HAL
open science

Lane-level vehicle localization with integrity monitