vehicles.

In a second approach, each vehicle registers sensors observations, and directly uploads them to centralized servers. The servers then use the observations retrieved from multiple vehicles to estimate the landmarks geo-positions. This solution enables to deport most calculations on the servers, but it can lead to higher requirements of communication bandwidth. Note that, as the mapping module does not need to be run in real-time, this bandwidth issue is not critical. At the moment, we consider this second approach to be more in line with existing limits on intelligent vehicles. For this reason, we chose this architecture to illustrate the overview of crowdsourced mapping in Figure 4.1. Nevertheless, in this thesis, we do not make any assumption on which strategy is applied, and we propose a solution that is independent from the location on which calculations are processed.

In this work, we focus on the development of the mapping module, which builds and updates a map of geolocalized landmarks, making use of pre-processed sensors observations retrieved from multiple vehicles.

4.2 Triangulation Optimization for Crowdsourced Mapping

In order to assess the potential of crowdsourced mapping, we start by building a simple and intuitive solution. We propose a crowdsourced mapping approach based on a triangulation optimization, as illustrated in Figure 4.2, with the different operations depicted as colored rectangles and the data shown as white ovals.

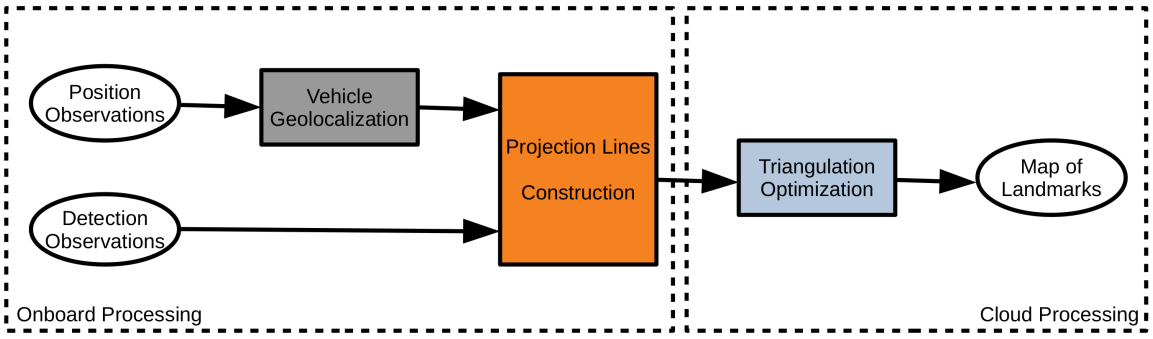

Figure 4.2: Triangulation-based Approach for Crowdsourced Mapping.

This triangulation-based approach functions as follows. Consider any vehicle that follows a given trajectory, detects surrounding landmarks, and registers a given set of measurements. In order to build the map of landmarks using these measurements, it is required that the vehicle can provide at least two types of observations: position observations (such as GNSS measurements), and detection observations (such as landmarks detections within camera images). Therefore, in this approach, we consider any vehicle that is able to provide these two types of observations. Whenever a position observation and a detection observation are obtained, a projection line is established as passing through:

- The center of the perception sensor associated with the detection observation.
- The geo-position of the detected landmark.

Among the perception sensors that can be embedded in driving vehicles, we either find sensors that provide only bearing information (such as monocular cameras) or sensors that give both range and bearing information (such as Lidar or Radar sensors). While bearing information does not provide direct information about the landmarks geo-positions, it provides sufficient information to build the associated projection lines. Therefore, projection lines can be established considering any type of perception sensors embedded in the vehicles. The formulation of projection lines typically depends on the type of perception sensors that are used, and is not detailed here in order to keep a general discussion of the approach.

While driving, each vehicle registers multiple projection lines related to various detected landmarks. When it reaches the end of its trajectory, it uploads all established projection lines towards centralized servers. Thus, the servers receive multiple projection lines from many different vehicles, and use them to estimate the geo-positions of detected landmarks. The position of each landmark L_i is estimated independently from the positions of other landmarks, by minimizing the following cost function:

$$\hat{L}_i = \underset{L_i}{\operatorname{argmin}} \sum_{k=0}^{K} dist(L_i, Z_k(L_i)) \tag{4.1}$$

where $\hat{L}_i$ is the estimated geo-position of L_i, and $Z_0(L_i), ..., Z_K(L_i)$ are all the projection lines established for L_i. Meanwhile, $dist(L, Z)$ correspond to the orthogonal distance between the geo-position L and the line Z. To solve this equation, we use a simple least-squares optimization as formulated in [97].

Overall, this triangulation-based approach represents an intuitive solution to the crowd-sourced mapping problem, and was used to assess the potential of crowdsourced mapping during early field-tests (see section 5.1). Nevertheless, some limitations associated with this approach could be identified as follows:

- This method does not use all of the available information, such as displacement observations (provided for instance by motor encoders), which could have an impact on the map accuracy.
- The estimated accuracy of landmarks geo-positions in the map is not provided, although this information can be necessary to allow efficient use of the map [5].
- At every step, all projection lines established since the beginning are used, making this method not computationally scalable to a high number of vehicles.

4.3 Graph-based Approaches

In order to build an accurate map of landmarks, and address issues associated with the triangulation optimization, we aimed to develop a more elaborate solution for crowdsourced mapping, making use of methods such as those used in recent SLAM techniques. Indeed, SLAM methods have been widely used within the robotics community to merge measurements from a variety of sensors, taking into account their respective accuracy. Furthermore, SLAM methods typically provide a Bayesian formulation of the mapping problem, and enable to compute an estimation for the accuracy of landmarks geo-positions within the map. Among SLAM methods, graph optimization has shown better performances than other filtering-based techniques [7, 8]. In this work, we propose a crowdsourced mapping solution based on graph optimization, along with three different methods for updating the map. This enables to compare different crowdsourcing approaches, and identify the technique that achieves the best performances in terms of map quality and computational scalability.

In the following, we formulate the SLAM problem, and derive its graph-based solution, in order to lay theoretical ground for the subsequent presentation of our crowdsourced mapping pipeline.

4.3.1 SLAM Formulation

Before formulating the SLAM problem, we define some basic variables, without losing generality of the discussion. Consider any vehicle that moves from discrete time instants $k = 0$ to $k = K$ along a given trajectory $X_{0:K} = \{X_0, ..., X_K\}$, where X_k contains both the position and orientation of the vehicle at instant k. While driving, the vehicle detects a given number N of landmarks whose geo-positions define a map $M = \{L_1, ..., L_N\}$. In the remainder of this document, the notation L_i will indifferently be used to design the landmark of index i, and its geo-position. While driving, the vehicle also acquires a given set of observations $Z_{0:K} = \{Z_0, ..., Z_K\}$ from its embedded sensors, where Z_k contains all

the measurements issued at instant k. Considering the existing sensors that can be embedded in vehicles and used for localization and mapping purposes, any variable Z_k may contain three different types of measurements:

- Displacement observations, that inform about the vehicle motion between two successive instants. These measurements, noted U_k, are provided by proprioceptive sensors through a motion model f:

$$X_{1:K} = f(X_{0:K-1}, U_{1:K}) + \epsilon_{u,1:K} \tag{4.2}$$

  where $\epsilon_{u,1:K}$ is the prediction noise.

- Position observations, that inform about the position of the vehicle at a particular instant. These measurements, noted Y_k, are provided by exteroceptive sensors through a sensor model g:

$$Y_{0:K} = g(X_{0:K}) + \epsilon_{y,0:K} \tag{4.3}$$

  where $\epsilon_{y,0:K}$ is the sensor noise. Position observations are not defined in many SLAM works, as they mostly originate from GNSS receivers, which are not used within indoor robotics applications. However, such sensors are commonly used in outdoor and vehicular applications, which motivates their inclusion in our formulation of the SLAM problem.

- Detection observations, that inform about the detection of one or more landmarks by the vehicle. These measurements, noted D_k, are provided by exteroceptive sensors through a sensor model h:

$$D_{0:K} = h(X_{0:K}, M) + \epsilon_{d,0:K} \tag{4.4}$$

  where $\epsilon_{d,0:K}$ is the sensor noise.

The SLAM problem aims to jointly estimate the vehicle trajectory and landmarks geo-positions, considering all observations provided by the sensors. To deal with the noise present in the measurements, this problem is formulated in a probabilistic manner, in which estimations of the vehicle trajectory $\widehat{X}_{0:K}$ and landmarks geo-positions $\widehat{M}$ correspond to the maximum of the posterior PDF (Probability Density Function) $\mathcal{P}$ given all observations [4]:

$$\widehat{X}_{0:K}, \widehat{M} = \underset{X_{0:K}, M}{\arg\max}\, \mathcal{P} = \underset{X_{0:K}, M}{\arg\max}\, P(X_{0:K}, M | Z_{0:K}) \tag{4.5}$$

where $P(X|Y)$ corresponds to the PDF of X given Y.

The posterior PDF $\mathcal{P}$ contains all inter-relations between $X_{0:K}$, M and $Z_{0:K}$, and is generally too complex to estimate. In order to simplify $\mathcal{P}$ and solve Equation 4.5, we make a number of justified assumptions that are discussed in the following. Naturally, these assumptions concern the known variables, i.e. observations $Z_{0:K}$, and not the unknown variables, i.e. states $X_{0:K}$ and M.

The posterior PDF $\mathcal{P}$ expresses the probability of unknown states $X_{0:K}$ and M depending on the known measurements $Z_{0:K}$. This complicates the simplification of $\mathcal{P}$ through assumptions on $Z_{0:K}$, and motivates the use of the Bayes theorem as follows, in order to inverse the role of known measurements $Z_{0:K}$ and unknown states $X_{0:K}$ and M within $\mathcal{P}$:

$$\mathcal{P} = P(X_{0:K}, M | Z_{0:K}) = P(Z_{0:K} | X_{0:K}, M) \frac{P(X_{0:K}, M)}{P(Z_{0:K})} \tag{4.6}$$

$P(X_{0:K}, M)$ and $P(Z_{0:K})$ can be considered constant for a given application, i.e. given vehicle trajectory and landmarks configuration, which gives:

$$\mathcal{P} \propto P(Z_{0:K} | X_{0:K}, M) \tag{4.7}$$

We re-write Equation 4.7 by splitting $Z_{0:K}$ into the associated displacement, position

and detection observations $U_{1:K}$, $Y_{0:K}$ and $D_{0:K}$:

$$\mathcal{P} \propto P(U_{1:K}, Y_{0:K}, D_{0:K} | X_{0:K}, M) \tag{4.8}$$

Displacement and position observations $U_{1:K}$ and $Y_{0:K}$, due to their inherent nature, are clearly independent from landmarks geo-positions M. Furthermore, displacement, position and detection observations $U_{1:K}$, $Y_{0:K}$ and $D_{0:K}$ are respectively processed by different sensors, which generally do not interfere with each other. Thus, we can reasonably consider that these different types of measurements are conditionally independent, i.e. that:

- Observations $U_{1:K}$ do not directly depend on observations $Y_{0:K}$ or $D_{0:K}$, but only on vehicle states $X_{0:K}$.
- Observations $Y_{0:K}$ do not directly depend on observations $U_{1:K}$ or $D_{0:K}$, but only on vehicle states $X_{0:K}$.
- Observations $D_{0:K}$ do not directly depend on observations $U_{1:K}$ or $Y_{0:K}$, but only on vehicle states $X_{0:K}$ and landmarks geo-positions M.

Following, we can write:

$$\mathcal{P} \propto P(U_{1:K} | X_{0:K})\ P(Y_{0:K} | X_{0:K})\ P(D_{0:K} | X_{0:K}, M) \tag{4.9}$$

By taking a look at the formulation of sensors models g and h (see respectively Equation 4.3 and Equation 4.4), it can be seen that they enable to directly derive the probabilities $P(Y_{0:K}|X_{0:K})$ and $P(D_{0:K}|X_{0:K}, M)$. Oppositely, the motion model f (see Equation 4.2) does not allow to directly derive $P(U_{1:K}|X_{0:K})$, but instead $P(X_{0:K}|U_{1:K})$. This motivates a second use of the Bayes theorem as follows:

$$P(U_{1:K} | X_{0:K}) = P(X_{0:K} | U_{1:K}) \frac{P(U_{1:K})}{P(X_{0:K})} \tag{4.10}$$

$P(U_{1:K})$ and $P(X_{0:K})$ can be considered constant for a given application, which gives:

$$P(U_{1:K}|X_{0:K}) \propto P(X_{0:K}|U_{1:K}) \tag{4.11}$$

And:

$$\mathcal{P} \propto P(X_{0:K}|U_{1:K})\ P(Y_{0:K}|X_{0:K})\ P(D_{0:K}|X_{0:K}, M) \tag{4.12}$$

Until now, we made use of basic assumptions that can be considered valid in the general case. Unfortunately, this is not sufficient to make the SLAM problem solvable, as the probabilities $P(X_{0:K}|U_{1:K})$, $P(Y_{0:K}|X_{0:K})$ and $P(D_{0:K}|X_{0:K}, M)$ remain typically too complex to estimate.

4.3.2 Graph Model

To further simplify the posterior PDF $\mathcal{P}$ and solve the SLAM problem, we make use of two subsequent assumptions that are at the basis of the graph formulation of SLAM [50]. These assumptions are discussed in the following, and represent both the interest and challenges of the graph-based approach. Indeed, while they enable to simplify the SLAM problem into a solvable form, they induce hypotheses on the measurements noises that can be challenged in real conditions. In fact, a substantial part of our experiments is dedicated to the evaluation of the impact that such hypotheses can have on the performances of crowdsourced mapping.

The first assumption consists in assuming that all observations are conditionally independent, i.e. that:

- Any observation U_k does not depend on another observation $U_{k' \neq k}$.
- Any observation Y_k does not depend on another observation $Y_{k' \neq k}$.
- Any observation D_k does not depend on another observation $D_{k' \neq k}$.

This translates into considering that the measurements are not affected by auto-correlated noises, although this is generally the case in real conditions, especially for position observations Y_k. Indeed, such observations are generally obtained from GNSS receivers, whose measurements are known to be affected by auto-correlated noises, mainly due to common delaying errors of GNSS signals when travelling the ionosphere and troposphere, as well as to multi-path effects. In the case of strong auto-correlated noises, which could severely affect the resulting map quality, this could potentially be dealt with by applying covariance inflation [98], in order to avoid over-confidence of the map.

Overall, this assumption leads to the basic principle of the Markov model, in which all past information is considered to be contained within X_{k-1}. This transforms the motion and sensors models f, g and h as follows:

$$X_k = f(X_{k-1}, U_k) + \epsilon_{u,k} \tag{4.13}$$

$$Y_k = g(X_k) + \epsilon_{y,k} \tag{4.14}$$

$$D_k = h(X_k, M) + \epsilon_{d,k} \tag{4.15}$$

And it transforms Equation 4.12 into:

$$P \propto P(X_0) \prod_{k=1}^{K} P(X_k|X_{k-1}, U_k) \prod_{k=0}^{K} P(Y_k|X_k) \prod_{k=0}^{K} P(D_k|X_k, M) \tag{4.16}$$

$P(X_0)$ can be considered constant for a given application:

$$P \propto \prod_{k=1}^{K} P(X_k|X_{k-1}, U_k) \prod_{k=0}^{K} P(Y_k|X_k) \prod_{k=0}^{K} P(D_k|X_k, M) \tag{4.17}$$

This formulation can be viewed as a factor graph [99], in which the vehicle states $X_{0:K}$ and landmarks geo-positions M are nodes, and the observations $U_{1:K}$, $Y_{0:K}$ and $D_{0:K}$ are

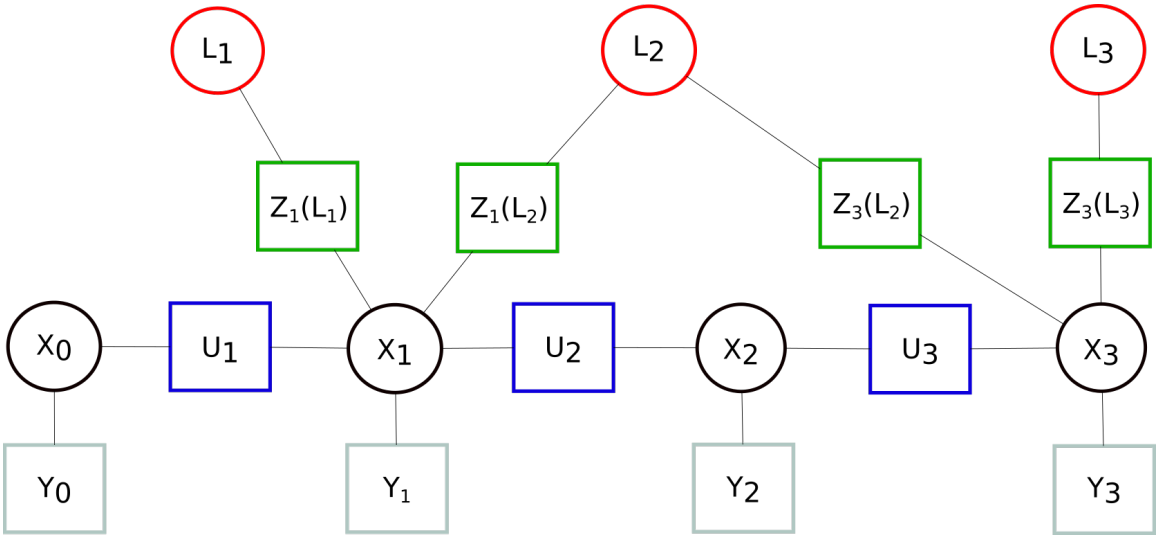

Figure 4.3: The SLAM problem modeled as a graph.

edges which constrain the nodes. The SLAM problem, modeled as a graph, comes down to estimating the unknown nodes states which best satisfy the known constraints. Figure 4.3 depicts a sample graph in the case of a vehicle that detects landmarks L_1 and L_2 at time $k = 1$, and L_2 and L_3 at time $k = 3$. The nodes are represented as circles, and the edges as squares.

The second assumption consists in assuming that all observations are affected by zero-mean, Gaussian noises:

- $\epsilon_{u,k}$ is modeled as being sampled from a zero-mean Gaussian with covariance $\Omega_{u,k}^{-1}$ and information $\Omega_{u,k}$.
- $\epsilon_{y,k}$ is modeled as being sampled from a zero-mean Gaussian with covariance $\Omega_{y,k}^{-1}$ and information $\Omega_{y,k}$.
- $\epsilon_{d,k}$ is modeled as being sampled from a zero-mean Gaussian with covariance $\Omega_{d,k}^{-1}$ and information $\Omega_{d,k}$.

Gaussian distributions are widely used in many fields to model random processes, and it is fairly reasonable to consider that noises in SLAM can be modeled as such. However,

making use of zero-mean distributions implies that the measurements are not affected by bias, although this is generally the case in real conditions. In typical vehicular applications, the main source of bias comes from the presence of calibration errors, which are constant errors that affect all sensors measurements. Especially, errors associated with the sensors orientations are known to have a more important effect on the perceived measurements than errors associated with the sensors positions. In the presence of severe bias, this could be dealt with by applying covariance inflation [98], or modifying the graph structure [100], in order to account for such bias.

Overall, making use of this assumption leads to:

$$P \propto \prod_{k=1}^{K} exp(-\frac{1}{2}\, e_{u,k}^{\top}\, \Omega_{u,k}\, e_{u,k}) \prod_{k=0}^{K} exp(-\frac{1}{2}\, e_{y,k}^{\top}\, \Omega_{y,k}\, e_{y,k}) \prod_{k=0}^{K} exp(-\frac{1}{2}\, e_{d,k}^{\top}\, \Omega_{d,k}\, e_{d,k}) \tag{4.18}$$

where $e_{u,k}$, $e_{y,k}$ and $e_{d,k}$ are the error functions respectively associated with displacement, position and detection observations U_k, Y_k and D_k:

$$e_{u,k} = X_k - f(X_{k-1}, U_k) \tag{4.19}$$

$$e_{y,k} = Y_k - g(X_k) \tag{4.20}$$

$$e_{d,k} = D_k - h(X_k, M) \tag{4.21}$$

Factorizing Equation 4.18, we can write:

$$P \propto exp(-\frac{1}{2}\, (\sum_{k=1}^{K} e_{u,k}^{\top}\, \Omega_{u,k}\, e_{u,k}\, + \sum_{k=0}^{K} e_{y,k}^{\top}\, \Omega_{y,k}\, e_{y,k}\, + \sum_{k=0}^{K} e_{d,k}^{\top}\, \Omega_{d,k}\, e_{d,k})) \tag{4.22}$$

We define the cost function F as:

$$F = \sum_{k=1}^{K} e_{u,k}^{\top} \, \Omega_{u,k} \, e_{u,k} + \sum_{k=0}^{K} e_{y,k}^{\top} \, \Omega_{y,k} \, e_{y,k} + \sum_{k=0}^{K} e_{d,k}^{\top} \, \Omega_{d,k} \, e_{d,k} \tag{4.23}$$

Which gives:

$$\widehat{X}_{0:K}, \widehat{M} = \underset{X_{0:K}, M}{\arg\max} \, exp(-\frac{1}{2} \, F) = \underset{X_{0:K}, M}{\arg\min} \; F \tag{4.24}$$

This equation must be solved, in order to estimate the vehicle states $\widehat{X}_{0:K}$ and landmarks geo-positions $\widehat{M}$. This step, called graph optimization in reference to the graph structure illustrated in Figure 4.3, can be processed using an iterative optimization method. While many such methods have been developed, they all follow the same principles. Beforehand, all graph nodes (vehicle states and landmarks geo-positions) are provided with initial values, which are then iteratively improved until convergence, making use of graph edges (sensors observations) as known constraints. In all these methods, there exists a strong requirement that the initial values of vehicle states and landmarks geo-positions are sufficiently close to their true values [4]. Otherwise, the estimation provided by the optimization may largely differ from the reality. It is a well-known issue of graph optimization, and has been addressed by several works [101, 102].

Among available optimization methods, we can find conventional non-linear least-squares solvers, such as the Gauss-Newton or Levenberg-Marquadt algorithms, as well as more recent techniques developped by the robotics community, such as g2o [72] and the Ceres Solver [103]. These more recent methods claim to provide better performances, both in terms of map quality and computational cost. On the other hand, conventional solvers present the advantage of being simple to implement and easy to manipulate, which helps to study performances and identify limitations of a given mapping approach. This makes conventional solvers appear more in line with the needs of our work.

The most basic solver is the Gauss-Newton algorithm, which makes use of a second-

order update to provide fast convergence, but remains sensible to the quality of the initial estimate and the quadratic fit of the objective function [104]. Meanwhile, the Levenberg-Marquadt algorithm makes use of heuristic parameters to improve its robustness to such conditions [104]. Nevertheless, heuristic parameters can lead to additional errors if badly selected, and the Gauss-Newton algorithm remains widely used in many SLAM applications.

The authors of [99] describe the Gauss-Newton algorithm for a typical robotics application, considering that GNSS signals are unavailable, and that only the vehicle states must be estimated. Thus, within their formulation:

- All constraints link exactly two nodes, as only displacement and detection observations are considered.
- The graph is under-constrained, and at least one node needs to be fixed.

In our case, the landmarks geo-positions definitely need to be estimated. Furthermore, an outdoor vehicle equipped with a GNSS receiver would provide position observations $Y_{0:K}$, which would constrain each one vehicle node, and result in the graph being fully-constrained. Thus, we provide a more general formulation of the Gauss-Newton algorithm in Algorithm 1. Note that, in Algorithm 1, the vehicle states $X_{0:K}$ and landmarks geo-positions M are stacked into a single vector X. Similarly, displacement, position and detection observations $U_{1:K}$, $Y_{0:K}$ and $D_{0:K}$ are stacked into $\mathcal{C}$.

After graph optimization, we obtain through $\hat{X}$ an estimation of the vehicle states $\widehat{X}_{0:K}$ and landmarks geo-positions $\widehat{M}$. Due to the preceding assumptions, such estimation is assumed to be affected by noises sampled from a zero-mean, Gaussian distribution, whose global covariance matrix Σ correspond to the inverse of the global information matrix Ω:

$$\Sigma = \Omega^{-1} \tag{4.25}$$

Algorithm 1 Gauss-Newton Algorithm for Graph Optimization

Require:
$\breve{X} = [\breve{X}_{0:K}, \breve{M}]$: Initial guess of nodes states
$\mathcal{C} = [c_1, c_2, ...]$: Observations of constraints
$\Omega_{\mathcal{C}} = [\Omega_{c_1}, \Omega_{c_2}, ...]$: Information matrices of constraints

Ensure:
$\hat{X} = [\hat{X}_{0:K}, \hat{M}]$: Estimation of nodes states
Ω: Global information matrix

while not converged **do**:
  $B \leftarrow 0$
  $H \leftarrow 0$
  ▷ Compute the Jacobians of all constraints
  for all $c \in \mathcal{C}$ **do**:
    ▷ Let $x_{I:J} = \{x_I, ..., x_J\}$ be the set of nodes constrained by c
    $x_{I:J} = Constrained(c)$
    for all $x_i \in x_{I:J}$ **do**:
      $J_{c,i} = \frac{\partial e_c}{\partial x_i}\Big|_{x_i = \breve{x}_i}$
    end for
  end for
  ▷ Compute contribution of all constraints to linear system
  for all $c \in \mathcal{C}$ **do**:
    $x_{I:J} = Constrained(c)$
    for all $x_i \in x_{I:J}$ **do**:
      $B_{[i]} \mathrel{+}= J_{c,i}^T \, \Omega_c \, e_c$
      for all $x_j \in x_{I:J}$ **do**:
        $H_{[i,j]} \mathrel{+}= J_{c,i}^T \, \Omega_c \, J_{c,j}$
      end for
    end for
  end for
  ▷ Solve linear system
  $\Delta X \leftarrow Solve(H \times \Delta X = -B)$
  ▷ Update current guess
  $\breve{X} \mathrel{+}= \Delta X$
end while
$\hat{X} \leftarrow \breve{X}$
$\Omega \leftarrow H$
return $[\hat{X}, \Omega]$

Σ contains the variances and covariances of all nodes in the graph:

$$\Sigma = \begin{pmatrix} \Sigma_X & \Sigma_{X,M} \\ \Sigma_{X,M} & \Sigma_M \end{pmatrix} \tag{4.26}$$

where:

- Σ_X is the covariance of vehicle states.
- Σ_M is the covariance of landmarks geo-positions.
- $\Sigma_{X,M}$ is the cross-covariance between vehicle states and landmarks geo-positions.

The estimated vehicles states and their covariance consist in the tuple $\widehat{X}_{0:K}, \Sigma_X$. Similarly, the estimated landmarks geo-positions and their covariance, which represent the map, consist in the tuple $\widehat{M}, \Sigma_M$.

4.4 Graph Optimization for Crowdsourced Mapping

Crowdsourced mapping consists in retrieving measurements from multiple vehicles, in order to build and update an accurate map of geolocalized landmarks. To merge the different measurements and build the map, we propose a crowdsourced mapping pipeline based on graph optimization. Furthermore, we introduce three different methods for updating the map, and identify the technique that achieves the best performances in terms of map quality and computational scalability. In the following, we formulate the problem of crowdsourced mapping, we detail our setup and models, and we present the three different map update methods.

4.4.1 Crowdsourced Model

Previously to the formulation of the crowdsourced mapping problem, we define some basic variables, without any loss of generality. We consider a given environment that

contains N landmarks whose geo-positions define a map $M = \{L_1, ...L_N\}$. Within this environment, a given number P of vehicles $V_{1:P} = \{V_1, ..., V_P\}$ drive along respective trajectories $X_{1:P} = \{X_1, ..., X_P\}$. These vehicles may drive at the same time, or during separate time windows. For clarity purposes, we omit time indexes and only specify vehicle indexes in the following. While driving, each vehicle V_p detects a given set of landmarks M_p, which is a subset of M, and is potentially different for every vehicle:

$$M_p \subset M \qquad \forall\, V_p \in V_{1:P} \tag{4.27}$$

Each vehicle V_p registers a number of observations Z_p, which can be split between:

- Displacement observations U_p associated with a motion model f, as in Equation 4.2.
- Position observations Y_p associated with a sensor model g, as in Equation 4.3.
- Detection observations D_p associated with a sensor model h, as in Equation 4.4.

The crowdsourced mapping problem consists in jointly estimating the trajectories of all vehicles, as well as the landmarks geo-positions, considering all registered observations. This problem can be formulated in a probabilistic manner, where estimations of the vehicles trajectories $\widehat{X}_{1:P}$ and landmarks geo-positions $\widehat{M}$ correspond to the maximum of the posterior PDF $\mathcal{P}$ given all observations:

$$\widehat{X}_{1:P}, \widehat{M} = \underset{X_{1:P}, M}{\arg\max}\, \mathcal{P} = \underset{X_{1:P}, M}{\arg\max}\, P(X_{1:P}, M | Z_{1:P}) \tag{4.28}$$

The posterior PDF $\mathcal{P}$ is generally too complex to solve. Therefore, in order to simplify it, we want to apply an independency hypothesis between the measurements coming from different vehicles, as explained further. Before doing so, we need to inverse the role of $X_{1:P}$, M and $Z_{1:P}$ within $\mathcal{P}$, making use of the Bayes Theorem:

$$\mathcal{P} = P(X_{1:P}, M | Z_{1:P}) = P(Z_{1:P} | X_{1:P}, M) \frac{P(X_{1:P}, M)}{P(Z_{1:P})} \tag{4.29}$$

$P(X_{1:P}, M)$ and $P(Z_{1:P})$ can be considered constant for a given application, which gives:

$$\mathcal{P} \propto P(Z_{1:P}|X_{1:P}, M) \tag{4.30}$$

Now, we introduce our hypothesis: we assume that all observations retrieved by a given vehicle are conditionally independent from observations retrieved by other vehicles, i.e. that observations Z_p do not depend on observations $Z_{p' \neq p}$.

Two conditions can potentially contradict this hypothesis. First, environmental conditions could affect similarly all the measurements issued by different vehicles that are driving at the same time. In reality, such phenomena would mainly affect position observations, as those are generally provided by GNSS receivers, which are subject to various environmental effects such as multi-path or atmospheric delaying. Secondly, the measurements issued by different vehicles could be affected by the same systematic errors, due to the use of the same sensors, or the same calibration procedures. Both of these effects could lead to the presence of similar auto-correlated and biased noises affecting the measurements of different vehicles, which would contradict our hypothesis. If those effects were revealed to severely affect the map quality, this could be dealt with by applying covariance inflation [98], or adapting the graph structure [100].

Overall, making use of this assumption transforms Equation 4.30 into:

$$\mathcal{P} \propto P(Z_1|X_{1:P}, M) \times ... \times P(Z_P|X_{1:P}, M) \tag{4.31}$$

The observations Z_p retrieved by a given vehicle V_p only depend from the trajectory X_p of this vehicle and the map M, hence:

$$\mathcal{P} \propto P(Z_1|X_1, M) \times ... \times P(Z_P|X_P, M) \tag{4.32}$$

We make use of the Bayes theorem again, in order to reverse back the variables:

$$\mathcal{P} \propto P(X_1, M|Z_1) \times ... \times P(X_P, M|Z_P) \tag{4.33}$$

In Equation 4.33, each probability $P(X_p, M|Z_p)$ corresponds to the posterior PDF of the original SLAM problem formulated for a single vehicle (see Equation 4.5). Theoretically, this could be solved by using all the measurements retrieved by the different vehicles inside a single graph optimization. In practice, this would be infeasible, as the large number of vehicles would lead to unscalable computations. This motivates the design of an iterative process as described below, which is able to update an existing map using the measurements retrieved by a single vehicle, in order to iteratively compute the map estimation.

At the beginning of crowdsourced mapping, the map estimation $\hat{M}$ is empty, as no landmark geo-position has been estimated yet. As the first vehicle V_1 detects a set of landmarks M_1, and uploads its observations Z_1, the map estimation $\widehat{M}$ can be computed by solving the following SLAM problem through graph optimization:

$$\widehat{X}_1, \widehat{M} = \underset{X_1, M}{\arg\max}\, P(X_1, M|Z_1) \tag{4.34}$$

Now, consider any vehicle V_p that is not the first vehicle. In this case, an estimation of the map $\widehat{M}_{Old}$ has already been obtained, making use of the measurements provided by the vehicles V_1 to V_{p-1}:

$$\widehat{X}_{1:p-1}, \widehat{M}_{Old} = \underset{X_{1:p-1}, M_{Old}}{\arg\max}\; P(X_{1:p-1}, M_{Old}|Z_{1:p-1}) \tag{4.35}$$

As the vehicle V_p detects a given set of landmarks M_p, and uploads its observations Z_p, the new estimation of the map $\widehat{M}$ corresponds to:

$$\widehat{X}_{1:p}, \widehat{M} = \underset{X_{1:p}, M}{\arg\max}\, P(X_{1:p}, M|Z_{1:p}) \tag{4.36}$$

Due to the hypothesis of measurements independency between vehicles:

$$\widehat{X}_{1:p}, \widehat{M} = \underset{X_{1:p},M}{\arg\max}\, P(X_{1:p-1}, M|Z_{1:p-1})P(X_p, M|Z_p) \tag{4.37}$$

To solve this equation, we have to consider two different cases. On one hand, if M_{Old} and M_p are completely disjoint, i.e. if they do not contain any landmarks in common, the problem can be split into two independent subproblems:

$$\widehat{X}_{1:p}, \widehat{M} = \underset{X_{1:p-1},M_{Old}}{\arg\max}\, P(X_{1:p-1}, M_{Old}|Z_{1:p-1}) \times \underset{X_p,M_p}{\arg\max}\, P(X_p, M_p|Z_p) \tag{4.38}$$

In this case, the problem on the left has already been solved (see Equation 4.35), while the problem on the right can be solved through graph optimization. On the other hand, if M_{Old} and M_p are not disjoint, i.e. if they include landmarks in common, M_p is conditionally dependent on M_{Old}, which transforms the problem into:

$$\widehat{X}_{1:p}, \widehat{M} = \underset{X_{1:p-1},M_{Old}}{\arg\max}\, P(X_{1:p-1}, M_{Old}|Z_{1:p-1}) \times \underset{X_p,M_p}{\arg\max}\, P(X_p, M_p|Z_p, M_{Old}) \tag{4.39}$$

Again, the problem on the left has already been solved (see Equation 4.35). Meanwhile, the problem on the right can be solved through graph optimization, by including the previous estimation of the map $\hat{M}_{Old}$ as a constraint within the graph.

This process, which is illustrated in Figure 4.4, makes use of sensors observations provided by the current vehicle, as well as the previous map estimation, to update the map with new landmarks geo-positions and covariance. This way, all the information related to the landmarks configuration brought either by the previous vehicles, or by the current one, can be used. As more and more vehicles upload their observations to update the map, this should lead to a more accurate map estimation, with a thinner covariance Σ_M. Nevertheless, this induces a strong requirement on the consistency of the map, i.e. on the validity of Σ_M. Indeed, as the map is successively used and updated within a closed loop,

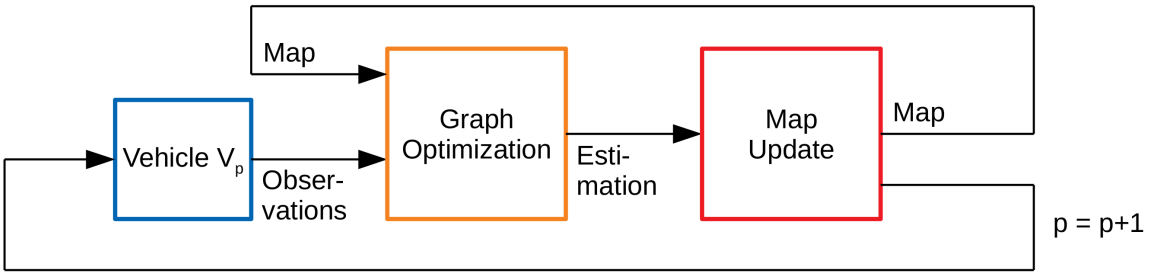

Figure 4.4: Crowdsourced Mapping Pipeline - As a vehicle V_p uploads its observations, graph optimization is performed, a new estimation is obtained, and the map is updated.

any over-confidence in the estimations could produce biased results in the long-run.

4.4.2 Setup and Sensors

So far, we have presented our crowdsourced mapping solution in the general case, without specifying the type of sensors embedded in the vehicles. In practice, during our experiments, we consider any type of vehicle equipped with at least the following set of three different sensors: motor encoders, a GNSS receiver and a monocular camera. This minimum setup was chosen for its generality, as these sensors are either already embedded on current vehicles (motor encoders), or expected to be soon part of the standard equipment (GNSS receiver, monocular camera). Furthermore, these sensors provide the different types of measurements that enable to obtain geo-position information about the surrounding landmarks. In the following, we detail the vehicle and landmarks states, and introduce the different sensors models.

Vehicle and Landmarks States The following experiments are realized on a local, horizontal plane. To avoid unnecessary complexity, we generally express all positions and orientations in the same local, horizontal plane. Beforehand, we define a number of reference frames illustrated in Figure 4.5. The local frame is depicted in black, while the vehicle frame, GNSS receiver frame and camera frame are respectively shown in blue, grey and green. In the remaining of this document, we express all variables in the local frame by

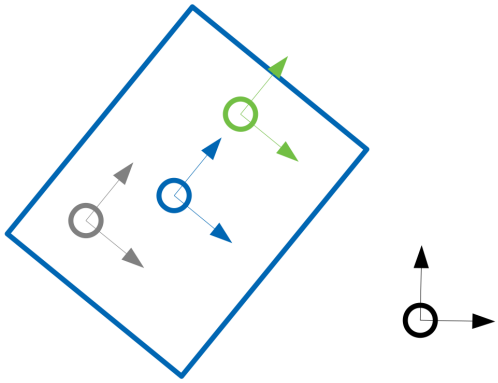

Figure 4.5: 2D representation of frames.

default. If a variable X is expressed in another frame F, it will be written ${}^{F}X$.

The vehicle state X_k at instant k is defined as:

$$X_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} \tag{4.40}$$

where x_k, y_k are the position coordinates of the vehicle, and θ_k corresponds to its orientation.

The geo-position of landmark L_i is defined as:

$$L_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{4.41}$$

where x_i, y_i are the position coordinates of the landmark.

Motor Encoders For each instant k corresponding to the frequency f_u, motor encoders provide a measurement U_k:

$$U_k = \begin{pmatrix} \nu_k \\ \xi_k \end{pmatrix} + \begin{pmatrix} \epsilon_{\nu,k} \\ \epsilon_{\xi,k} \end{pmatrix} \tag{4.42}$$

where ν_k and ξ_k correspond respectively to the ego-speed and driving wheel angle of the vehicle, while $\epsilon_{\nu,k}$ and $\epsilon_{\xi,k}$ are the associated noises. To provide information about the vehicle displacement between successive instants, the measurement U_k is associated with a motion model f (see Equation 4.13). The motion model f is defined as the bicycle model, which is a popular model for four-wheel robots that adapts well to the geometry of vehicles in standard driving conditions [105]:

$$X_k = f(X_{k-1}, U_k) + \epsilon_{u,k} = \begin{pmatrix} x_{k-1} + \frac{\nu_k}{f_u}\, cos(\theta_{k-1} + \frac{\nu_k\, sin(\xi_k)}{2\, f_u\, R}) \\ y_{k-1} + \frac{\nu_k}{f_u}\, sin(\theta_{k-1} + \frac{\nu_k\, sin(\xi_k)}{2\, f_u\, R}) \\ \theta_{k-1} + \frac{\nu_k\, sin(\xi_k)}{f_u\, R} \end{pmatrix} + \epsilon_{u,k} \tag{4.43}$$

where $\epsilon_{u,k}$ is the prediction noise, and R is the length of the vehicle's axle.

GNSS Receiver For each instant k corresponding to the frequency f_y, the GNSS receiver provides a measurement Y_k:

$$Y_k = \begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} + \epsilon_{y,k} \tag{4.44}$$

where $\bar{x}_k$, $\bar{y}_k$ correspond to the position coordinates of the center of the GNSS receiver, and $\epsilon_{y,k}$ is the sensor noise. The measurement Y_k is associated with a sensor model g (see Equation 4.14) defined as follows:

$$Y_k = g(X_k) + \epsilon_{y,k} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} cos(\theta_k) & -sin(\theta_k) \\ sin(\theta_k) & cos(\theta_k) \end{pmatrix} T_V^{GNSS} + \epsilon_{y,k} \tag{4.45}$$

where T_V^{GNSS} is the translation vector from the vehicle frame to the GNSS receiver frame, obtained through an extrinsic calibration procedure.

Monocular Camera For each instant k corresponding to the frequency f_d, the monocular camera provides an image $\mathcal{I}_k$. During the different experiments presented in chapter 5,

we use various methods to detect landmarks within images, such as using a CNN-derived architecture, or operating the detections manually. In the following, we define the general camera model without specifying the landmarks detection method.

As stated previously, we express all positions in a local, horizontal plane. Furthermore, within the extrinsic calibration of the camera, we consider this latter to remain aligned with the horizontal plane. Therefore, we choose to only consider the horizontal pixel coordinate within landmarks detections. For each landmark L_i detected in the image $\mathcal{I}_k$, a measurement $D_k(L_i)$ is obtained:

$$D_k(L_i) = \lambda + \epsilon_{d,k} \tag{4.46}$$

where λ is the horizontal pixel coordinate of the landmark L_i within the image $\mathcal{I}_k$, and $\epsilon_{d,k}$ is the sensor noise. The measurement $D_k(L_i)$ is associated with a sensor model h (see Equation 4.15) defined below.

$$^{V}L_i = \begin{pmatrix} cos(\theta_k)\,(x_i - x_k) + sin(\theta_k)\,(y_i - y_k) \\ -sin(\theta_k)\,(x_i - x_k) + cos(\theta_k)\,(y_i - y_k) \\ 0 \end{pmatrix} \tag{4.47}$$

$$^{C}L_i = (R_V^C)^{-1} \times {}^{V}L_i - (R_V^C)^{-1} \times T_V^C \tag{4.48}$$

with (R_V^C) and (T_V^C) being respectively the rotation matrix and translation vector from the vehicle frame to the camera frame, both obtained through an extrinsic calibration procedure.

$$D_k(L_i) = -K(0,0)\,\frac{^{C}L_i(1)}{^{C}L_i(0)} + K(0,2) + \epsilon_{d,k} \tag{4.49}$$

where K is the intrinsic calibration matrix of the camera, $K(x,y)$ is the value of K at the x^{th} row and y^{th} column, and $^{C}L_i(x)$ is the x^{th} value of $^{C}L_i$. This equation is based on the pinhole projection model, which is a popular model designed for monocular cameras.

4.4.3 Map Update Strategies

As a given vehicle V_p drives along a trajectory $X_{0:K}$, it registers observations $Z_{0:K}$, which can be split between displacement, position and detection observations $U_{1:K}$, $Y_{0:K}$, $D_{0:K}$. These measurements are periodically uploaded towards centralized servers, in order to compute graph optimization, and build and update the map $\{\hat{M}, \Sigma_M\}$ (see Figure 4.4). In the case where a prior version of the map $\{\hat{M}_{Old}, \Sigma_{M_{Old}}\}$ has already been established, its content should be included within graph optimization. To achieve this, we propose three different methods:

- Correlated Constraint (CC).
- Decorrelated Constraints (DC).
- Correlated Constraint and Graph Subdivisions (CC+GSD).

These different approaches correspond to different trade-offs between the map quality and the computational cost, as explained in the following. We compare their performances later in this document, in order to assess the best strategy. To do so, we make use of simulation experiments, in which different configurations with different numbers of landmarks are considered. In each configuration, multiple vehicles are driven along a given trajectory, in order to provide successive map updates.

Correlated Constraint In a first approach (CC), we consider that all the information available in the map, including cross-correlations between landmarks geo-positions, should be included. Therefore, we propose to use a correlated map constraint CC. Let $L_{I:J} = \{L_I, ..., L_J\}$ be the subset of landmarks detected by the vehicle that are also in the map. We build a correlated constraint CC of covariance Ω_{CC}^{-1} and information Ω_{CC} as:

$$CC = \hat{M}_{Old}(L_{I:J}) \tag{4.50}$$

$$\Omega_{CC} = \Omega_{M_{Old}}(L_{I:J}) \tag{4.51}$$

where:

- $\hat{M}_{Old}(L_{I:J})$ is the subvector of $\hat{M}_{Old}$ containing the geo-positions of all landmarks in $L_{I:J}$.
- $\Omega_{M_{Old}} = \Sigma^{-1}_{M_{Old}}$, and $\Omega_{M_{Old}}(L_{I:J})$ is the submatrix of $\Omega_{M_{Old}}$ containing the information of all landmarks in $L_{I:J}$. Note that $\Omega_{M_{Old}}(L_{I:J})$ contains both the information and cross-information of landmarks in $L_{I:J}$.

Previously, we assumed measurements independency between vehicles. Therefore, the correlated constraint CC, which contains knowledge brought in the map by previous vehicles, can be assumed conditionally independent from observations $\{U_{1:K}, Y_{0:K}, D_{0:K}\}$ of the current vehicle. The posterior PDF (right probability in Equation 4.39) corresponds to:

$$P = P(X_{0:K}, M | U_{1:K}, Y_{0:K}, D_{0:K}, CC) \tag{4.52}$$

with:

$$P \propto P(CC|M) \prod_{k=1}^{K} P(X_k|X_{k-1}, U_k) \prod_{k=0}^{K} P(Y_k|X_k) \prod_{k=0}^{K} P(D_k|X_k, M) \tag{4.53}$$

The correlated constraint CC provides an observation of the geo-positions of all landmarks in $L_{I:J}$, as illustrated in Figure 4.6, and is assumed to be affected by a zero-mean, Gaussian noise of covariance Ω^{-1}_{CC} and information Ω_{CC}. Thus, we can rewrite the cost function of graph optimization (see Equation 4.23) as:

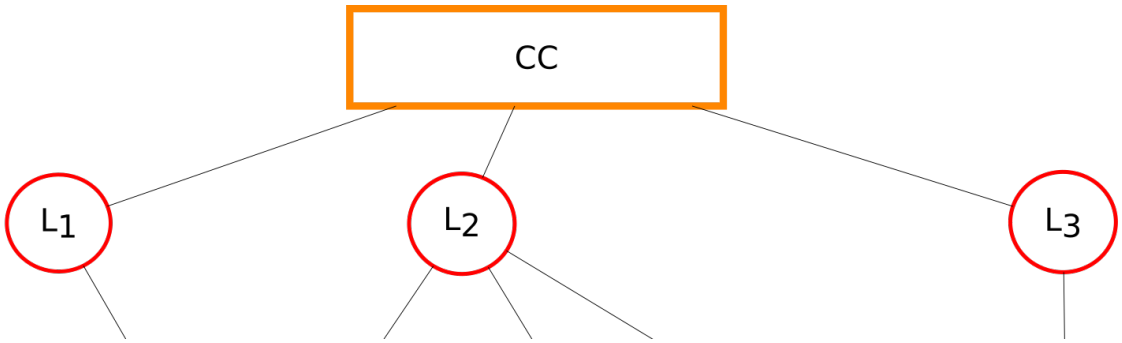

Figure 4.6: Graph with correlated map constraint.

$$
\begin{aligned}
F = & e_{CC}^{\top} \times \Omega_{CC} \times e_{CC} + \sum_{k=1}^{K} e_{u,k}^{\top} \times \Omega_{u,k} \times e_{u,k} \\
& + \sum_{k=0}^{K} e_{y,k}^{\top} \times \Omega_{y,k} \times e_{y,k} + \sum_{k=0}^{K} e_{d,k}^{\top} \times \Omega_{d,k} \times e_{d,k}
\end{aligned}
\tag{4.54}
$$

where e_{CC} is the error function associated with the correlated constraint CC:

$$
e_{CC} = CC - L_{I:J} \tag{4.55}
$$

and:

$$
\widehat{X}_{0:K}, \widehat{M} = \underset{X_{0:K}, M}{\arg\min} F \tag{4.56}
$$

At the end of graph optimization, we obtain a new estimation of the map, with new landmarks geo-positions $\hat{M}$, along with the covariance Σ_M which can be extracted from the global covariance Σ (see Equation 4.26). This operation is equivalent to marginalizing the vehicle nodes, and leads to a fully dense map with a dense covariance Σ_M. As a result, the correlated constraint CC will have a dense information Ω_{CC}, which can potentially make the graph dense as well, and lead to intensive computations during graph optimization. Thus, the correlated constraint CC corresponds to a strategy in which we favor map accuracy and consistency, potentially at the expense of scalability.

Decorrelated Constraints In order to lower the computational cost associated with graph optimization, some works proposed to maintain graph sparsity after marginalization, by approximating the dense graph structure into a new sparse structure [106]. These methods determine a new sparse topology for the graph, and compute new constraints parameters that best approximate the original graph. Such approximations can lead to an information loss and a degradation of the map accuracy, especially if they are repeated multiple times, as would be the case in a long-term crowdsourced mapping process.

To verify whether this drawback can be critical, we propose to use a simple sparsification approximation as a second approach (DC). Let $L_{I:J} = \{L_I, ..., L_J\}$ be the subset of landmarks detected by the vehicle that are also in the map. We assume that all landmarks estimations are conditionally independent from each other, which translates into neglecting cross-correlations between landmarks in the map, and building multiple decorrelated map constraints $DC = \{DC_I, ..., DC_J\}$ as follows. For each landmark L_i in $L_{I:J}$, we build a decorrelated constraint DC_i of covariance $\Omega^{-1}_{DC_i}$ and information Ω_{DC_i} as:

$$DC_i = \hat{M}_{Old}(L_i) \tag{4.57}$$

$$\Omega_{DC_i} = \Sigma_{M_{Old}}(L_i)^{-1} \tag{4.58}$$

where:

- $\hat{M}_{Old}(L_i)$ is the subvector of $\hat{M}_{Old}$ containing the geo-position of landmark L_i.
- $\Sigma_{M_{Old}}(L_i)$ is the diagonal block of $\Sigma_{M_{Old}}$ containing the covariance of landmark L_i.

The decorrelated constraints DC are built using knowledge brought by previous vehicles, and can be assumed conditionally independent from observations $\{U_{1:K}, Y_{0:K}, D_{0:K}\}$ of the current vehicle, due to our hypothesis of measurements independency between vehicles. Furthermore, we assumed all decorrelated constraints DC to be conditionally independent from

each other. Therefore, the posterior PDF (right probability in Equation 4.39) corresponds to:

$$P = P(X_{0:K}, M | U_{1:K}, Y_{0:K}, D_{0:K}, DC) \tag{4.59}$$

with:

$$P \propto \prod_{i=I}^{J} P(DC_i|M) \prod_{k=1}^{K} P(X_k|X_{k-1}, U_k) \prod_{k=0}^{K} P(Y_k|X_k) \prod_{k=0}^{K} P(D_k|X_k, M) \tag{4.60}$$

Any decorrelated constraint DC_i provides an observation of the geo-position of landmark L_i, as illustrated in Figure 4.7, and is assumed to be affected by a zero-mean, Gaussian noise of covariance $\Omega_{DC_i}^{-1}$ and information Ω_{DC_i}. Thus, we can rewrite the cost function of graph optimization (see Equation 4.23) as:

$$\begin{aligned} F = &\sum_{i=I}^{J} e_{DC_i}^{\top} \times \Omega_{DC_i} \times e_{DC_i} + \sum_{k=1}^{K} e_{u,k}^{\top} \times \Omega_{u,k} \times e_{u,k} \\ &+ \sum_{k=0}^{K} e_{y,k}^{\top} \times \Omega_{y,k} \times e_{y,k} + \sum_{k=0}^{K} e_{d,k}^{\top} \times \Omega_{d,k} \times e_{d,k} \end{aligned} \tag{4.61}$$

where e_{DC_i} is the error function associated with the decorrelated constraint DC_i:

$$e_{DC_i} = DC_i - L_i \tag{4.62}$$

and:

$$\widehat{X}_{0:K}, \widehat{M} = \underset{X_{0:K}, M}{\arg\min}\, F \tag{4.63}$$

Overall, in this approach, we neglect all cross-correlations between landmarks in the map. Doing so, we neglect some of the map information used within graph optimization, which may lead to over-confident map updates, and biased results in the long-run. On the other hand, this should keep the graph sparse, and thereby improve the computational cost of graph optimization. Hence, this strategy, which consists in making use of decorrelated

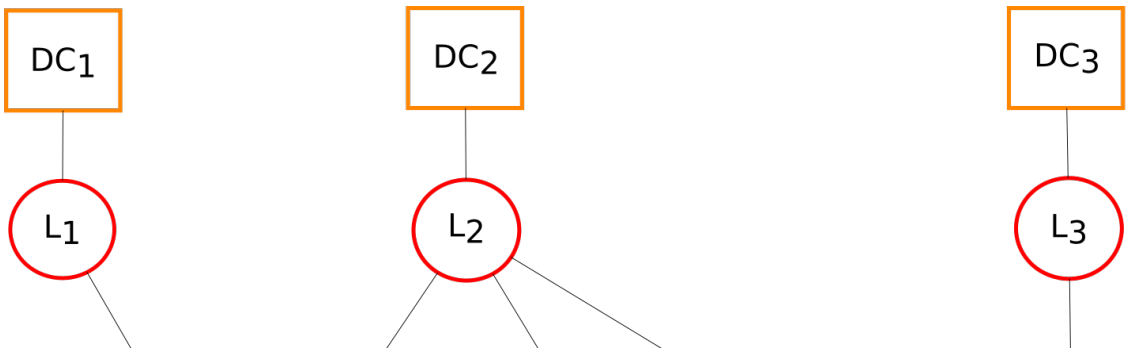

Figure 4.7: Graph with decorrelated map constraints.

constraints DC, favors scalability over map accuracy and consistency.

Graph Subdivisions To limit computations, another strategy consists in limiting the graph size, i.e. the number of nodes within the graph. Generally, SLAM methods build a map in a single-run, which makes it difficult to limit the graph size without losing information, or impeding the accuracy [5]. In the case of crowdsourced mapping, observations are retrieved from multiple vehicles to build and update a map. To achieve this, observations originating from different vehicles are considered independently through an iterative process. Thus, a specific advantage of the crowdsourcing approach is the ability to divide each graph into a number of independent subgraphs, which can then be processed as if they originated from different vehicles. This would enable to control the size of each subgraph, and thus to limit the computational cost.

As a third approach (CC + GSD), we propose to use the following graph subdivisions strategy. When a vehicle detects a set of landmarks and uploads its observations, a graph $\mathcal{G}$ is built. Instead of optimizing the whole graph, we divide it into multiple independent subgraphs $\{\mathcal{G}_1, \mathcal{G}_2, ...\}$, as shown in Figure 4.8, such that any subgraph $\mathcal{G}_i$ respects the following dimension condition:

$$D(\mathcal{G}_i) \leq \mathcal{D} \tag{4.64}$$

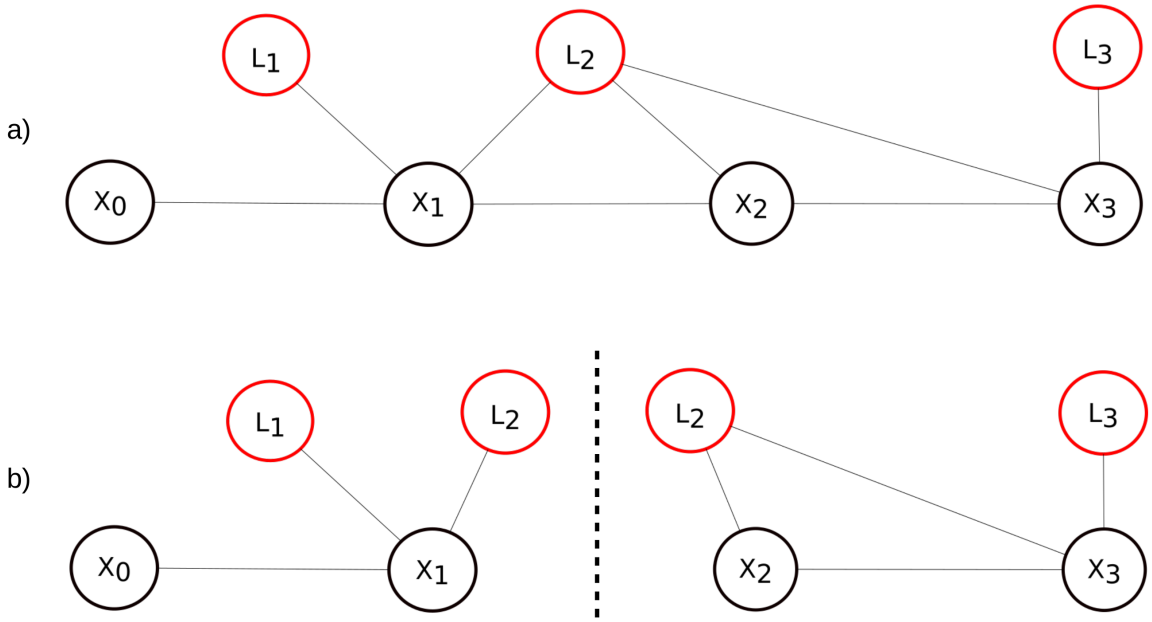

Figure 4.8: Graph subdivision.

where:

- $D(\mathcal{G}_i)$ corresponds to the dimension of the global state vector of $\mathcal{G}_i$, which is the vector containing the states of all vehicle nodes and landmarks nodes within $\mathcal{G}_i$.
- $\mathcal{D}$ is the maximum dimension accepted per subgraph.

Each subgraph $\mathcal{G}_i$ is processed as any other graph, by applying graph optimization and updating the map. The map updates take place after each graph optimization, and subgraph $\mathcal{G}_i$ can profit from the map update obtained after the optimization of subgraph $\mathcal{G}_{i-1}$. A depiction of this process is shown in Figure 4.8. The graph (a) is divided into 2 subgraphs (b). A first optimization leads to the update of L_1 and L_2 in the map, while a second optimization leads to the update of L_2 and L_3 in the map. During optimization of the subgraphs, we consider the correlated constraint CC, which models all the information available in the map. Although this potentially makes the subgraphs dense, the fact that we control their dimension enables to limit the computational burden.

As we divide the graph $\mathcal{G}$ into independent subgraphs $\{\mathcal{G}_1, \mathcal{G}_2, ...\}$, we inevitably lose

some information. Especially, as illustrated in Figure 4.8, we lose the constraints associated with displacement observations that link the vehicle nodes at the break points where the graph is split. This may slightly worsen the resulting accuracy of each map update, but not its consistency, as we simply remove a few constraints and do not make any approximation. Thus, as more and more vehicles upload their observations, the accuracy of the map should still increase. In this regard, this strategy, which consists in associating the correlated constraint CC with graph subdivisions, favors scalability and map consistency. This is potentially at the expense of the rate at which the map accuracy is improved, but not at the expense of map accuracy.

4.5 Conclusion

Crowdsourced mapping represents a cost-effective solution for building an accurate, and always up-to-date, map of landmarks. It consists in collecting measurements from many different vehicles equipped with standard sensors, in order to build and update the map. The typical crowdsourcing process contains several different modules, such as the acquisition, pre-processing and mapping modules. In this work, we focus on the mapping process, which consists in using pre-processed sensors measurements to provide map updates.

To evaluate the potential of crowdsourced mapping, we proposed and formulated a crowdsourcing solution based on triangulation optimization. This solution was evaluated using field-tests, which showed the interest of the collaborative approach, but also confirmed limitations associated with triangulation optimization. To improve our mapping system, we therefore proposed and formulated another crowdsourcing solution based on the use of graph optimization, and we introduced three different strategies for including and updating the map within the optimization process. The first method, CC (Correlated Constraint), corresponds to a strategy that favors map accuracy at the potential expense of scalability. The two other methods, DC (Decorrelated Constraints) and CC+GSD (Correlated Constraint and Graph Subdivisions), represent two modifications of the CC approach that aim to reduce

the computational cost. Indeed, the objective for crowdsourced mapping is to establish an accurate map of geo-positioned landmarks, while remaining computationally scalable, in order to deal with a large quantity of measurements and build a map that can span over large areas. In the next chapter, we perform extensive simulation experiments and field-tests, in order to establish the best-performing method, as well as to evaluate the potential of crowdsourced mapping to build an accurate map of landmarks.

CHAPTER 5
EVALUATION OF CROWDSOURCED MAPPING PERFORMANCES

Crowdsourced mapping aims at taking advantage from measurements retrieved by multiple vehicles in order to build an accurate map of geolocalized landmarks. To assess its capability to successively improve the map accuracy as more and more vehicles upload their observations, we start by evaluating the performances of our triangulation-based approach (see section 4.2) during field-tests.

Nevertheless, in order to provide a highly-accurate map, crowdsourced mapping must be able to scale to a large number of vehicles, as well as to merge measurements that can be of different types and affected by different noises levels. For this reason, we proposed another crowdsourced mapping pipeline based on graph optimization (see subsection 4.4.1), along with three related strategies to include and update the map within the optimization (see subsection 4.4.3). To assess the performances of our solution, and confirm the benefit of the crowdsourcing approach, we make use of extensive simulation and field experiments, in which we investigate whether the accuracy of the built map increases with the number of participating vehicles. We expect that the map will reach a high accuracy at the condition that it remains consistent, i.e. that its estimated accuracy remains valid. Further, we compare the different map update strategies, and assess the scalability potential of crowdsourced mapping. To fully understand its strengths and limitations, we also confront our solution to various types of noises during simulation experiments, including GNSS auto-correlated noises and camera calibration bias, and to real data retrieved by our test-vehicle. Finally, we assess in simulation the potential of our solution to enhance the localization of vehicles, by evaluating whether a typical map built by our crowdsourced mapping solution enables to reach the targeted positioning accuracy.

The structure of this chapter is as follows. In section 5.1, we study the map accuracy

provided by the triangulation-based approach during early field-tests. In section 5.2, we assess the accuracy and scalability potential of the graph-based approach, using various numbers of landmarks and noises configurations during simulation experiments. In section 5.3, we evaluate the performances of our graph-based solution during field experiments, and we compare those with results obtained during simulation experiments, in order to draw conclusions both from a theoretical and practical viewpoint. In section 5.4, we evaluate the benefits of the proposed approach for localization purposes, by making use of simulation experiments with various numbers of landmarks and noises configurations.

5.1 Triangulation-based Approach: Evaluation through Field-Tests

To evaluate the potential of crowdsourced mapping, we perform field-tests with our test-vehicle, and build a map of landmarks using the triangulation-based approach. In particular, we aim to verify that the resulting map accuracy effectively improves as more measurements are issued by the vehicle. These experiments were realized at the beginning of our work, and constitute a basis to assess the interest of the crowdsourcing approach. Moreover, they allow the reader to figure out the configuration and environment associated with field-tests.

5.1.1 Field Setup

Throughout the experiment, the test-vehicle was driven in a rather flat area. Thus, in order to avoid unnecessary complexity, we perform the experiment in 2D, by projecting all positions in a local, horizontal plane. The test-vehicle was driven along a $4\ km$ loop in the city center of Versailles, France, for a total of 10 passages. The 10 passages were registered through numerous driving sessions, which spanned over multiple days. Thus, they properly represent 10 different vehicles following the given trajectory, as they were impacted by different environmental conditions, such as different satellites configurations. To get insight about the vehicle trajectory, GNSS positions received during one of the passages are shown in Figure 5.9. Along the loop, traffic signs were identified, and used as landmarks constituting

the map. We measured their geo-positions using a static RTK-GPS receiver, in order to provide a ground truth to compare our results with. Due to this demanding process, we used a limited number of landmarks: $M = L_1, ..., L_{10}$.

The test-vehicle that was used is a Citroën Picasso car equipped with:

- Motor encoders attached to the four wheels of the vehicle, as well as to its driving wheel. The encoders attached to the wheels enable to compute the ego-speed of the vehicle through a temporal filter, while the encoder attached to the driving wheel provides its angle. Thus, at each instant k corresponding to the frequency $f_u = 25.0\ Hz$, a displacement observation U_k can be obtained.

- A natural GNSS receiver equipped with an NL-8004U antenna. Although this antenna is able to receive GNSS signals from multiple satellite networks (GPS, Galileo, etc...), we only use GPS signals within our experiment. We transform the (Latitude, Longitude) coordinates provided by the GNSS receiver into (East, North) coordinates in the local, horizontal plane. Thus, at each instant k corresponding to the frequency $f_y = 1.0\ Hz$, the GNSS receiver provides a position observation Y_k.

- A monocular camera AXIS P3905-R, providing images in 1080p (1920x1080 pixels) with a view-angle of $60°$. The camera was installed behind the vehicle windscreen, and looking in front of the vehicle. For each instant k corresponding to the frequency $f_d = 30\ Hz$, the camera provides an image, in which traffic signs are detected using a CNN-derived architecture [107]. While this method enables to detect visible traffic signs, it does not perform data association, i.e. it does not inform about which sign is being detected. Therefore, we perform this step manually, by selecting for each detection the corresponding traffic sign, in order to evaluate the performances of our crowdsourced mapping solution independently from data association issues.

5.1.2 Evolution of Map Accuracy

To build the map of landmarks, we apply the triangulation-based approach defined in section 4.2, which was implemented in Python. This approach makes use of position and detection measurements, but not displacement measurements. During each vehicle passage, spanning for instance from $k = 0$ to $k = K$, position observations $Y_{0:K}$ and detection observations $D_{0:K}$ are used to build projection lines for all detected landmarks as follows.

For each position observation Y_k, we consider the closest detection observation $D_k(L_i)$ of a given landmark L_i. The projection line $Z_k(L_i)$ is built as passing through two points A and B defined as:

$$A = R_V^W \, T_{GNSS}^C + Y_k \tag{5.1}$$

$$B = A + R_V^W R_C^V \begin{pmatrix} 1 \\ -K^{-1}(0,0) \; D_k(L_i) - K^{-1}(0,2) \end{pmatrix} \tag{5.2}$$

where R_V^W is the rotation matrix from the vehicle frame to the local frame:

$$R_V^W = \begin{pmatrix} cos(\hat{\theta}_k) & -sin(\hat{\theta}_k) \\ sin(\hat{\theta}_k) & cos(\hat{\theta}_k) \end{pmatrix} \tag{5.3}$$

and $\hat{\theta}_k$ is the estimation of the vehicle orientation, obtained using successive position observations. T_{GNSS}^C is the translation vector from the GNSS receiver frame to the camera frame, expressed in the vehicle frame. R_C^V is the rotation matrix from the camera frame to the vehicle frame, and K corresponds to the intrinsic calibration matrix of the camera. In Figure 5.1, we depict in red a projection line associated with the detection of a traffic sign, along with the local frame, vehicle frame, GNSS receiver frame and camera frame respectively shown in black, blue, grey and green.

After each vehicle passage, all projection lines established by the current vehicle, as well

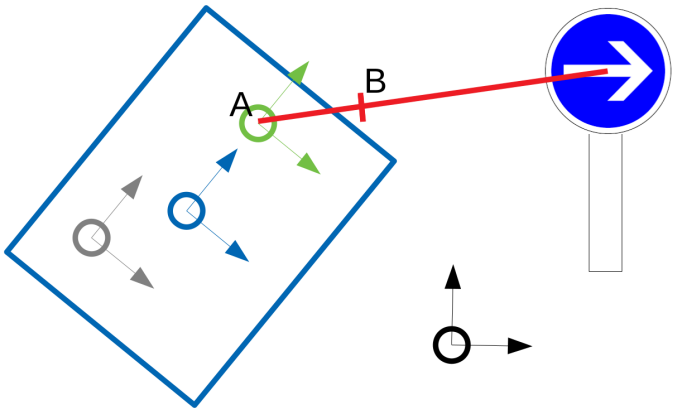

Figure 5.1: 2D representation of frames, along with a projection line associated with the detection of a traffic sign.

as the previous ones, are used within a triangulation optimization in order to update the map. In Figure 5.2, the evolution of the distance error is shown for each traffic sign, along with the evolution of the average map error over all traffic signs, which is shown at the bottom. We observe that the average map error goes from $12.2\ m$ after the first vehicle passage, to $5.0\ m$ after the last vehicle passage, which illustrates the capacity of crowdsourced mapping to take advantage from measurements issued by multiple vehicles equipped with standard sensors. Nevertheless, the errors associated with the positioning of traffic signs 3 to 8 do not appear to decrease much over the vehicle passages (see Figure 5.2), and the average map error tends to converge towards a bias of $5.0\ m$. This important error is likely due to the presence of unmodeled noises within GNSS measurements and camera detections. Furthermore, while the triangulation optimization remains a simple and intuitive approach, it does not make use of all the available information, and especially of the displacement observations obtained from motor encoders.

5.2 Graph-based Approach: Evaluation through Simulation Experiments

To address issues associated with triangulation optimization, we proposed another crowdsourcing pipeline based on graph optimization (see subsection 4.4.1), and introduced three

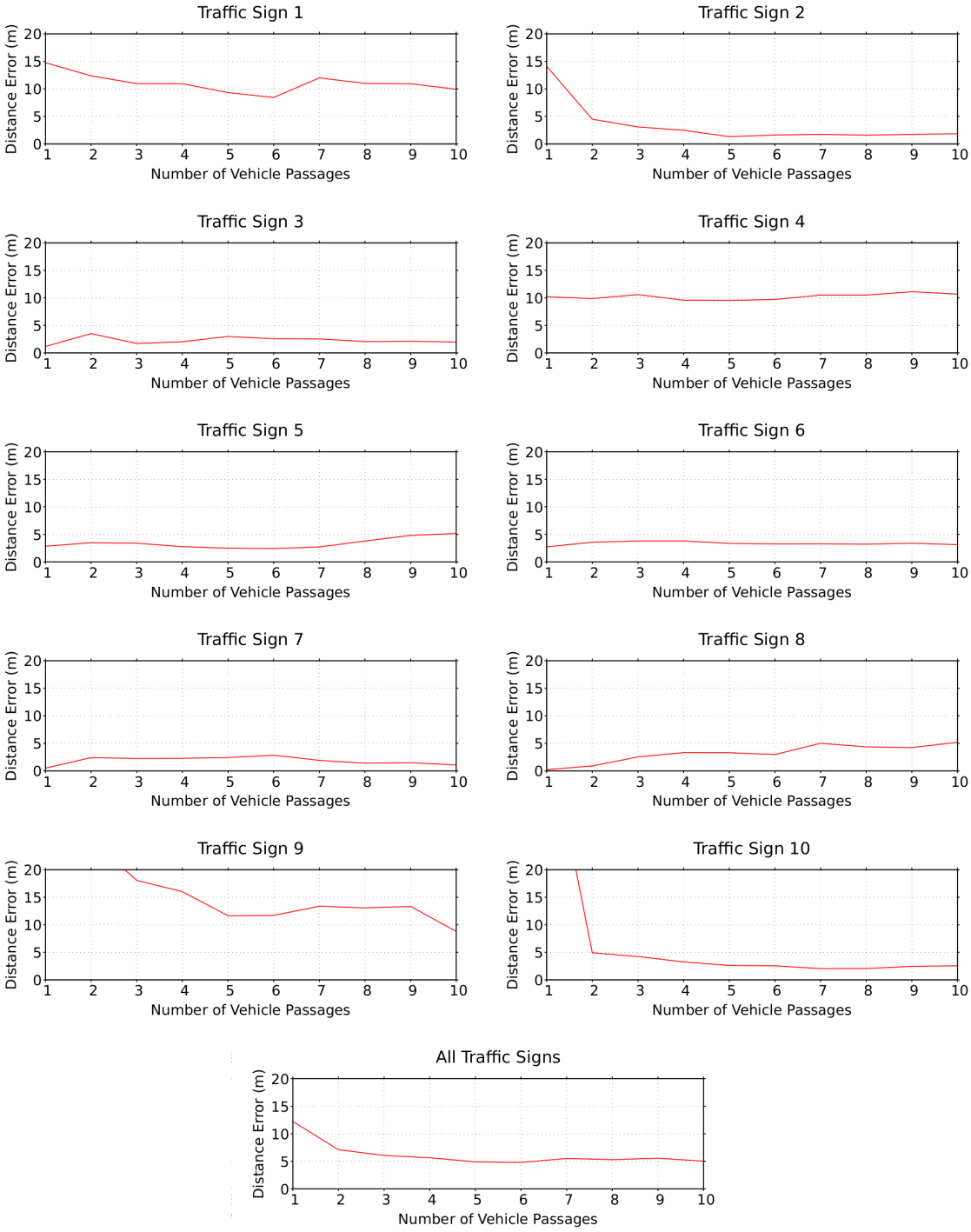

Figure 5.2: Evolution of distance errors for the positioning of traffic signs estimated by the triangulation-based approach during early field-tests.

different strategies for updating and including the map within the optimization (see subsection 4.4.3). To evaluate the performances of this crowdsourced mapping solution, we first perform simulation experiments using different configurations. In subsection 5.2.1, we introduce the simulation setup. In subsection 5.2.2, we make the number of landmarks vary and assess the scalability potential of crowdsourced mapping. In subsection 5.2.3, we introduce different types of noises, including auto-correlated noises and calibration bias, and evaluate the robustness of our solution.

5.2.1 Simulation Setup

The simulation was implemented in C++, and built using the same vehicle path, embedded sensors, and noises amplitudes as during field-tests, in order to produce a realistic simulation. Similarly as during field-tests, we perform simulation experiments in 2D, by projecting all positions in a local, horizontal plane. The trajectory followed by the vehicles is generated assuming that all vehicles follow the same trajectory, which is depicted in Figure 5.3. This trajectory corresponds to a portion of the real path driven by our test-vehicle in real situation (see Figure 5.9), and therefore possesses a realistic shape and curvature. We decided to only use a portion of about $2\ km$ of the real path in the following experiments, mainly for simplification purposes, since we focus here on studying the potential of our solution to scale to different densities of landmarks, and to account for different types of noises.

The static landmarks $M = \{L_1, ..., L_N\}$ are generated following 5 different configurations, which respectively contain $N = 4$, $N = 25$, $N = 50$, $N = 75$, and $N = 100$ landmarks. The first configuration contains $N = 4$ landmarks, which correspond to 4 traffic signs identified on the real path. This constitutes a sparse configuration with realistic landmarks geo-positions. The other configurations contain the same 4 landmarks, as well as a given number of additional landmarks, which were randomly generated along the vehicle trajectory. For example, the dense configuration with $N = 100$ landmarks contains the same 4 landmarks and 96 randomly-generated landmarks. The additional landmarks were

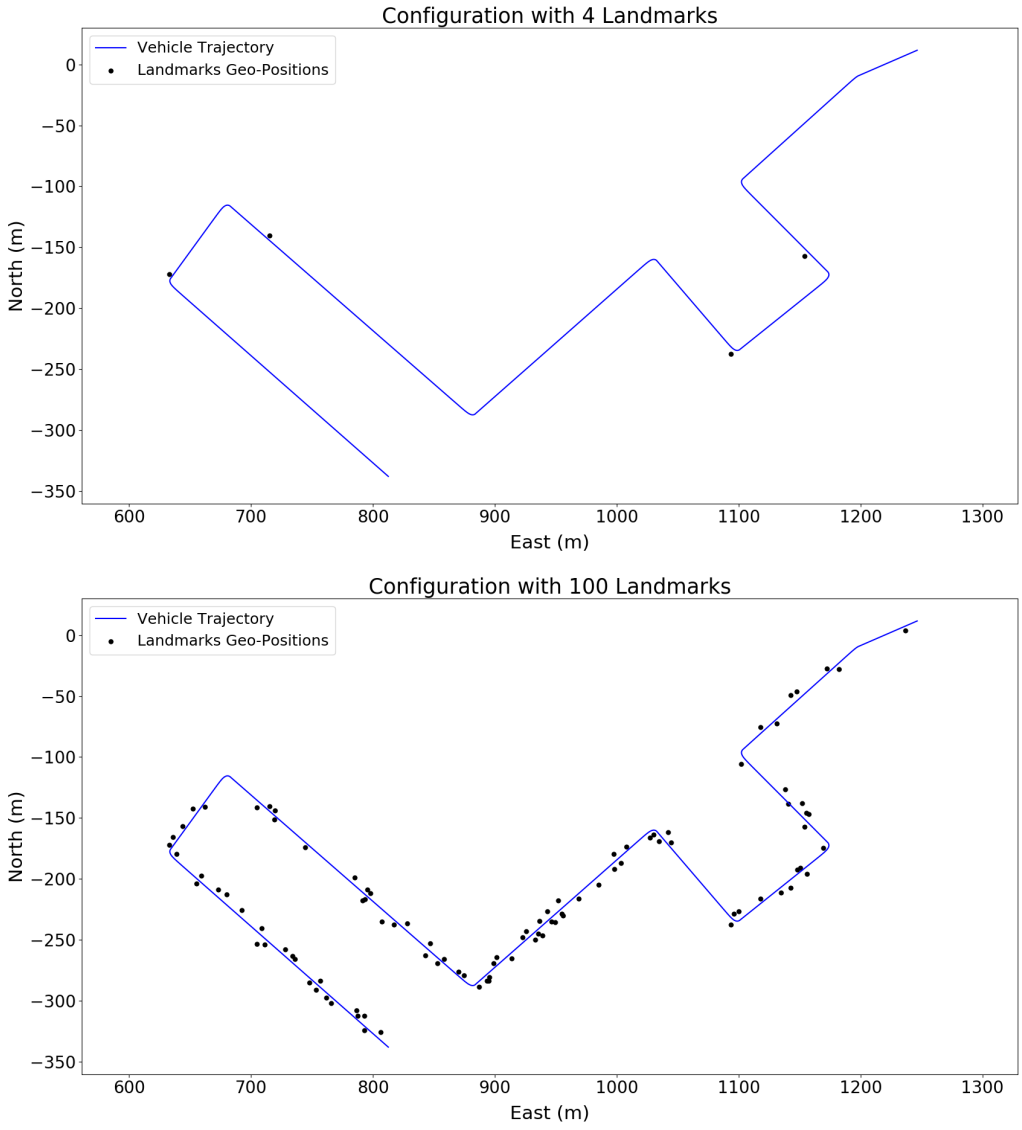

Figure 5.3: Ground truth vehicle trajectory and landmarks geo-positions used in simulation experiments.

generated making use of a uniform distribution. Although this might not accurately represent the true distribution of some types of landmarks such as traffic signs, this is not relevant here, since we focus on studying the scalability and robustness potential of our solution. For readability purposes, we only show the configurations with $N = 4$ and $N = 100$ landmarks in Figure 5.3.

We consider each vehicle to be equipped with motor encoders, a GNSS receiver, and a monocular camera. This enables to reproduce similar types of measurements as during field-tests, as the same sensors are embedded on our test-vehicle. We also consider the sensors to be installed with the same extrinsic calibration, and operated at similar frequencies, as on our test-vehicle:

- The motor encoders provide displacement measurements U_k, as defined in Equation 4.42, at a frequency of $f_u = 25.0\ Hz$.
- The GNSS receiver provides position measurements Y_k, as defined in Equation 4.44, at a frequency of $f_y = 1.0\ Hz$.
- The monocular camera provides detection measurements D_k, as defined in Equation 4.46, at a frequency of $f_d = 2.0\ Hz$. This frequency does not correspond to the frequency at which images are produced, but to the frequency at which the landmarks detection algorithm is operated. The considered intrinsic calibration matrix K (see Equation 4.49) is the same as the one measured on the real monocular camera.

5.2.2 Scalability Assessment

In order to assess the scalability potential of our solution, we evaluate its capacity to build an accurate and consistent map, which can contain various densities of landmarks, while remaining computationally scalable. We consider different configurations that contain different numbers of landmarks, and compare the performances obtained in these configurations using each of the three map update strategies proposed in subsection 4.4.3. These strategies

correspond to different trade-offs between the map quality and the computational cost, and their comparison will allow to conclude on the scalability potential of crowdsourced mapping.

White Gaussian Noises To keep the focus on the assessment of scalability, we consider noises to be sampled from white and Gaussian distributions as follows. This corresponds to the basic assumption used to formulate the graph optimization problem (see subsection 4.3.2).

- $\epsilon_{u,k}$, the noise associated with a given displacement measurement U_k, is sampled from a zero-mean, Gaussian distribution with covariance Σ_u and information Σ_u^{-1}:

$$\Sigma_u = \begin{pmatrix} \delta_\nu^2 & 0.0 \\ 0.0 & \delta_\xi^2 \end{pmatrix} \tag{5.4}$$

- $\epsilon_{y,k}$, the noise associated with a given position measurement Y_k, is sampled from a zero-mean, Gaussian distribution with covariance Σ_y and information Σ_y^{-1}:

$$\Sigma_y = \begin{pmatrix} \delta_{\bar{x}}^2 & 0.0 \\ 0.0 & \delta_{\bar{y}}^2 \end{pmatrix} \tag{5.5}$$

  In order to ease the evaluation of the scalability potential, it is desirable to avoid unnecessary variability of the noises covariances. Therefore, we assume a constant DOP (Dilution Of Precision) for all GNSS measurements. This translates into the constant matrix Σ_y defined in Equation 5.5.

- $\epsilon_{d,k}$, the noise associated with a given detection measurement D_k, is sampled from a zero-mean, Gaussian distribution with covariance Σ_d and information Σ_d^{-1}:

$$\Sigma_d = \begin{pmatrix} \delta_\lambda^2 \end{pmatrix} \tag{5.6}$$

Motor Encoders		GNSS Receiver		Camera
$\delta_\nu\ (m/s)$	$\delta_\xi\ (rd)$	$\delta_{\bar{x}}\ (m)$	$\delta_{\bar{y}}\ (m)$	$\delta_\lambda\ (Pixels)$
0.56 ($\sim 2.0\ km/h$)	0.044 ($\sim 2.5\ °$)	10.0	10.0	5.0

Table 5.1: Standard deviations of noises applied to sensors measurements during simulation experiments.

To generate sensors measurements with realistic noises amplitudes, we studied the typical errors obtained in real conditions for each sensor on our test-vehicle, and looked for the worst-case scenarios. For example, odometry measurements were most impacted by noise when the vehicle was driving at around $50\ km/h$, and GNSS measurements were most affected by noise in dense environments. In the simulation, we set all the noises standard deviations to the typical errors obtained in the respective worst-case scenarios. We summarize these standard deviations in Table 5.1. Note that δ_λ is the standard deviation of the noise affecting camera detections, and is thus expressed in pixels. Considering the properties of the monocular camera, this corresponds to a bearing deviation of around 0.006 $rd\ (\sim 0.3\ °)$.

In total, we simulate 1000 vehicles driving along the trajectory, and generate their respective displacement, position and detection measurements. During each vehicle passage, every landmark can be detected multiple times, which leads to multiple detection measurements for each landmark. In practice, we observed that making use only of the last detections, i.e. the detections whose pixel-coordinates are the most distant from each other, leads to a better stability of the optimization process. Thus, at each vehicle passage, when a landmark is detected by the vehicle, we only consider the last 5 detection measurements obtained for this landmark.

Results For each configuration, containing from $N = 4$ to $N = 100$ landmarks, we process the 1000 vehicle passages as follows. At the end of a given passage, as the vehicle reaches the end of the trajectory, it shares its observations with a centralized server, which

then performs graph optimization, and updates the map with new landmarks geo-positions and covariance. This mapping process was implemented in Python, and run on a local computer with an Intel Xeon Silver 4114 CPU. For a given passage, the graph contains around 165 position constraints, as well as 5 detection constraints for each landmark, while displacement constraints are built to link all vehicle nodes together. Whenever prior knowledge regarding the landmarks geo-positions is available, which happens as soon as with the second vehicle passage, we add a map constraint to the graph (see subsection 4.4.1). To model this map constraint, the three different map update strategies introduced in subsection 4.4.3 are considered:

- CC, the correlated constraint (see paragraph 4.4.3).
- DC, the decorrelated constraints (see paragraph 4.4.3).
- $CC + GSD$, the correlated constraint along with graph subdivisions (see paragraph 4.4.3) using the following dimension condition:

$$D(\mathcal{G}_i) \leq 500 \tag{5.7}$$

  This means that each graph is divided into a number of subgraphs, such that any subgraph has a global state vector whose dimension does not exceed 500. This condition was chosen arbitrarily, in order to study the results obtained with a given $CC + GSD$ strategy. In practice, this divides each graph between 2 and 3 subgraphs during the following experiments.

To assess the accuracy of the maps built in the different configurations, using each of the three map update strategies, we compute their average distance errors after each vehicle passage. In Table 5.2, we show the average distance error obtained after the last vehicle passage in each case. For a given number of landmarks, the lowest distance error, indicating the best-performing map update strategy, is highlighted in blue. In Figure 5.4, we also depict

	Distance Errors (m)			Computation Time by Passage (s)		
	CC	DC	$CC + GSD$	CC	DC	$CC + GSD$
4 Landmarks	0.29	0.31	0.30	4.90	4.77	4.31
25 Landmarks	0.72	1.10	0.71	5.44	5.51	4.78
50 Landmarks	0.51	1.63	0.51	5.90	5.92	5.22
75 Landmarks	0.41	1.86	0.41	6.51	6.63	5.82
100 Landmarks	0.31	2.13	0.33	7.02	7.32	5.87

Table 5.2: Average distance errors after the last vehicle passage and average computation time by vehicle passage, obtained using white and Gaussian noises during simulation experiments.

the evolution of the average distance errors obtained for the configurations with $N = 4$ landmarks and $N = 100$ landmarks. In fact, results obtained for the configurations with $N = 25$, $N = 50$ and $N = 75$ landmarks are similar to those obtained for the configuration with $N = 100$ landmarks, and are therefore not displayed.

Generally, a map is considered consistent when the estimated covariance is not smaller than the true covariance [108]. Unfortunately, this cannot be checked as the true covariance of estimations is not available. Instead, to evaluate the consistency of a built map, we compute after each vehicle passage the errors and 3σ deviation ranges obtained on the East and North axes for each landmark. Considering such errors follow a chi-square distribution with 1 degree of freedom, their respective 3σ deviation ranges correspond to a $99.7\ \%$ confidence interval [109]. Thus, we can consider that a given map is likely consistent (resp. inconsistent) if the average errors on the East and North axes contain (resp. do not contain) the ground truth (null error) within their 3σ deviation ranges. In Figure 5.5, we depict the evolution of the average errors on the East and North axes, along with their respective standard deviations as $[-3\sigma, 3\sigma]$ hashed ranges. Similarly as before, we only show results for the configurations with $N = 4$ and $N = 100$ landmarks.

For the sake of clarification, in Figure 5.4, we show the average of distance errors, with the purpose of evaluating the map accuracy. Distance errors can only be positive and can not compensate each other when computing the average. Meanwhile, in Figure 5.5, we depict

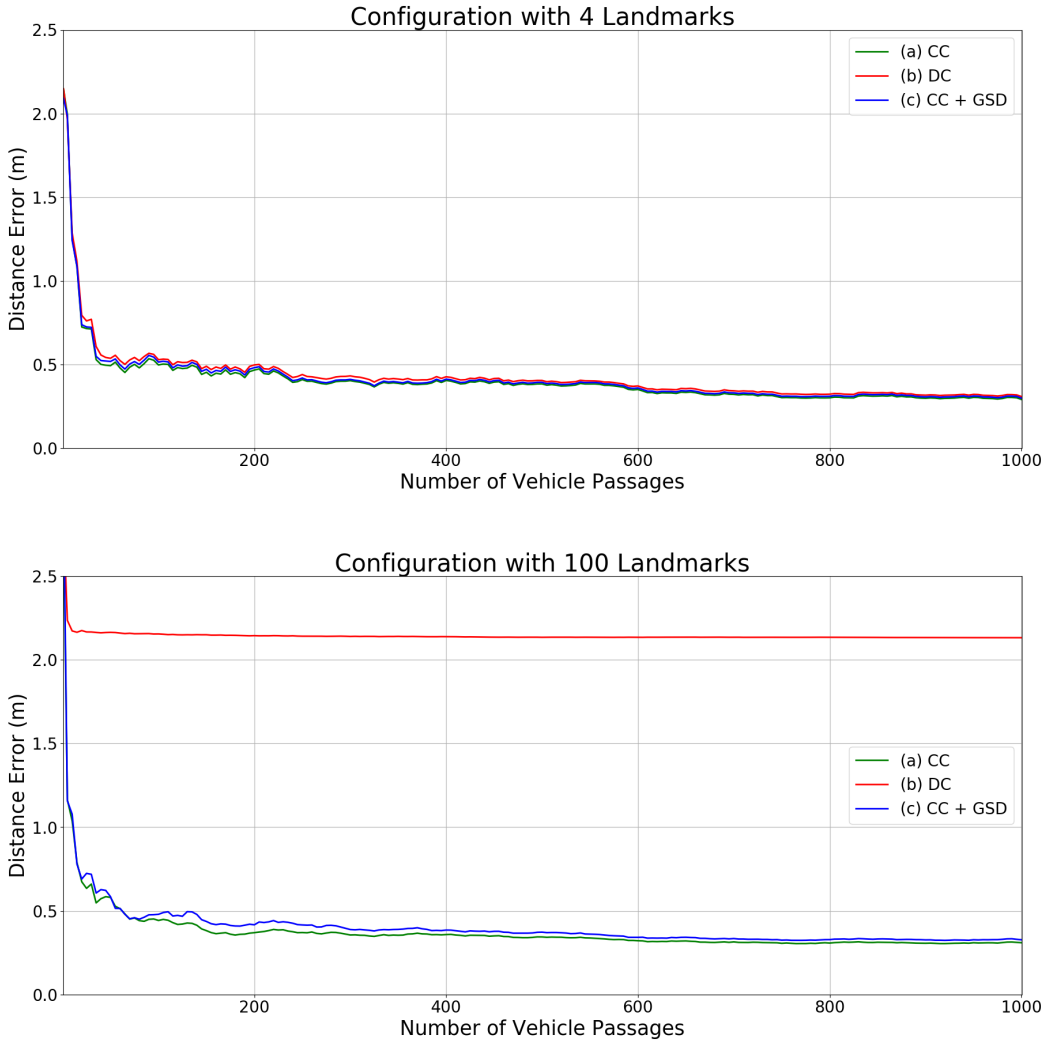

Figure 5.4: Evolution of average distance errors obtained using white and Gaussian noises during simulation experiments.

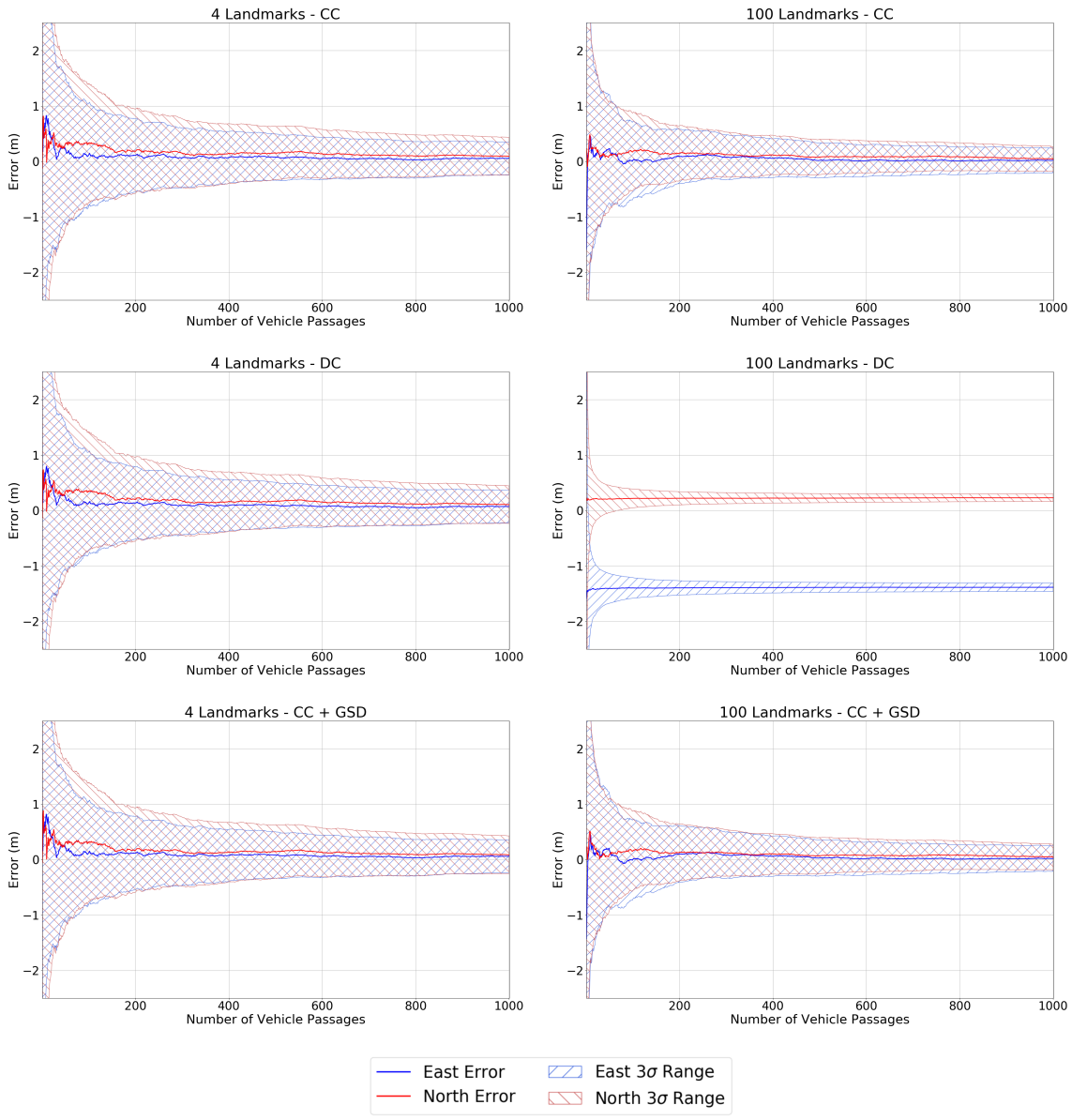

Figure 5.5: Evolution of average errors on the East and North axes, along with their respective 3σ deviation ranges, obtained using white and Gaussian noises during simulation experiments.

the average of East and North errors, with the aim of studying the map consistency. East and North errors can either be positive or negative, and can compensate each other when computing the average. Therefore, it is perfectly normal that the amplitude of the average distance error (see Figure 5.4) is higher than the amplitude of average East and North errors (see Figure 5.5).

To assess the scalability potential of each strategy, we retrieve the total computation time required to process the 1000 vehicle passages for each configuration. Then, we divide this time by 1000, in order to retrieve the average computation time by vehicle passage, which is shown in Table 5.2 for each case. Similarly as for distance errors, we highlight in blue the lowest computation time for a given number of landmarks. While computation times may appear extensive, they are much lower than the average driving time of a vehicle passage, which is approximately $165\ s$.

Discussion In all the studied strategies, the average distance errors decrease with the number of vehicle passages (see Figure 5.4). Nevertheless, they tend to converge towards an offset value, which is likely due to the linearization of non-linear motion and sensors models operated during graph optimization (see Algorithm 1).

In the configuration with $N = 4$ landmarks, all methods lead to similar distance errors (see Table 5.2). However, in all other configurations, the distance errors obtained with the DC approach remain high. As expected, this corresponds to the fact that the maps built with the DC approach in such configurations tend to be highly inconsistent, with average errors on the East and North axes clearly not containing the ground truth within their 3σ deviation ranges (see Figure 5.5). This shows that neglecting cross-correlations between landmarks estimations may be valid when landmarks are far from each other, such as in the configuration with $N = 4$ landmarks, but it is clearly not valid in general. Furthermore, the DC approach does not even provide much computational efficiency, as compared with the CC approach (see Table 5.2). This is likely due to the fact that using an inconsistent map as

prior knowledge slows down graph optimization.

Oppositely, the CC strategy is able to provide accurate maps in all configurations, with distance errors reaching between $0.29\ m$ and $0.72\ m$ after the last vehicle passage (see Table 5.2). The relationship between map accuracy and consistency is also confirmed, as the maps built by the CC approach appear to remain consistent, with average errors on the East and North axes always containing the ground truth within their 3σ deviation ranges (see Figure 5.5). Furthermore, the CC approach presents computation times that reach between $4.90\ s$ and $7.02\ s$ on average by vehicle passage (see Table 5.2), which is still very reasonable when compared to the average driving time of $165\ s$.

Finally, the $CC + GSD$ strategy is able to build maps that have a similar accuracy as the CC approach, while requiring much less computations (see Table 5.2). The $CC + GSD$ strategy appears as computationally scalable, as the gap between the computational costs of the $CC + GSD$ and CC strategies tends to grow with the number of landmarks. The distance errors obtained with the $CC + GSD$ strategy are slightly higher than those obtained with the CC approach, which corresponds to the loss of information created by the graph subdivisions, as explained in paragraph 4.4.3. Nevertheless, as the $CC + GSD$ strategy does not make any approximation, the built maps remain consistent (see Figure 5.5), and their accuracy continues to improve with the number of vehicle passages. This confirms that the use of graph subdivisions is an effective method that enables to limit the amount of computations while maintaining an accurate and consistent map.

Overview of Scalability Assessment In order to assess the scalability of our crowd-sourced mapping solution, we performed simulation experiments using configurations that contained between $N = 4$ and $N = 100$ landmarks. We compared the performances reached by each of the proposed map update strategies: CC, DC and $CC + GSD$. We aimed to evaluate their capacity to build accurate and consistent maps, independently from the number of considered landmarks, while remaining computationally scalable.

The DC approach, which neglects cross-correlations between landmarks estimations, provides over-confident map updates that lead to inconsistent maps, whose errors remain high even after multiple vehicle passages. Oppositely, the CC approach, which does not neglect any cross-correlation, is able to build consistent and accurate maps of landmarks. Furthermore, its computation time remains reasonable in comparison with the driving time of the vehicles. Finally, the $CC + GSD$ strategy is able to reach similar performances than the CC approach, while requiring much less computations. This shows that making use of graph subdivisions is an effective way to lower computations while maintaining accuracy and consistency of the maps, and makes the $CC + GSD$ strategy appear as the most efficient method to provide a scalable solution for crowdsourced mapping.

5.2.3 Robustness Assessment

To assess the robustness of our solution, we aim to evaluate its ability to build an accurate and consistent map, even in the presence of various types of noises such as GNSS auto-correlated noises and camera calibration bias, which do not correspond to the basic assumption used by graph optimization (see subsection 4.3.2) but can be present in real conditions. We consider a single landmarks configuration: the one with $N = 50$ landmarks, as this corresponds to the approximate number of traffic signs along the real path, and therefore represents a realistic situation, in which we define 4 noises configurations that correspond to 4 different types of noises affecting the sensors measurements. In fact, in the following, the term "configuration" will not refer to the number of landmarks anymore, but rather to the given types of noises. The first configuration corresponds to the case defined in paragraph 5.2.2, in which noises are sampled from white and Gaussian distributions, and serves as a reference to discuss the obtained results. The other configurations correspond to more realistic cases, and are presented in the following.

GNSS Auto-Correlated Noises GNSS measurements are usually affected by auto-correlated noises that originate from the signals delaying when travelling atmospheric layers, and the signals reflectance created by multi-path effects. Auto-correlated noises are prone to create biased estimations, and can be complex to model, especially in the case of multi-path effects. Therefore, it is essential to assess the impact that such noises can have on the performances of our crowdsourced mapping solution.

We define a second configuration, in which the displacement measurements U_k and detection measurements D_k are generated similarly as in the first configuration (see paragraph 5.2.2), with noises sampled from white and Gaussian distributions. Meanwhile, new position measurements Y_k are generated, and affected by typical GNSS auto-correlated noises. The authors of [110] showed that such noises can be modeled efficiently using an AR (Auto-Regressive) model of first order. Thus, for each instant k corresponding to the GNSS receiver frequency f_y, a measurement Y_k affected by a noise $\epsilon_{y,k}$ is generated with:

$$\epsilon_{y,k} = \alpha\, \epsilon_{y,k-1} + \beta\, \epsilon_b \tag{5.8}$$

$\epsilon_{y,k-1}$ is the GNSS noise at the previous instant, and ϵ_b is sampled from a zero-mean, Gaussian distribution with covariance Σ_b. The coefficients α and β are two scalars that model the amplitude of GNSS auto-correlated noises, and that need to be estimated. In the following, we briefly summarize the method used to estimate α and β, and refer the reader to Appendix A for more details.

To generate realistic auto-correlated noises, we performed a field experiment using another test-vehicle than the one used within our field-tests, which was equipped with both a standard GNSS receiver and an RTK-GPS receiver. Making use of the measurements provided by the RTK-GPS receiver as ground truth positions, we could infer the auto-correlation function associated with measurements from the standard GNSS receiver. This function is depicted in blue in Figure 5.6 and represents a typical auto-correlation function associated with measurements from a standard GNSS receiver in a dense environment.

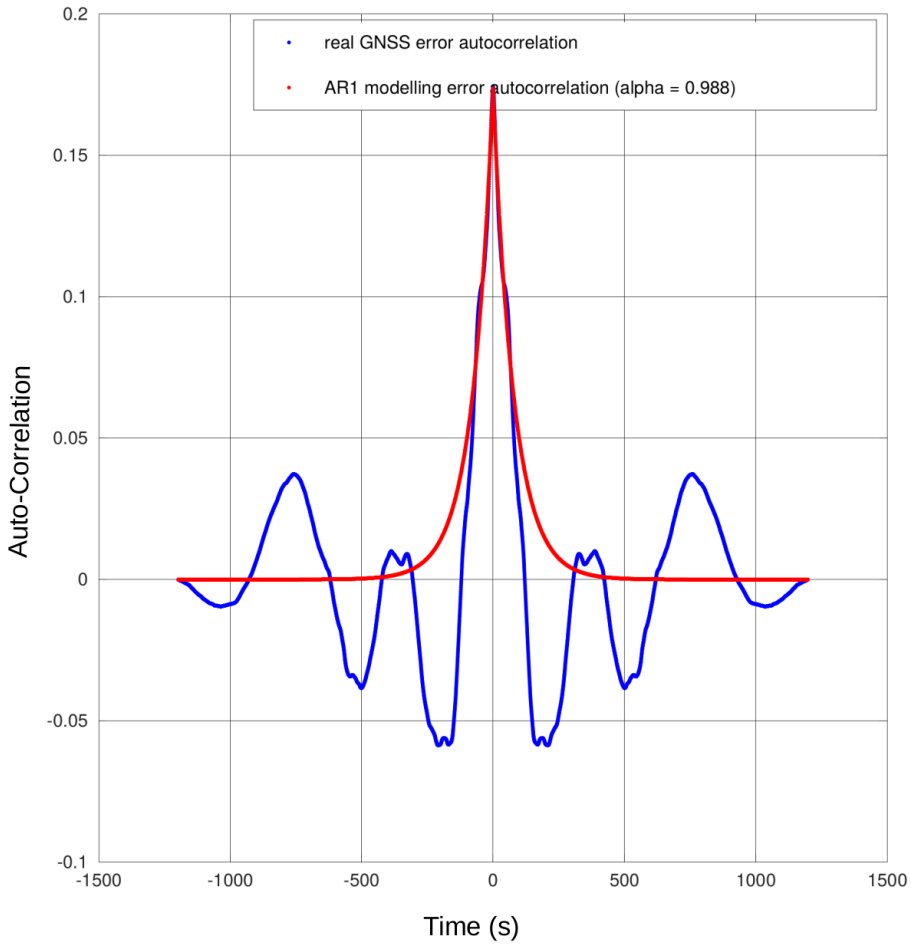

Figure 5.6: Auto-correlation function of GNSS measurements obtained using our test-vehicle in a dense environment.

Making use of this function, we could compute the value of α as indicated in Appendix A.

In order to properly study the impact of auto-correlated noises, we need to apply noises that have a similar covariance as in other configurations. Thus, we impose the noise $\epsilon_{y,k}$ to have the same covariance matrix as ϵ_b:

$$\Sigma_b = \Sigma_y = \begin{pmatrix} \delta_{\bar{x}}^2 & 0.0 \\ 0.0 & \delta_{\bar{y}}^2 \end{pmatrix} \tag{5.9}$$

α	β	$\delta_{\bar{x}}\,(m)$	$\delta_{\bar{y}}\,(m)$
0.988	0.15	10.0	10.0

Table 5.3: GNSS auto-regressive model parameters and noises standard deviations used to generate GNSS auto-correlated noises during simulation experiments.

To satisfy this condition, as detailed in Appendix A, β is computed as:

$$\beta = \sqrt{1 - \alpha^2} \tag{5.10}$$

The parameters used to generate GNSS auto-correlated noises are summarized in Table 5.3.

Camera Calibration Bias Another typical source of systematic bias in vehicular applications are calibration errors. Especially, errors associated with the sensors orientations are known to have a more important effect on the perceived measurements than errors associated with the sensors positions. In our setup, the only sensor potentially affected by such orientation errors is the monocular camera, whose orientation on the vehicle can be expressed as a set of 3 angles: roll, pitch and yaw. Calibration errors associated with roll and pitch angles can be neglected, as we consider a 2D simulation on the North-East plane, and an extrinsic calibration that aligns the camera with this plane. Oppositely, the calibration error associated with the yaw angle has to be considered, as it can have a substantial effect on the detection measurements.

We define a third configuration, in which the displacement measurements U_k and position measurements Y_k are generated similarly as in the first configuration (see paragraph 5.2.2), with noises sampled from white and Gaussian distributions. Meanwhile, new detection measurements D_k are generated, considering that the extrinsic calibration of the monocular camera is erroneous. In order to apply a calibration error of realistic amplitude, we refer to

advanced calibration methods [111], which can typically reach orientation errors around:

$$e_{yaw} = 0.009\,rd \qquad (\sim 0.5°) \tag{5.11}$$

Thus, when generating the detection measurements D_k, we apply a calibration error e_{yaw} on the yaw angle of the camera.

GNSS Auto-Correlated Noises and Camera Calibration Bias In the fourth configuration, we want to combine the effect of GNSS auto-correlated noises and camera calibration bias. Thus, we generate displacement measurements U_k as in the first configuration (see paragraph 5.2.2), with noises sampled from white and Gaussian distributions. We generate position measurements Y_k as in the second configuration (see paragraph 5.2.3), with GNSS auto-correlated noises. And we generate detection measurements D_k as in the third configuration (see paragraph 5.2.3), considering an extrinsic calibration error on the yaw angle of the camera.

Results For each configuration, we process the 1000 vehicle passages as follows. As a given vehicle reaches the end of the trajectory, it shares its observations with a centralized server. The server then performs graph optimization using the $CC + GSD$ strategy with the dimension condition defined in Equation 5.7, as this method was found to be the best map update strategy during previous simulation experiments (see subsection 5.2.2). We apply the exact same algorithm as before, assuming that all measurements are affected by noises sampled from white and Gaussian distributions, even when it is not the case, such as in the presence of GNSS auto-correlated noises and camera calibration bias. For a given passage, the complete graph contains around 165 position constraints, as well as 5 detection constraints for each landmark, while displacement constraints link vehicle nodes together. As a result, the typical graph ends up being subdivided into 2 subgraphs. This time, we run the mapping process on a laptop with an Intel i5-6200U CPU.

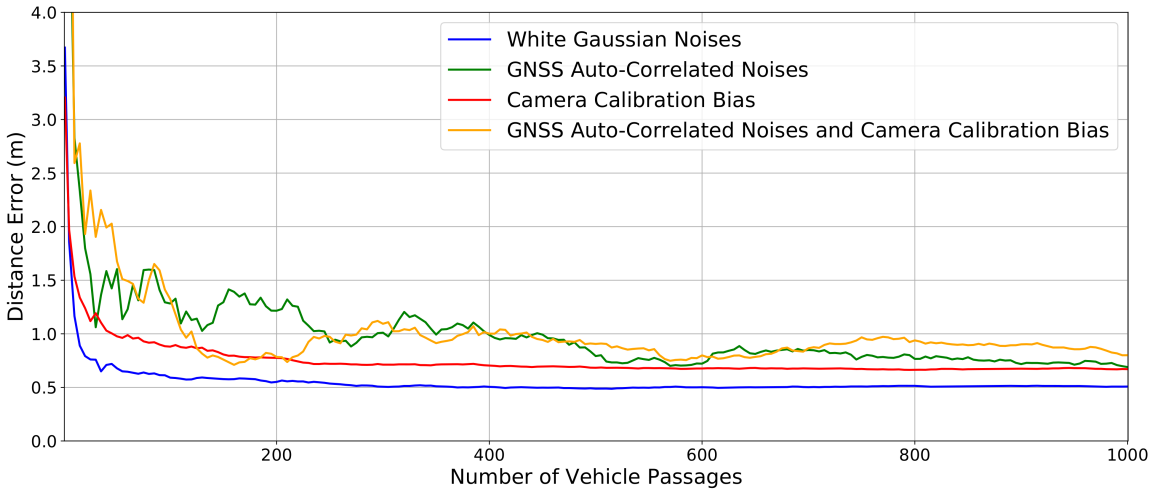

Figure 5.7: Evolution of average distance errors obtained using different types of noises during simulation experiments.

In Figure 5.7, we depict the evolution of the average distance errors obtained for each configuration, while in Figure 5.8, we show the evolution of the average errors on the East and North axes, along with their respective standard deviations depicted as $[-3\sigma, 3\sigma]$ hashed ranges.

Discussion In all configurations, the average distance error decreases with the number of vehicle passages (see Figure 5.7). As during previous simulation experiments, it also tends to converge towards an offset value. While this can be explained in part by the linearization of motion and sensors models during graph optimization (see Algorithm 1), this also appears to be affected by the presence of GNSS auto-correlated noises and camera calibration bias. Indeed, the distance error after the last vehicle passage reaches $0.51\ m$ when noises are sampled from white and Gaussian distributions. In the meanwhile, it reaches higher values, i.e. $0.67\ m$, $0.69\ m$ and $0.80\ m$ when noises are respectively affected by camera calibration bias, GNSS auto-correlated noises or both of these effects.

Considering the shape of the evolution of distance errors, we can distinguish two different behaviors, that depend on whether GNSS measurements are affected or not by auto-correlated noises. In the first and third configurations, in which GNSS measurements

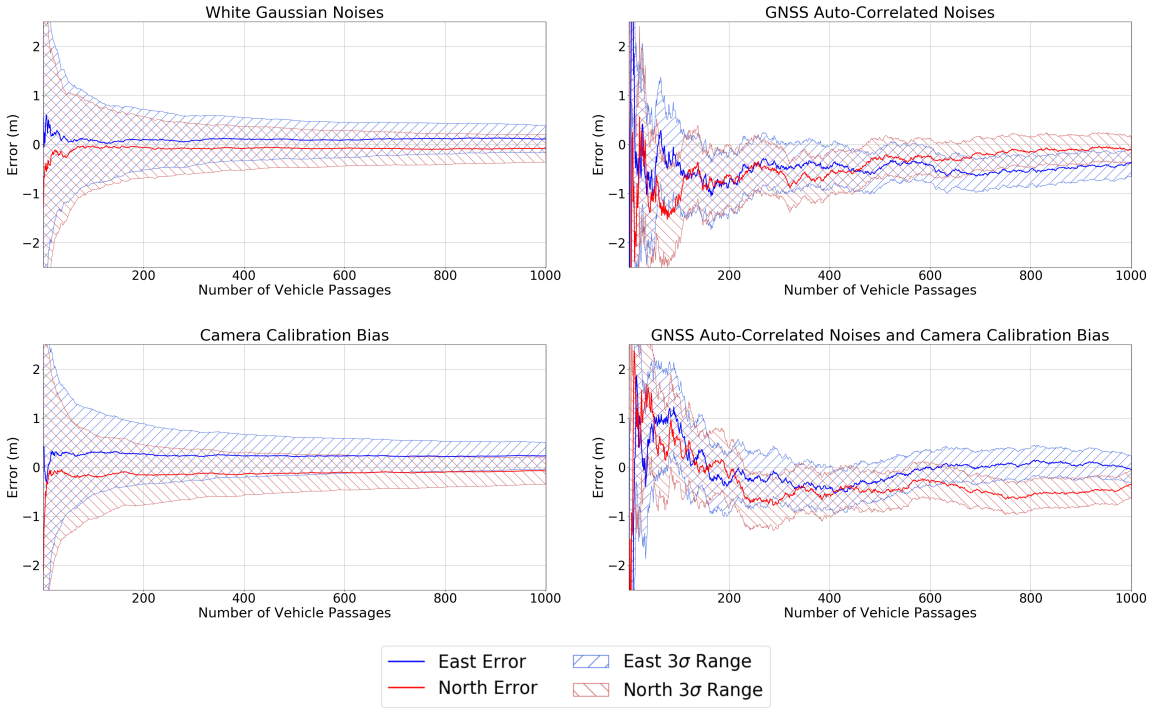

Figure 5.8: Evolution of average errors on the East and North axes, along with their respective 3σ deviation ranges, obtained using different types of noises during simulation experiments.

are not affected by auto-correlated noises, the distance errors follow a regular and stable evolution similar to an inverse square function. This corresponds to the expected evolution of the distance error for these configurations, as their errors on the East and North axes always contain the ground truth within their 3σ deviation ranges (see Figure 5.8), indicating that the built maps always remain consistent.

Oppositely, in the second and fourth configurations, in which GNSS measurements are affected by auto-correlated noises, distance errors follow a more volatile behavior (see Figure 5.7). Furthermore, it can be observed that the errors on the East and North axes for these configurations do not always contain the ground truth within their 3σ deviation ranges (see Figure 5.8), which indicates that the built maps do not always remain consistent. Nevertheless, as the number of vehicle passages increases, the 3σ deviation ranges tend to contain the ground truth again. This can be explained by the fact that, in these two configurations, the noises affecting GNSS measurements are auto-correlated within each vehicle passage, but not between the different vehicle passages. Thus, by making use of multiple vehicle passages, crowdsourced mapping is able to alleviate the effect of GNSS auto-correlated noises and build an accurate map of landmarks.

Overall, in all configurations, the distance errors decrease substantially within the first vehicle passages, and then decrease more slowly. In the first and third configurations, without GNSS auto-correlated noises, only the first few dozens of vehicle passages are able to bring important accuracy improvements. Meanwhile, in the second and fourth configurations, with GNSS auto-correlated noises, this is the case for the first few hundreds of vehicle passages. This indicates that, in the presence of GNSS auto-correlated noises, which is usually the case in real conditions, a higher number of vehicle passages within a crowdsourcing approach is beneficial towards the establishment of an accurate map of landmarks.

Overview of Robustness Assessment In order to evaluate the robustness of our crowdsourced mapping solution, we performed simulation experiments with various types of

noises, such as GNSS auto-correlated noises and camera calibration bias. We performed graph optimization using the $CC + GSD$ strategy, and aimed to evaluate the accuracy and consistency of the built maps.

For all configurations, the accuracy of the map increases with the number of vehicle passages. When GNSS measurements are not affected by auto-correlated noises, the map always remains consistent, and the evolution of its error follows a stable decrease towards an offset value. Oppositely, in the presence of GNSS auto-correlated noises, the map is not always consistent, resulting in an evolution of its error that is more volatile. Nevertheless, by making use of more vehicle passages, crowdsourced mapping is able to address this issue and build an accurate map of landmarks.

5.3 Graph-based Approach: Evaluation through Field-Tests

To corroborate simulation results, we evaluate the performances of our crowdsourced mapping solution in real conditions. We drive our test-vehicle for multiple passages along the same loop as during early field-tests (see section 5.1), which is situated within the city center of Versailles, France. We aim to assess the capacity of crowdsourced mapping to build an accurate map of landmarks, while remaining computationally scalable. Furthermore, we compare the obtained results with those of previous simulation experiments, in order to validate the simulation setup, and support the conclusions drawn from simulation results.

5.3.1 Field Setup

Similarly as before, we perform the field-tests in 2D, and project all positions to a local North-East plane. We drive our test-vehicle along the loop of around $4\ km$ for a total of 40 passages. Again, the 40 passages were registered through numerous driving sessions, and properly represent 40 different vehicles. To provide insight on the trajectory, geo-positions retrieved during the first passage by the embedded GNSS receiver are shown in Figure 5.9. As can be seen in the zoomed section of the trajectory (see Figure 5.9 (b)),

GNSS measurements can have large errors that reach $\sim 20\,m$ in some cases. Furthermore, these errors appear to be directed in the same direction, when observing a short part of the trajectory. This validates the considered standard deviations $\delta_{\bar{x}}$ and $\delta_{\bar{y}}$ (see Table 5.1), as well as the fact that GNSS measurements are affected by auto-correlated noises in reality.

Along the loop, traffic signs were identified, and used as landmarks constituting the map. We manually measured their geo-positions using an RTK-GPS receiver to provide a ground truth to compare our results with. In order to evaluate the performances of our solution independently from the influence of image processing techniques, we manually operated, at a frequency $f_d = 2.0\,Hz$, the detection and recognition of traffic signs within images provided by the embedded camera. Due to this manual process, we used a limited number of landmarks: $M = L_1, ..., L_5$, whose ground truth geo-positions are shown in Figure 5.9, along with images of the corresponding traffic signs taken from the camera.

5.3.2 Evaluation of Mapping Performances

To assess the performances of our crowdsourced mapping solution in real conditions, we aim to evaluate its capacity to build an accurate and consistent map, making use of measurements retrieved by our test-vehicle, while remaining computationally scalable. As the test-vehicle reaches the end of a passage, it shares its observations with a centralized server. The latter then performs graph optimization using the $CC + GSD$ strategy with the dimension condition defined in Equation 5.7, in order to build and update the map. During graph optimization, all measurements are assumed to be affected by noises sampled from white and Gaussian distributions with the same covariances as in the simulation (see Table 5.1). For a given passage, the complete graph contains around 583 position constraints, which is higher than during simulation experiments, as we now consider the full loop of around $4\,km$. The graph also contains 5 detection constraints for each landmark, and displacement constraints that link vehicle nodes together. Consequently, it is typically subdivided into 4 or 5 subgraphs, depending on the vehicle passage. As previously, this mapping process was

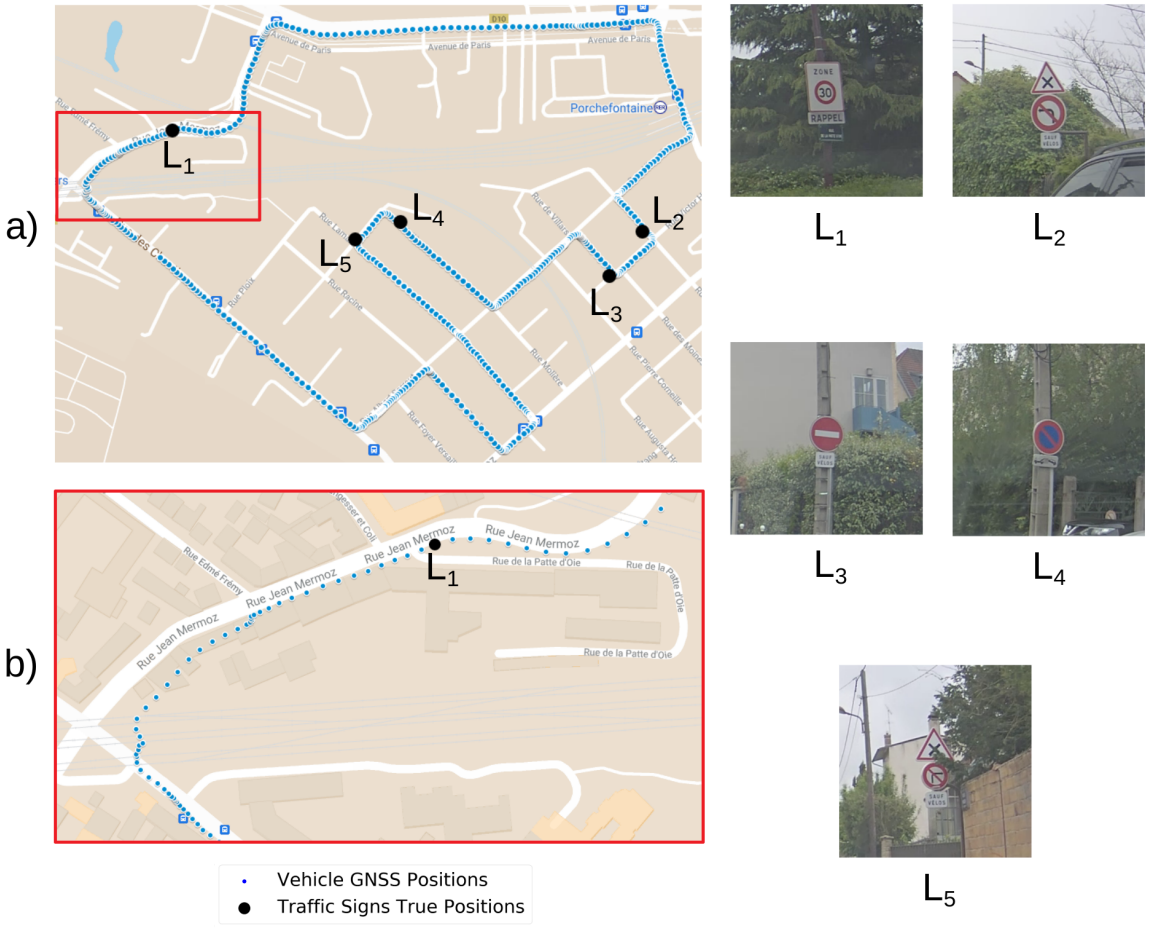

Figure 5.9: Vehicle GNSS positions at first passage and traffic signs ground truth positions during field-tests.

implemented in Python, and run on a laptop with an Intel i5-6200U CPU.

Results In Figure 5.10, we depict the evolution of errors obtained for each traffic sign, as well as for the average over all traffic signs. On the left, errors on the East and North axes are shown as thick lines, along with their respective standard deviations shown as $[-3\sigma, 3\sigma]$ hashed ranges. Meanwhile, on the right, distance errors are depicted. The average computation time needed to process a given vehicle passage was around $17\ s$. This does not include the detection process, which was done manually in our case, and could be run on-board the vehicles in real-time. In comparison, the average driving time of a vehicle passage was around $708\,s$, which confirms that the use of the $CC + GSD$ strategy represents a scalable solution for crowdsourced mapping.

Discussion During our field-tests, we made use of 40 vehicle passages, which makes our experiment more complete and realistic than related previous works (see Table 3.1). For most traffic signs, the distance error tends to decrease with the number of vehicle passages (see Figure 5.10), and the average distance error for all traffic signs goes from $4.02\ m$ after the first passage to $1.90\ m$ after the last passage. For comparison purposes, in dense environments, standard GNSS receivers typically suffer from larger positioning errors, which can reach up to a few dozens of meters. In fact, such positioning errors could clearly be observed during our field-tests (see Figure 5.9 (b)).

For the traffic sign L_3, the errors on the East and North axes always contain the ground truth within their 3σ deviation ranges (see Figure 5.10). This indicates that estimations of this traffic sign remain consistent. However, this is not verified for estimations of other traffic signs. As confirmed by previous simulation experiments, this is likely due to the presence of auto-correlated noises affecting GNSS measurements, which can lead to over-confident map updates. Nevertheless, simulation experiments also showed that crowdsourced mapping can generally deal with this issue by using measurements retrieved during a greater number of vehicle passages. Furthermore, at the end of the 40^{th} passage, estimations for traffic signs

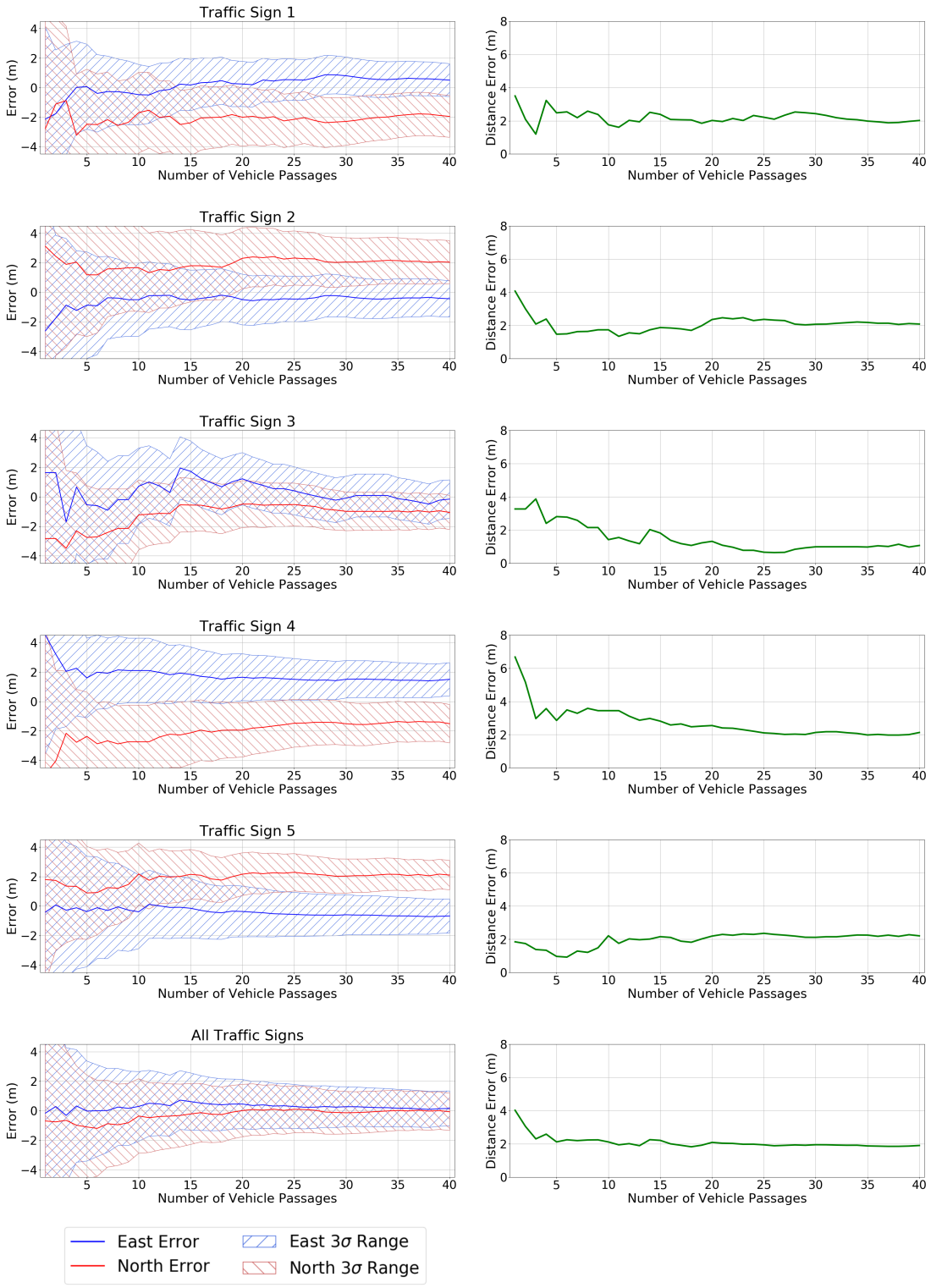

Figure 5.10: Evolution of errors for the positioning of traffic signs L_1, L_2, L_3, L_4 and L_5, and for the average over all traffic signs, obtained during field-tests.

L_1, L_2, L_4 and L_5 reach respective distance errors of $2.02\ m$, $2.08\ m$, $2.13\ m$ and $2.20\ m$, which is already accurate considering the use of standard sensors.

The average distance error obtained after the 40^{th} vehicle passage during these field-tests is $1.90\ m$. This is comparable with the average distance error of $1.99\ m$ obtained after the 40^{th} vehicle passage during simulation experiments, considering the most realistic configuration which combines both GNSS auto-correlated noises and camera calibration bias (see Figure 5.7). This confirms that simulation experiments correspond to realistic conditions, and therefore that simulation results provide useful conclusions regarding the performances of crowdsourced mapping.

5.4 Evaluation of Contributions for Vehicles Positioning

One of the main motivations for building a map is to support the localization of vehicles. Therefore, we would like to conclude this chapter by evaluating the contribution of the proposed crowdsourced mapping solution for vehicles positioning. Since the data used during field experiments contains only a small portion of the traffic signs, we rely on simulation to assess such contribution. To this end, a simulation setup based on realistic data and landmarks distribution is considered. A map with a similar accuracy and consistency as the one built during field-tests is generated, and used for localization purposes. Based on this setup, different configurations with various numbers of landmarks and different types of noises are considered, and enable to draw conclusions on the capacity of our crowdsourced mapping solution to improve the localization performances of the vehicles, by providing a map which can be used to support their positioning.

5.4.1 Simulation Setup

As in previous simulation experiments, the simulation was implemented in C++, and the experiments were performed in 2D by projecting all positions in a local, horizontal plane. However, oppositely to previous simulation experiments, we now generate the vehicle

trajectory by considering the complete loop of around $4\,km$ followed by our test-vehicle in real situation. The ground truth vehicle trajectory is depicted in Figure 5.11.

The static landmarks are generated following 2 distributions. The first one includes $N = 44$ landmarks which correspond to 44 traffic signs identified on the real path, and whose geo-positions are shown in Figure 5.11. Nevertheless, this remains a small portion of all the traffic signs that are present in real conditions and that could be detected. Thus, this distribution does not allow to assess the real impact of the map on vehicles positioning. Furthermore, while we previously restricted the built maps to contain traffic signs for simplification purposes, other types of landmarks such as traffic lights and road markings would generally be considered for building the map. For the second distribution, we want to include a realistic density of landmarks in the map. To do so, we refer to the work of [112], which built a map of pole-shaped landmarks (e.g. traffic signs, street lights), and found that there were approximately 1 such landmark every $8\,m$ in a city center. Therefore, we build a dense landmarks distribution with $N = 500$ landmarks, which corresponds on average to 1 landmark every $8\,m$ along the $4\,km$ loop. Among these 500 landmarks, whose geo-positions are shown in Figure 5.11, 44 landmarks correspond to the real traffic signs introduced previously, and 456 landmarks are randomly generated through a uniform law. Such uniform law is well-adapted to the current distribution, as we now consider that any pole-shaped landmark can be used, including street lights which can be detected all along the vehicle trajectory.

The vehicle is assumed to be equipped with motor encoders, a GNSS receiver and a monocular camera, which are installed with the same extrinsic calibration, and operated at similar frequencies, as on our test-vehicle. To generate measurements, we consider 5 different noises configurations that are defined in the following.

White Gaussian Noises In the first configuration, we consider the case defined in paragraph 5.2.2, in which noises are sampled from white and Gaussian distributions, with

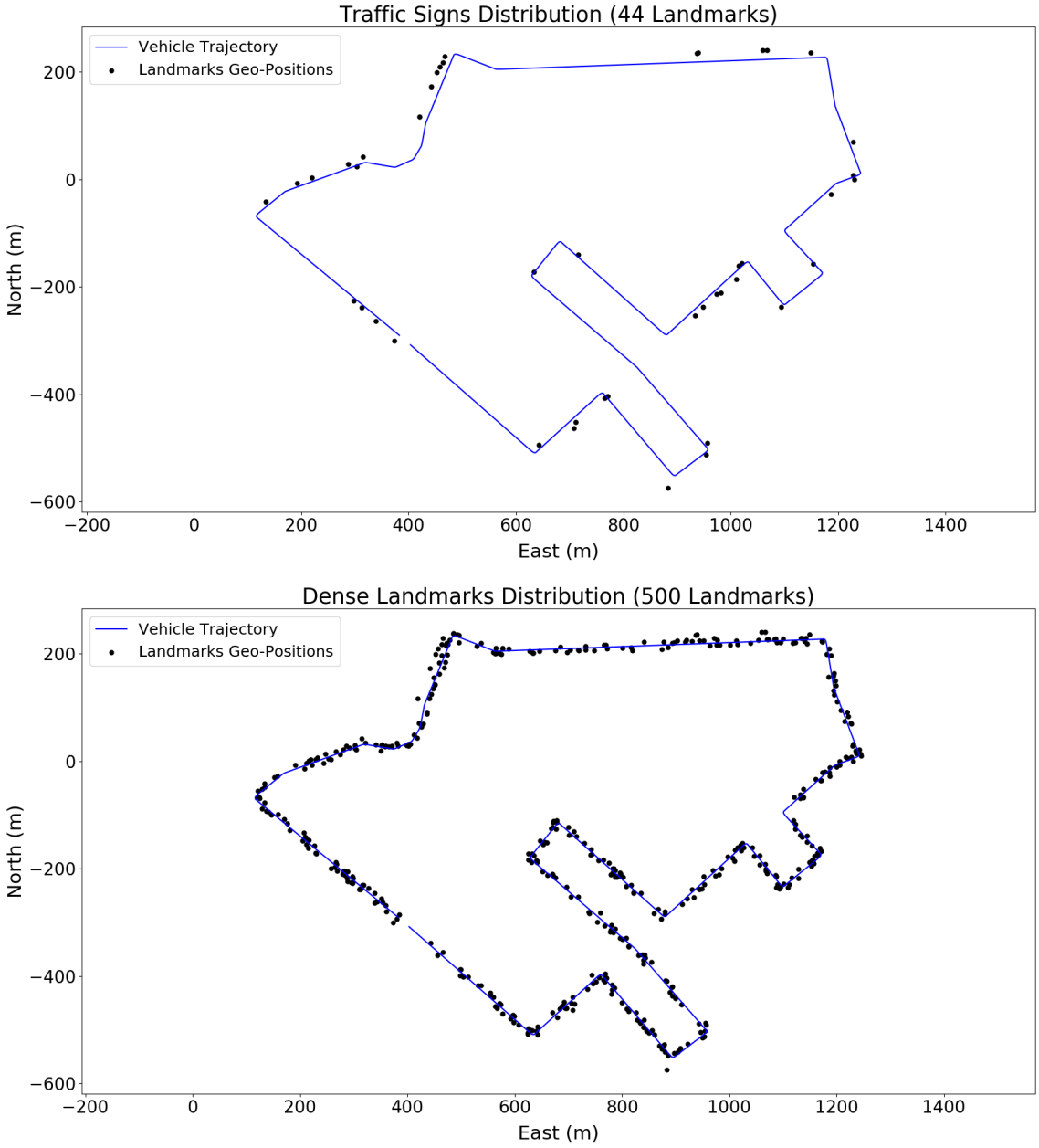

Figure 5.11: Ground truth vehicle trajectory and landmarks geo-positions used in simulation evaluation of crowdsourced mapping contributions for localization.

standard deviations given in Table 5.1. This configuration represents an optimistic setup, and will serve as a reference to discuss the obtained results.

In order to evaluate the contributions of our approach for vehicles positioning, we aim to generate a map and use it to support the localization of the vehicle. This map must possess an accuracy that is typical of the one provided by our crowdsourced mapping solution, using as reference the accuracy obtained during previous field experiments (see section 5.3). Thus, we generate a map $\{\hat{M}, \Sigma_M\}$, where $\hat{M}$ represents the estimated landmarks geo-positions, and Σ_M corresponds to their covariance, as follows:

- $\hat{M}$ is formed by the list of geo-position estimations of all landmarks in the considered distribution $M = \{L_1, ..., L_N\}$. For any landmark L_i, we generate its geo-position estimation $\hat{L}_i$ as:

$$\hat{L}_i = L_i + \epsilon_{l,i} \tag{5.12}$$

where L_i corresponds to the ground truth geo-position, and $\epsilon_{l,i}$ is the estimation noise. $\epsilon_{l,i}$ is sampled from a Gaussian distribution with mean μ_l and covariance Σ_l:

$$\mu_l = \begin{pmatrix} \mu_{l,e} \\ \mu_{l,n} \end{pmatrix} \tag{5.13}$$

$$\Sigma_l = \begin{pmatrix} \delta_{l,e}^2 & 0.0 \\ 0.0 & \delta_{l,n}^2 \end{pmatrix} \tag{5.14}$$

The values of $\mu_{l,e}$ and $\mu_{l,n}$ correspond respectively to the average errors on the East and North axes of the map obtained after the last vehicle passage during field experiments. Meanwhile, $\delta_{l,e}$ and $\delta_{l,n}$ correspond respectively to the standard deviations of these errors on the East and North axes. These values are summarized in Table 5.4.

- Σ_M is a block-diagonal matrix, in which each block corresponds to the covariance Σ_l used to generate estimation noises:

$$\Sigma_M = \begin{pmatrix} \Sigma_l & 0 & ... \\ 0 & \Sigma_l & ... \\ ... & ... & ... \end{pmatrix} \tag{5.15}$$

GNSS Auto-Correlated Noises During previous simulation experiments (see subsection 5.2.3), we studied the effect of different types of noises, including GNSS auto-correlated noises and camera calibration bias, on the mapping performances of our solution. In the following, we aim to evaluate the contribution of our approach for vehicles positioning in the presence of such types of noises.

To begin with, we consider the presence of GNSS auto-correlated noises. We define a second configuration in which position measurements are affected by auto-correlated noises as defined in paragraph 5.2.3, and we make use of the map previously generated (see paragraph 5.4.1) to support the localization of the vehicle.

Camera Calibration Bias Next, we follow by considering the presence of camera calibration bias. We define a third configuration in which detection measurements are affected by camera calibration bias as defined in paragraph 5.2.3, and we make use of the map previously generated (see paragraph 5.4.1) to support the localization of the vehicle.

GNSS Auto-Correlated Noises and Camera Calibration Bias Then, we consider the presence of both GNSS auto-correlated noises and camera calibration bias, which is generally the case in real situation. We define a fourth configuration in which position measurements are affected by auto-correlated noises, while detection measurements are affected by camera calibration bias, as defined in paragraph 5.2.3. Again, we make use of the map previously generated (see paragraph 5.4.1) to support the localization of the vehicle.

GNSS Auto-Correlated Noises, Camera Calibration Bias and Map Inconsistency In all configurations defined until now, we considered a consistent map, i.e. a map with a valid covariance Σ_M, to support the localization of the vehicle. Nevertheless, during previous field experiments (see section 5.3), we observed that the map built by our crowdsourced mapping solution tends to become over-confident, leading to an invalid covariance Σ_M, and concluded that it was mostly due to the presence of auto-correlated noises affecting GNSS measurements.

To study the effect that an inconsistent map can have on localization performances, we aim to generate and make use of another map $\{\hat{M}, \Sigma_M\}$, whose both accuracy and consistency are similar to those obtained during field-tests. Thus, we aim to build a map which contains not only noisy landmarks geo-positions $\hat{M}$, but also a noisy covariance Σ_M which does not properly reflect the distribution of estimation noises affecting landmarks geo-positions. This corresponds to the fact that auto-correlated and biased noises are not modeled through graph optimization, and would lead to over-confident map updates in reality.

We define a fifth configuration in which position measurements and detection measurements are respectively affected by auto-correlated noises and camera calibration bias, as defined in paragraph 5.2.3. Meanwhile, to support the localization of the vehicle, we make use of a new map $\{\hat{M}, \Sigma_M\}$ that is generated as follows:

- $\hat{M}$ is formed by the list of geo-position estimations of all landmarks, and is built similarly as for the previous map (see paragraph 5.4.1).

- Σ_M is a block-diagonal matrix:

$$\Sigma_M = \begin{pmatrix} \Sigma_m & 0 & ... \\ 0 & \Sigma_m & ... \\ ... & ... & ... \end{pmatrix} \tag{5.16}$$

$\mu_{l,e}\ (m)$	$\mu_{l,n}\ (m)$	$\delta_{l,e}\ (m)$	$\delta_{l,n}\ (m)$	$\delta_{m,e}\ (m)$	$\delta_{m,n}\ (m)$
0.15	-0.08	0.78	1.77	0.40	0.42

Table 5.4: Parameters used to generate the maps of landmarks during simulation evaluation of crowdsourced mapping contributions for localization.

in which Σ_m is not the covariance used to generate noises on geo-position estimations, but instead represents the estimated covariance obtained during field experiments:

$$\Sigma_m = \begin{pmatrix} \delta_{m,e}^2 & 0.0 \\ 0.0 & \delta_{m,n}^2 \end{pmatrix} \tag{5.17}$$

$\delta_{m,e}$ and $\delta_{m,n}$ correspond respectively to the average of the estimated deviations on the East and North axes, which were retrieved from the estimated covariance of the map obtained after the last vehicle passage during field experiments. These values are shown in Table 5.4.

5.4.2 Evaluation of Localization Performances

For each landmarks distribution, and for each noises configuration, we process a single vehicle passage, and generate corresponding sensors measurements and map of landmarks. As the vehicle reaches the end of the trajectory, it shares its observations with a centralized server, which then performs graph optimization using the $CC + GSD$ strategy with the dimension condition defined in Equation 5.7. To model the map constraint, we consider two cases:

- No map constraint. In this case, graph optimization uses only the generated sensors measurements.
- Map constraint. In this case, graph optimization also uses the generated map as prior knowledge regarding the landmarks distribution.

Although we focused until now on building and updating a map of landmarks, graph optimization also estimates the vehicle trajectory in the process. Therefore, in order to assess the performances of localization, we simply use the corresponding estimation provided by graph optimization. During our simulation experiments, the complete graph contains 512 position constraints, 5 detection constraints for each landmark, and displacement constraints that link vehicle nodes together. As a result, the graph is typically subdivided into 4 subgraphs (for the traffic signs distribution with 44 landmarks) or 9 subgraphs (for the dense landmarks distribution with 500 landmarks). As before, this process was implemented in Python and run on a laptop with an Intel i5-6200U CPU.

Results In Figure 5.12, we show the evolution of the distance errors of vehicle positioning obtained in the different cases. For each noises configuration, we depict the distance errors obtained in both landmarks distributions, using either only sensors measurements (without map), or sensors measurements along with the map constraint (with map). It is important to note that distance errors obtained in the first and third configurations, which do not consider GNSS auto-correlated noises, are typically much lower than distance errors obtained in other configurations, which consider GNSS auto-correlated noises. Thus, for visibility purposes, the vertical scale used to display results in Figure 5.12 is different for the first and third configurations, than for other configurations. To summarize the obtained results, we also show the average distance errors obtained in the different cases in Table 5.5. For a given landmarks distribution and noises configuration, the lowest error is highlighted in blue.

In order to gain further insight on the impact of the generated maps on localization performances, we depict in Figure 5.13 the estimated vehicle trajectories obtained in the different landmarks distributions. We also show the ground truth vehicle trajectory and landmarks geo-positions, and provide a zoom on a section of the trajectory. For visibility purposes, we only show results obtained in the last noises configuration, which considers the presence of GNSS auto-correlated noises, camera calibration bias, and map inconsistency.

	Traffic Signs Distribution (44 Landmarks)		Dense Landmarks Distribution (500 Landmarks)	
	Distance Errors Without Map (m)	Distance Errors With Map (m)	Distance Errors Without Map (m)	Distance Errors With Map (m)
White Gaussian Noises	2.38	1.89	2.76	0.72
GNSS Auto-Correlated Noises	12.74	7.39	12.39	0.87
Camera Calibration Bias	2.29	1.87	2.15	0.79
GNSS Auto-Correlated Noises, Camera Calibration Bias	17.28	9.53	17.29	0.94
GNSS Auto-Correlated Noises, Camera Calibration Bias, Map Inconsistency	17.28	8.14	17.29	0.92

Table 5.5: Average distance errors of vehicle positioning obtained during simulation evaluation of crowdsourced mapping contributions for localization.

In Figure 5.13, it can be seen that the estimated trajectories are not continuous and get cut at some points along the vehicle path. This corresponds to the fact that the graph gets subdivided into several subgraphs that are optimized independently. This is not a problem here, as we remain focused on evaluating the impact of the generated maps on localization performances. Nevertheless, in a real application, in which continuous localization could be required by some driving functions, such cuts could be avoided by estimating the vehicle trajectory with a different process, for instance by applying graph optimization on a sliding window of the last measurements as in [73].

Discussion As expected, if we compare the influence of the different types of noises, we can see that the lowest distance errors are obtained when making use of noises sampled from white and Gaussian distributions, while the highest distance errors are obtained in the presence of both GNSS auto-correlated noises and camera calibration bias (see Table 5.5). Nevertheless, the generated maps enable to decrease the positioning errors in all configurations, which shows the potential of our crowdsourced mapping approach to improve

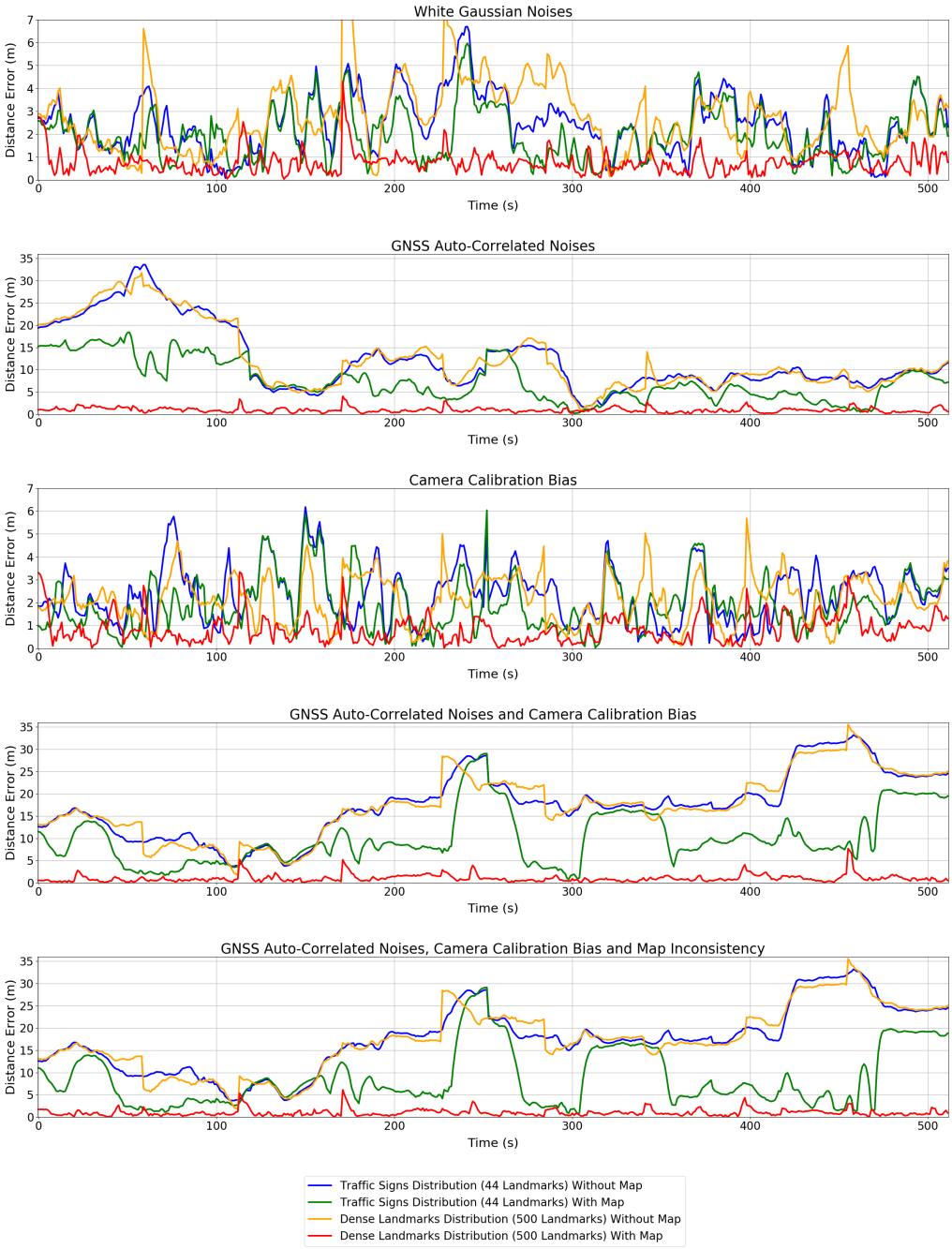

Figure 5.12: Evolution of distance errors associated with vehicle positioning, obtained during simulation evaluation of crowdsourced mapping contributions for localization.

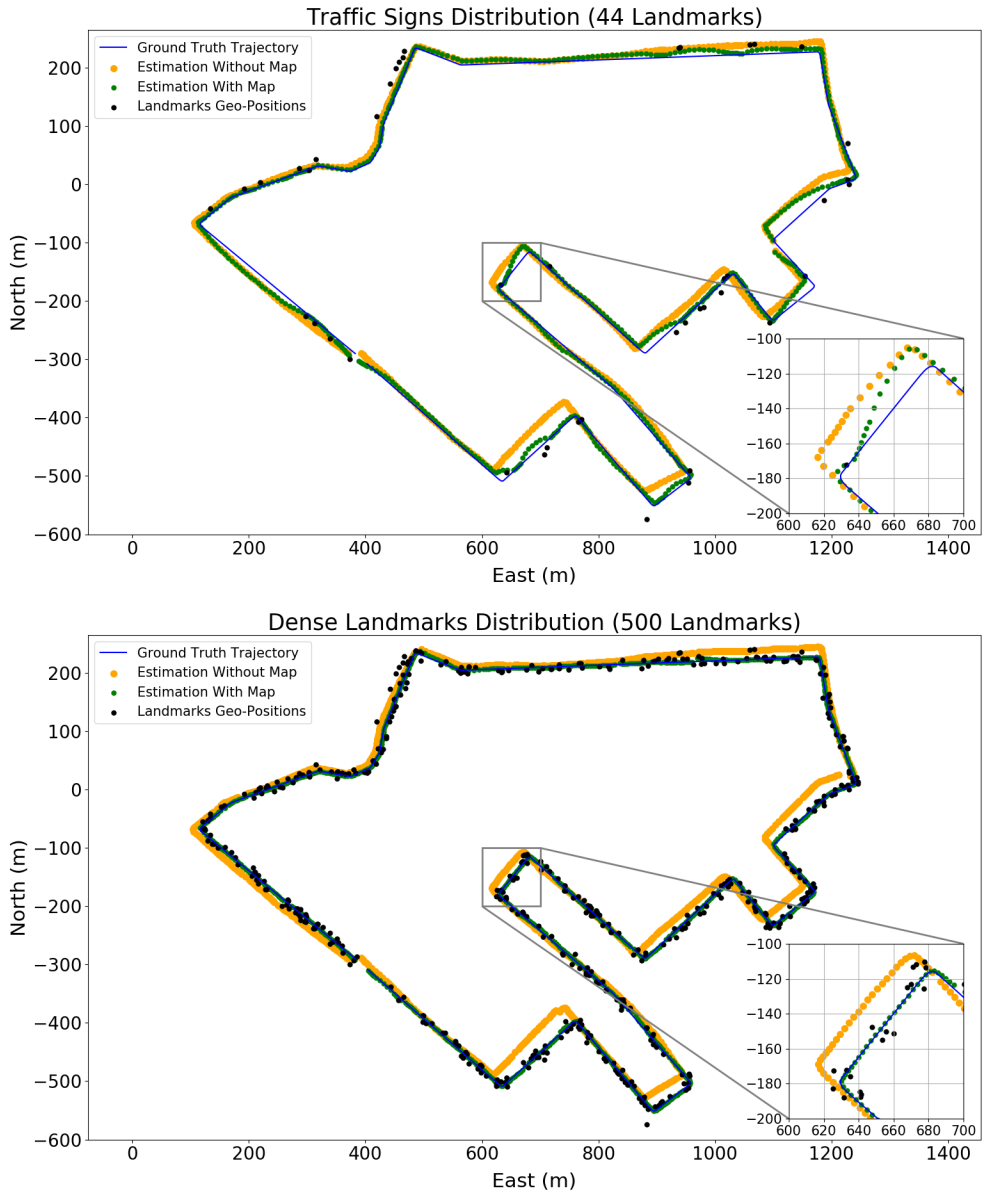

Figure 5.13: Estimated vehicle trajectories, obtained during simulation evaluation of crowd-sourced mapping contributions for localization, in the case where GNSS auto-correlated noises, camera calibration bias and map inconsistency are considered.

the localization performances in various situations. The last two configurations, which contain both GNSS auto-correlated noises and camera calibration bias, differ by whether they consider a consistent map (4^{th} configuration) or an inconsistent map (5^{th} configuration). By comparing the localization performances obtained in these two configurations, we observe that the influence of the map inconsistency remains negligible. The average distance error even appears slightly lower in the presence of map inconsistency (see Table 5.5), although this is likely arbitrary, and due to the randomly-generated noises used within the simulation.

On another note, by comparing the distance errors obtained in the dense landmarks distribution with those obtained in the traffic signs distribution, we observe that the positioning accuracy can be improved more significantly by including a higher number of landmarks in the map (see Figure 5.12). This corresponds to the fact that the map enables to improve the localization performances mainly in areas where nearby landmarks can be detected by the vehicle, but not much in other regions (see Figure 5.13). In the traffic signs distribution, the number of landmarks remains rather low, and there exists large portions of the vehicle trajectory in which not any landmark can be detected. As a result, although the map enables to improve the positioning accuracy, distance errors of several meters typically remain. Meanwhile, in the dense landmarks distribution, a higher number of landmarks, which are spread out along the vehicle path, is considered. Consequently, the vehicle is able to continuously detect them, which results in an enhancement of the positioning accuracy over the entire trajectory. In fact, when considering a realistic noises configuration with both GNSS auto-correlated noises and camera calibration bias, making use only of the sensors measurements to localize the vehicle leads to an average distance error of $17.29\ m$ (see Table 5.5). By making use also of a map of similar accuracy and consistency as the map built during field-tests, this error can be reduced to $0.92\ m$. This illustrates the effectiveness of our crowdsourced mapping solution to improve the localization performances of the vehicles, and its potential to enable reaching the targeted positioning accuracy, which is

associated with errors up to a few decimeters, by providing a map with a sufficient number of landmarks.

5.5 Conclusion

During early field-tests, making use of the triangulation-based approach, we verified the potential of crowdsourced mapping. Nevertheless, we also identified limitations associated with the triangulation optimization, which motivated our use of another solution based on graph optimization. In order to assess the scalability potential of the graph-based approach, we performed simulation experiments, considering various configurations with different numbers of landmarks, and we compared the performances obtained with the different map update strategies. The results showed the necessity to maintain cross-correlations between landmarks in the map, in order to produce an accurate and consistent map. Furthermore, they illustrated the efficiency of graph subdivisions in order to remain computationally scalable. For these two reasons, the $CC + GSD$ strategy appeared as an efficient solution for crowdsourced mapping.

Aiming to evaluate the robustness of the graph-based approach with the $CC + GSD$ strategy, we performed simulation experiments, considering a fixed number of landmarks and affecting the measurements with various types of noises, including GNSS auto-correlated noises and camera calibration bias, which are present in real situation. In all configurations, the proposed approach was able to build a map whose accuracy increases with the number of vehicle passages. Furthermore, we observed that, while GNSS auto-correlated noises lead to over-confident map updates, making use of multiple vehicle passages enables to alleviate this issue and build an accurate map of landmarks. Nevertheless, in all configurations, as the number of vehicle passages becomes high, the average distance error of the map tends to stop decreasing and converge towards an offset value. To continue improving the accuracy of the map, other issues need to be addressed, such as modelling and accounting for GNSS auto-correlated noises and camera calibration bias within the optimization.

To confront our crowdsourced mapping solution to a real situation, and assess that simulation experiments correspond to realistic conditions, we performed field-tests within a city center. Throughout multiple driving sessions, we drove our test-vehicle equipped with standard sensors for multiple passages along a loop. We built a map containing geo-positions and covariance of identified traffic signs, by making use of the graph-based approach with the $CC + GSD$ strategy. The obtained results illustrate the benefits of the crowdsourcing approach even in real conditions, as the accuracy of the built map increased with the number of vehicle passages. Especially, they show the advantage of using graph optimization rather than the triangulation optimization, as the average distance error of the map built by the triangulation-based approach during early field-tests was $5.0\,m$ after 10 vehicle passages, while the average distance error of the map obtained with the graph-based approach in these field-tests was $1.9\,m$ after the same number of vehicle passages. Furthermore, the amplitude of the obtained errors corresponds to those of the simulation experiments, and thus indicates that the simulation setup corresponds to realistic conditions.

In order to evaluate the potential of our crowdsourced mapping solution to improve the accuracy of vehicles positioning, we performed simulation experiments with various landmarks distributions and different types of measurements noises. For each landmarks distribution, we generated a map of landmarks of similar accuracy and consistency as the map built during field-tests, and studied its impact on the localization performances of a given vehicle. The obtained results illustrate the capacity of our solution to improve the accuracy of positioning in various situations. Furthermore, they show the importance to maintain a map with a sufficient landmarks density, and illustrate the potential of our approach to enable reaching the targeted positioning accuracy.

CHAPTER 6
CONCLUSION

The typical crowdsourced mapping system contains several modules, whose purpose is to register measurements, process them into usable observations, localize the vehicle and build the map of landmarks. For illustration purposes, we depict again such a system in Figure 6.1. In this thesis, we focused on the mapping part, and proposed a crowdsourced mapping solution based on graph optimization. We evaluated its performances in terms of accuracy, scalability and robustness to various types of noises by using extensive simulation and field experiments. The obtained results showed that this approach is able to take advantage from measurements issued by multiple vehicles, in order to build and update a map of landmarks while remaining computationally scalable.

6.1 Summary and Contributions

Initially, we proposed a crowdsourced mapping solution based on a triangulation optimization, and evaluated its performances during field-tests. The results analysis showed the potential of the crowdsourcing approach, as the map accuracy tends to improve through the successive vehicle passages. Nevertheless, the resulting map errors remain rather important, with an average distance error converging around $5.0\ m$. Furthermore, some critical limitations associated with the triangulation optimization could be identified, such as the inability to provide accuracy of the estimations, and the incapacity to scale to a large number of vehicles.

To address such issues, we proposed another crowdsourcing approach based on the use of graph optimization. We formulated the collaborative SLAM problem, and proposed three different strategies to update and use the map within graph optimization, which correspond to different trade-offs between the map quality and the computational scalability. The

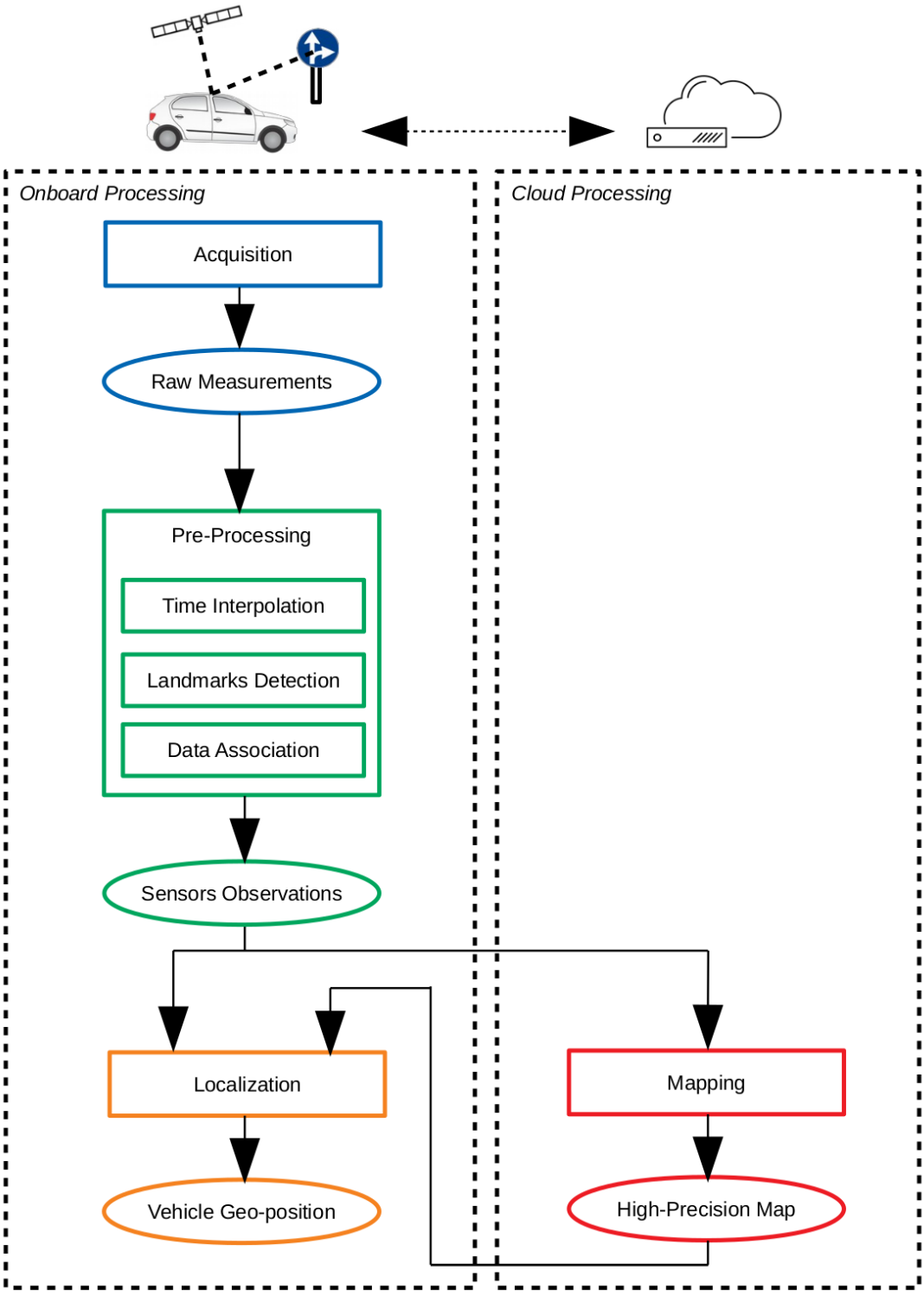

Figure 6.1: Overview of crowdsourced mapping, with modules split between onboard processing and cloud processing.

different methods were compared during simulation experiments, and one of the strategies could be identified as the most promising solution for crowdsourced mapping.

Making use of this strategy, we studied the performances and robustness of crowdsourced mapping during extensive simulation experiments with realistic errors models and sensors parameters. Especially, we studied the impact of auto-correlated and biased noises on the resulting map accuracy. We observed that, while the presence of typical GNSS auto-correlated noises can lead to inconsistent results, this can be alleviated by using a higher number of vehicle passages. In fact, in all configurations, average distance errors of the built maps reached below the meter-level after a few hundreds of vehicle passages at most.

We then performed field-tests using our test-vehicle, in order to confirm the results obtained in simulation, and draw conclusions both from a theoretical and a practical standpoint. The results showed the capability of our crowdsourced mapping approach to improve the map accuracy over the vehicle passages, and an average distance error of $1.90\,m$ was obtained after 40 vehicle passages. Furthermore, the amplitude of the obtained errors corresponds to those of the simulation experiments for the same number of vehicle passages, which indicates that the simulation setup represents realistic conditions.

Finally, we evaluated in simulation the contributions that can be expected from crowdsourced mapping to improve the localization of the vehicles. The results illustrated the effectiveness of the proposed approach in various conditions, while pointing out the importance of providing a sufficient density of landmarks in the map. To support the localization of a given vehicle, we generated a map with a dense distribution of landmarks and with an accuracy and consistency typical to those provided by our crowdsourced mapping solution. The average distance error of positioning could be improved from $17.29\,m$ to $0.92\,m$ in the presence of both GNSS auto-correlated noises and camera calibration bias, which illustrates the potential of our crowdsourcing approach to enable reaching the targeted positioning accuracy.

6.2 Future Works

Towards the extension of our solution to a complete mapping system, some improvements can be envisioned. First, the detection of landmarks could be enhanced by being automated, using for instance a CNN-derived architecture [107], and by considering other types of landmarks, such as road markings and traffic lights. Second, the data association could also be automated by processing automatically the tracking and matching of detected landmarks with the map content. This could be done by making use of visual and semantic information provided by the detection process, as well as landmarks geo-positions and their accuracy given by the map [34]. Third, the mapping process could be enhanced by including a method for automatically adding or removing landmarks from the map, when those have been consistently detected or missed by the detection process of nearby vehicles. Through these various improvements, we could maintain the map up-to-date, while increasing its quality, as well as the resulting localization accuracy.

In order to improve the mapping performances of our solution, further improvements can be envisioned that relate to the formulation of graph optimization. Indeed, during our experiments, we observed that after a given number of vehicle passages, new measurements are not able to improve the accuracy of the map, which tends to converge towards a bias. To deal with this issue, two strategies can be considered. In a first approach, we could account for unmodeled effects, such as biased and auto-correlated noises, by making use of covariance inflation as described in [98], which would enable to prevent the map from becoming over-confident. In a second approach, we could modify the graph structure similarly as in [100], and estimate the biased and auto-correlated noises within the optimization. Although this would require to model such noises, it could lead to substantial improvements of the map accuracy. In fact, a complete mapping solution would likely integrate both of these approaches, by modeling and estimating part of the biased and auto-correlated noises, and applying covariance inflation to account for other noises that are too complex to estimate.

Furthermore, crowdsourced mapping could lead to additional applications, such as remote diagnosis and maintenance. Indeed, as vehicles are expected to embed more and more sensors needed for a variety of ADAS functions, it is crucial to ensure that such sensors remain well calibrated and functioning. Within a crowdsourcing approach, vehicles with degraded equipment could be identified by comparing landmarks geo-positions estimated using their measurements, with landmarks geo-positions present in the map. Whenever both estimations would differ too severely for continuous periods of time, this could indicate the likely presence of defective sensors or calibration. Subsequently, such vehicles could be notified and addressed to a garage for repair purposes. In the case of a defective calibration, one could also imagine to avoid going through garage maintenance, by using the map as prior knowledge to quantify and compensate for such calibration error on the fly.

Appendices

APPENDIX A

GNSS AUTO-CORRELATED NOISES

A.1 Introduction

In the following, we describe how we generate realistic GNSS measurements by affecting them with auto-correlated noises, in order to assess the potential of our graph-based approach.

In the simulation, any position measurement Y_k provided by the GNSS receiver is assumed to be affected by a stationary, non-white noise $\epsilon_{y,k}$ that is sampled from a zero-mean, Gaussian distribution with covariance matrix Σ_y. This covariance matrix is assumed to be diagonal, which implies that noises generated on the East and North axes are independent. For clarity purposes, in the following, we write all variables as scalars associated with a given axis, thus notations such as Y_k and $\epsilon_{y,k}$ correspond to measurements and noises that are either on the East axis, or on the North axis.

A.2 AR Modeling

The objective consists in generating a noise $\epsilon_{y,k}$, known to be non-white, by using auto-correlated errors associated with real GNSS measurements. To achieve this, we model such auto-correlated noise by making use of an AR model of first order [110]:

$$\epsilon_{y,k} = \alpha\, \epsilon_{y,k-1} + \beta\, \epsilon_{b,k} \tag{A.1}$$

where $\epsilon_{y,k-1}$ corresponds to the noise at the previous instant, and $\epsilon_{b,k}$ is a stationary, white noise sampled from a zero-mean, Gaussian distribution with covariance matrix Σ_b. The coefficients α and β are scalars that model the amplitude of auto-correlations, and that need

to be estimated.

First, we express β relatively to α, as this will help further developments. The noise $\epsilon_{b,k}$ is white, and therefore independent from the previous noise $\epsilon_{y,k-1}$. Thus, we can write:

$$\Sigma_y = \alpha^2 \, \Sigma_y + \beta^2 \, \Sigma_b \tag{A.2}$$

In order to properly study the effect of auto-correlated noises, we need to generate noises whose covariance matrix remain similar as in other configurations (see subsection 5.2.3). Thus, we impose the noise $\epsilon_{y,k}$ to have the same covariance matrix as $\epsilon_{b,k}$:

$$\Sigma_y = \Sigma_b \tag{A.3}$$

We can re-write Equation A.2 as:

$$\Sigma_y = \alpha^2 \, \Sigma_y + \beta^2 \, \Sigma_y = (\alpha^2 + \beta^2)\Sigma_y \tag{A.4}$$

Generally, we can assume the covariance matrix Σ_y to be invertible:

$$\Sigma_y \Sigma_y^{-1} = (\alpha^2 + \beta^2)\Sigma_y \Sigma_y^{-1} \tag{A.5}$$

Finally, we obtain:

$$\beta = \sqrt{1 - \alpha^2} \tag{A.6}$$

A.3 Calculation of α

In order to obtain the value of α, we express the auto-correlation function ϕ_n associated with auto-correlated noises:

$$\phi_n = E(\epsilon_{y,k} \, \epsilon_{y,k-n}) \tag{A.7}$$

where $E(x)$ corresponds to the expectation of a variable x.

By using Equation A.1, we write:

$$\phi_n = E((\alpha\epsilon_{y,k-1} + \beta\epsilon_{b,k})(\alpha\epsilon_{y,k-1-n} + \beta\epsilon_{b,k-n})) \tag{A.8}$$

$$\phi_n = \alpha^2 E(\epsilon_{y,k-1}\,\epsilon_{y,k-1-n}) + \alpha\beta(E(\epsilon_{y,k-1}\,\epsilon_{b,k-n}) + E(\epsilon_{y,k-1-n}\,\epsilon_{b,k})) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-n}) \tag{A.9}$$

Due to the fact that we consider a stationary noise $\epsilon_{y,k}$ in the simulation:

$$E(\epsilon_{y,k-1}\,\epsilon_{y,k-1-n}) = E(\epsilon_{y,k}\,\epsilon_{y,k-n}) = \phi_n \tag{A.10}$$

And:

$$\phi_n = \alpha^2\phi_n + \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-n}) + \alpha\beta E(\epsilon_{y,k-1-n}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-n}) \tag{A.11}$$

$$(1-\alpha^2)\phi_n = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-n}) + \alpha\beta E(\epsilon_{y,k-1-n}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-n}) \tag{A.12}$$

$$\beta^2\phi_n = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-n}) + \alpha\beta E(\epsilon_{y,k-1-n}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-n}) \tag{A.13}$$

In the following, we consider different values of n, and develop the expression of ϕ_n. Generally, n can be either negative or positive, with developments of equations being very similar in both cases. Therefore, for simplification purposes, we only develop cases where n is positive.

A.3.1 Case: n = 0

The auto-correlation function ϕ_0 is given by:

$$\beta^2\phi_0 = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k}) + \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k}) \tag{A.14}$$

The noise $\epsilon_{b,k}$ is white and independent from the previous noise $\epsilon_{y,k-1}$:

$$\beta^2\phi_0 = \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k}) = \beta^2\Sigma_b \tag{A.15}$$

And finally:

$$\phi_0 = \Sigma_b \tag{A.16}$$

A.3.2 Case: n=1

The auto-correlation function ϕ_1 is given by:

$$\beta^2\phi_1 = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-1}) + \alpha\beta E(\epsilon_{y,k-2}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-1}) \tag{A.17}$$

The noise $\epsilon_{b,k}$ is white and independent from the noise $\epsilon_{y,k-2}$:

$$\beta^2\phi_1 = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-1}) \tag{A.18}$$

$$\beta\phi_1 = \alpha E(\epsilon_{y,k-1}\,\epsilon_{b,k-1}) \tag{A.19}$$

The noises $\epsilon_{y,k}$ and $\epsilon_{b,k}$ follow a stationary behavior:

$$\beta\phi_1 = \alpha E(\epsilon_{y,k}\,\epsilon_{b,k}) \tag{A.20}$$

The use of Equation A.1 gives:

$$\beta\phi_1 = \alpha E((\alpha\epsilon_{y,k-1} + \beta\epsilon_{b,k})\epsilon_{b,k}) \tag{A.21}$$

$$\beta\phi_1 = \alpha^2 E(\epsilon_{y,k-1}\,\epsilon_{b,k}) + \alpha\beta E(\epsilon_{b,k}\,\epsilon_{b,k}) = \alpha\beta E(\epsilon_{b,k}\,\epsilon_{b,k}) \tag{A.22}$$

And finally:

$$\phi_1 = \alpha E(\epsilon_{b,k}\,\epsilon_{b,k}) = \alpha\Sigma_b \tag{A.23}$$

A.3.3 Case: n=2

The auto-correlation function ϕ_2 is given by:

$$\beta^2\phi_2 = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-2}) + \alpha\beta E(\epsilon_{y,k-3}\,\epsilon_{b,k}) + \beta^2 E(\epsilon_{b,k}\,\epsilon_{b,k-2}) \tag{A.24}$$

The noise $\epsilon_{b,k}$ is white and independent from the noise $\epsilon_{y,k-3}$:

$$\beta^2\phi_2 = \alpha\beta E(\epsilon_{y,k-1}\,\epsilon_{b,k-2}) \tag{A.25}$$

$$\beta\phi_2 = \alpha E(\epsilon_{y,k-1}\,\epsilon_{b,k-2}) \tag{A.26}$$

The noises $\epsilon_{y,k}$ and $\epsilon_{b,k}$ follow a stationary behavior:

$$\beta\phi_2 = \alpha E(\epsilon_{y,k}\,\epsilon_{b,k-1}) \tag{A.27}$$

The use of Equation A.1 gives:

$$\beta\phi_2 = \alpha E((\alpha\epsilon_{y,k-1} + \beta\epsilon_{b,k})\epsilon_{b,k-1}) \tag{A.28}$$

$$\beta\phi_2 = \alpha^2 E(\epsilon_{y,k-1}\,\epsilon_{b,k-1}) + \alpha\beta E(\epsilon_{b,k}\,\epsilon_{b,k-1}) = \alpha^2 E(\epsilon_{y,k-1}\,\epsilon_{b,k-1}) \tag{A.29}$$

The use of Equation A.19 gives:

$$\beta\phi_2 = \alpha\beta\phi_1 \tag{A.30}$$

And finally:

$$\phi_2 = \alpha\phi_1 = \alpha^2\Sigma_b \tag{A.31}$$

A.3.4 General case

Through recurrence, we find the general form of the auto-correlation function ϕ_n, where n can be either negative or positive:

$$\phi_n = \alpha^{|n|}\Sigma_b \tag{A.32}$$

To compute the value of α, we express the derivative of ϕ_n with respect to n:

$$\frac{\partial \phi_n}{\partial n} = sgn(n)\, \alpha^{|n|}\, log(\alpha)\, \Sigma_b \tag{A.33}$$

where $sgn(x)$ corresponds to the sign of x.

We evaluate the slope K of the auto-correlation function ϕ_n at $n = 0^+$:

$$K = \left.\frac{\partial \phi_n}{\partial n}\right|_{n=0^+} = log(\alpha)\, \Sigma_b \tag{A.34}$$

Let n_0 be defined such that the slope K passes through Σ_b at $n = 0$, and through the horizontal axis at $n = n_0$, as illustrated in Figure A.1. This gives:

$$K\, n_0 + \Sigma_b = 0 \tag{A.35}$$

And:

$$n_0 = \frac{-\Sigma_b}{K} = \frac{-\Sigma_b}{log(\alpha)\, \Sigma_b} = \frac{-1}{log(\alpha)} \tag{A.36}$$

It is known that α is close to 1 for typical GNSS measurements:

$$n_0 \approx \frac{-1}{\alpha - 1} \tag{A.37}$$

And finally:

$$\alpha \approx 1 - \frac{1}{n_0} \tag{A.38}$$

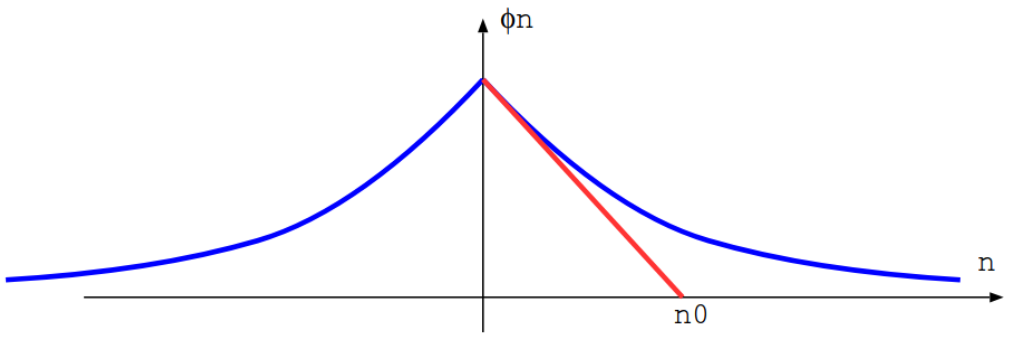

Figure A.1: Example of an auto-correlation function (blue), with its slope at $n = 0^+$ (red).

A.4 Conclusion

In order to compute α and β, we use the following procedure:

- Estimate the auto-correlation function ϕ_n.
- Compute the value of α using Equation A.38.
- Compute the value of β using Equation A.6.

REFERENCES

[1] T. G. R. Reid et al., "Localization requirements for autonomous vehicles," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 3, Sep. 2019.

[2] L. Narula, M. J. Murrian, and T. E. Humphreys, "Accuracy limits for globally-referenced digital mapping using standard gnss," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3075–3082.

[3] A. Stoven-Dubois et al., "Graph-based approach for crowdsourced mapping: Evaluation through field experiments," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020, pp. 260–265.

[4] G. Bresson et al., "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, Sep. 2017.

[5] V. Ila et al., "SLAM++-A highly efficient and temporally scalable incremental SLAM framework," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017.

[6] K. Doherty, D. Fourie, and J. Leonard, "Multimodal semantic slam with probabilistic data association," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2419–2425.

[7] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?" In *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2657–2664.

[8] D. Wilbers, C. Merfels, and C. Stachniss, "A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 220–225.

[9] A. Stoven-Dubois et al., "A Collaborative Framework for High-Definition Mapping," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 1845–1850.

[10] ——, "Graph optimization methods for large-scale crowdsourced mapping," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020, pp. 1–8.

[11] National Highway Traffic Safety Administration, "Automated Vehicles for Safety", https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety, Accessed: 2021-09-14.

[12] V. Banks et al., “Is partially automated driving a bad idea? observations from an on-road study.,” *Applied ergonomics*, vol. 68, pp. 138–145, 2018.

[13] J. Kocić, N. Jovičić, and V. Drndarević, “Sensors and sensor fusion in autonomous vehicles,” in *2018 26th Telecommunications Forum (TELFOR)*, 2018, pp. 420–425.

[14] C. Yan, “Can you trust autonomous vehicles : Contactless attacks against sensors of self-driving vehicle,” 2016.

[15] N. Lu et al., “Connected vehicles: Solutions and challenges,” *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[16] R. Schubert, E. Richter, and G. Wanielik, “Comparison and evaluation of advanced motion models for vehicle tracking,” in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–6.

[17] J. Borenstein and L. Feng, “Measurement and correction of systematic odometry errors in mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 869–880, 1996.

[18] Z. Fan et al., “Experimental evaluation of an encoder trailer for dead-reckoning in tracked mobile robots,” in *Proceedings of Tenth International Symposium on Intelligent Control*, 1995, pp. 571–576.

[19] M. Chirca, “Perception pour la navigation et le contrôle des robots mobiles. Application à un système de voiturier autonome,” Theses, Université Blaise Pascal - Clermont-Ferrand II, Dec. 2016.

[20] N. Kubo, K. Kobayashi, and R. Furukawa, “Gnss multipath detection using continuous time-series c/n0,” *Sensors*, vol. 20, no. 14, 2020.

[21] "Multi-Constellation and Multi-Frequency", https://novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/multi-constellation-and-multi-frequency, Accessed: 2021-12-05.

[22] "Les systèmes d’augmentation de la précision LBAS et SBAS", https://www.reseauteria.com/2019/12/30/les-systemes-daugmentation-sbas-et-lbas, Accessed: 2021-12-05.

[23] T. R. Lemmon and D. G. P. Gerdan, “The influence of the number of satellites on the accuracy of rtk gps positions,” *Australian Surveyor*, vol. 44, no. 1, pp. 64–70, 1999.

[24] M. Papoutsidakis et al., "Motion sensors and transducers to navigate an intelligent mechatronic platform for outdoor applications," *Sensors and Transducers*, vol. 198, pp. 16–24, Apr. 2016.

[25] J. Pérez, D. Gonzalez, and V. Milanés, "Vehicle control in adas applications," in *Intelligent Transport Systems*. John Wiley & Sons, Ltd, 2015, ch. 11, pp. 206–219.

[26] P. Wei et al., "Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics*, vol. 7, p. 84, May 2018.

[27] L. Heng et al., "Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4695–4702.

[28] "Intelligent transport systems (its); access layer specification for intelligent transport systems operating in the 5 ghz frequency band," European Telecommunications Standard Institute (ETSI), Standard, May 2013.

[29] "Ieee standard for information technology– telecommunications and information exchange between systems– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments," Institute of Electrical and Electronics Engineers (IEEE), Standard, Jul. 2010.

[30] M. Rondinone and A. Correa, "Definition of v2x message sets," Transition Areas for Infrastructure-Assisted Driving, Standard, May 2016.

[31] H. G. Seif and X. Hu, "Autonomous driving in the icity—hd maps as a key challenge of the automotive industry," *Engineering*, vol. 2, no. 2, pp. 159–162, 2016.

[32] B. Chenchana, "Localisation collaborative visuelle-inertielle de robots hétérogènes communicants," Theses, Université de Limoges, Mar. 2019.

[33] A. Birk et al., "A quantitative assessment of structural errors in grid maps," *Autonomous Robots*, vol. 28, pp. 187–196, Feb. 2010.

[34] X. Qu, B. Soheilian, and N. Paparoditis, "Landmark based localization in urban environment," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 90–103, 2018, Geospatial Computer Vision.

[35] K. Pirker, M. Rüther, and H. Bischof, "Cd slam - continuous localization and mapping in a dynamic world," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3990–3997.

[36] J. McCall and M. Trivedi, "Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 20–37, 2006.

[37] H. Zhu et al., "Overview of environment perception for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2584–2601, 2017.

[38] J. Aponte et al., "Quality assessment of a network-based RTK GPS service in the UK," *Journal of Applied Geodesy*, vol. 3, no. 1, pp. 25–34, Mar. 2009.

[39] I.-S. Lee and L. Ge, "The performance of rtk-gps for surveying under challenging environmental conditions," *Earth, Planets and Space*, vol. 58, pp. 515–522, 2006.

[40] S. Bauer, M. Obst, and G. Wanielik, "3d environment modeling for gps multipath detection in urban areas," in *International Multi-Conference on Systems, Signals Devices*, 2012, pp. 1–5.

[41] Z. Tao et al., "Mapping and localization using gps, lane markings and proprioceptive sensors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 406–412.

[42] M. Aftatah et al., "Gps/ins/odometer data fusion for land vehicle localization in gps denied environment," *Mathematical Models and Methods in Applied Sciences*, vol. 11, p. 62, 2016.

[43] K. Saadeddin, M. F. Abdel-Hafez, and M. A. Jarrah, "Estimating vehicle state by gps/imu fusion with vehicle dynamics," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013, pp. 905–914.

[44] A. Ndjeng Ndjeng et al., "Low cost IMU-Odometer-GPS ego localization for unusual maneuvers," *Information Fusion*, vol. 12, pp 264–274, Jan. 2011.

[45] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991, 1442–1447 vol.3.

[46] L. Wei et al., "Intelligent vehicle localization in urban environments using ekf-based visual odometry and gps fusion," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 13 776–13 781, 2011, 18th IFAC World Congress.

[47] M. Schreiber et al., "Vehicle localization with tightly coupled gnss and visual odometry," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 858–863.

[48] J. Andrade-Cetto, T. Vidal-Calleja, and A. Sanfeliu, "Unscented transformation of vehicle states in slam," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 323–328.

[49] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[50] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[51] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 195–200.

[52] A. J. Davison et al., "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[53] J. Civera et al., "Towards semantic slam using a monocular camera," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1277–1284.

[54] X. Chen et al., "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.

[55] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[56] G. Dissanayake et al., "An experimental and theoretical investigation into simultaneous localisation and map building," in *Experimental Robotics VI*, Springer London, 2000, pp. 265–274.

[57] F. Dellaert et al., "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, 1322–1328 vol.2.

[58] D. Wang et al., "Lidar scan matching ekf-slam using the differential model of vehicle motion," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 908–912.

[59] G. Bresson et al., "Real-Time Monocular SLAM With Low Memory Requirements," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1827–1839, 2015.

[60] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, I. Kadar, Ed., International Society for Optics and Photonics, vol. 3068, SPIE, 1997, pp. 182–193.

[61] L. Delobel et al., "A Real-Time Map Refinement Method Using a Multi-Sensor Localization Framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1644–1658, May 2019.

[62] S. Julier and J. Uhlmann, "General decentralized data fusion with covariance intersection (ci)," *Handbook of Multisensor Data Fusion: Theory and Practice*, Jun. 2001.

[63] M. Montemerlo et al., "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Eighteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, 2002, pp. 593–598.

[64] ——, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03, Morgan Kaufmann Publishers Inc., 2003, pp. 1151–1156.

[65] J. Z. Sasiadek, A. Monjazeb, and D. Necsulescu, "Navigation of an autonomous mobile robot using ekf-slam and fastslam," in *2008 16th Mediterranean Conference on Control and Automation*, 2008, pp. 517–522.

[66] A. Eudes and M. Lhuillier, "Error Propagations for Local Bundle Adjustment," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, IEEE, Jun. 2009.

[67] D. Steedly and I. Essa, "Propagation of innovative information in non-linear least-squares structure from motion," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 223–229.

[68] Z. Zhang and Y. Shan, "Incremental motion estimation through modified bundle adjustment," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 2, 2003, pp. II–343.

[69] E. Mouragnon et al., "Real time localization and 3d reconstruction," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 363–370.

[70] D. A. Cucci, M. Rehak, and J. Skaloud, "Bundle adjustment with raw inertial observations in uav applications," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 130, pp. 1–12, 2017.

[71] G. Grisetti et al., “Efficient estimation of accurate maximum likelihood maps in 3d,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3472–3478.

[72] R. Kümmerle et al., “G2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[73] D. Wilbers, C. Merfels, and C. Stachniss, “Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving,” in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5951–5957.

[74] A. Alharake et al., “Urban localization inside cadastral maps using a likelihood field representation,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1329–1335.

[75] L. Yu et al., “Monocular Urban Localization using Street View,” in *14th International Conference on Control, Automation, Robotics and Vision (ICARCV’2016)*, Nov. 2016.

[76] R. Spangenberg, D. Goehring, and R. Rojas, “Pole-based localization for autonomous vehicles in urban scenarios,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 2161–2166.

[77] G. Floros, B. van der Zander, and B. Leibe, “Openstreetslam: Global vehicle localization using openstreetmaps,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1054–1059.

[78] B. Suger and W. Burgard, “Global outer-urban navigation with openstreetmap,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1417–1422.

[79] G. Vaca-Castano et al., “City scale geo-spatial trajectory estimation of a moving camera,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1186–1193.

[80] P. Agarwal, W. Burgard, and L. Spinello, "Metric Localization using Google Street View", arXiv, 2015. eprint: 1503.04287 (cs.RO).

[81] J. Levinson, M. Montemerlo, and S. Thrun, “Map-based precision vehicle localization in urban environments,” in *Robotics: Science and Systems*, 2007.

[82] D. Zou and P. Tan, “Coslam: Collaborative visual slam in dynamic environments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 354–366, 2013.

[83] R. Dubé et al., “Segmap: 3d segment mapping using data-driven descriptors,” *Robotics: Science and Systems XIV*, Jun. 2018.

[84] R. Arumugam et al., “Davinci: A cloud computing framework for service robots,” in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3084–3089.

[85] A. Gil et al., “Multi-robot visual slam using a rao-blackwellized particle filter,” *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68–80, 2010.

[86] D. Zou, P. Tan, and W. Yu, “Collaborative visual slam for multiple agents:a brief survey,” *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 461–482, 2019, 3D Vision.

[87] G. Bresson, R. Aufrère, and R. Chapuis, “A General Consistent Decentralized Simultaneous Localization And Mapping Solution,” *Robotics and Autonomous Systems*, vol. 74, Part A, pp. 128–147, 2015.

[88] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM", 2017. arXiv: 1710.05772 [cs.RO].

[89] A. Zolotovitski et al., “Analysis of potential to improve maps using car probe data,” in *Proceedings of the 10th ACM SIGSPATIAL Workshop on Computational Transportation Science*, ser. IWCTS’17, 2017, pp. 24–29.

[90] H. Luo et al., “Constructing an indoor floor plan using crowdsourcing based on magnetic fingerprinting,” *Sensors*, vol. 17, no. 11, 2017.

[91] B. Zhou et al., “A graph optimization-based indoor map construction method via crowdsourcing,” *IEEE Access*, vol. 6, pp. 33 692–33 701, 2018.

[92] J. Huai et al., “Collaborative monocular slam with crowdsourced data,” *NAVIGATION*, vol. 65, no. 4, pp. 501–515, 2018.

[93] O. Dabeer et al., "An End-to-End System for Crowdsourced 3d Maps for Autonomous Vehicles: The Mapping Component", 2017. arXiv: 1703 . 10193 [cs.RO].

[94] M. Herb et al., “Crowd-sourced semantic edge mapping for autonomous vehicles,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7047–7053.

[95] T. Qin et al., "RoadMap: A Light-Weight Semantic Map for Visual Localization towards Autonomous Driving", 2021. arXiv: 2106.02527 [cs.CV].

[96] C. Kim et al., “Crowd-sourced mapping of new feature layer for high-definition map,” *Sensors*, vol. 18, no. 12, p. 4172, Nov. 2018.

[97] J. Shaw, "Least-Squares Intersection of Lines", https://silo.tips/download/least-squares-intersection-of-lines, Accessed: 2021-12-22.

[98] B. Noack, S. J. Julier, and U. D. Hanebeck, “Treatment of biased and dependent sensor data in graph-based slam,” in *2015 18th International Conference on Information Fusion (Fusion)*, 2015, pp. 1862–1867.

[99] G. Grisetti et al., “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[100] V. Indelman et al., “Information fusion in navigation systems via factor graph based incremental smoothing,” *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 721–738, 2013.

[101] L. Carlone et al., “Initialization techniques for 3d slam: A survey on rotation estimation and its use in pose graph optimization,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4597–4604.

[102] C. Campos et al., “Fast and robust initialization for visual-inertial slam,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1288–1294.

[103] S. Agarwal et al., "Ceres Solver", http://ceres-solver.org, Accessed: 2021-12-05.

[104] F. Dellaert and M. Kaess, “Factor graphs for robot perception,” *Foundations and Trends in Robotics*, vol. 6, pp. 1–139, Jan. 2017.

[105] P. Polack et al., “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818.

[106] J. Vallvé, J. Solà, and J. Andrade-Cetto, “Graph SLAM Sparsification With Populated Topologies Using Factor Descent Optimization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1322–1329, Jun. 2018.

[107] Z. Zhu et al., “Traffic-Sign Detection and Classification in the Wild,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2110–2118.

[108] H. Li and F. Nashashibi, “Cooperative Multi-Vehicle Localization Using Split Covariance Intersection Filter,” *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 2, pp. 33–44, 2013.

[109] B. Wang, W. Shi, and Z. Miao, "Confidence Analysis of Standard Deviational Ellipse and Its Extension into Higher Dimensional Euclidean Space," *PLoS ONE*, vol. 10, 2015.

[110] J. Laneurit, R. Chapuis, and F. Chausse, "Accurate vehicle positioning on a numerical map," *International Journal of Control Automation and Systems*, vol. 3, pp. 15–31, 2005.

[111] X. Gong and Y. Lin and J. Liu, "3D LIDAR-Camera Extrinsic Calibration Using an Arbitrary Trihedron," *Sensors*, vol. 13, no. 2, pp. 1902–1918, 2013.

[112] C. Brenner, "Vehicle localization using landmarks obtained by a lidar mobile mapping system," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 38, Jan. 2010.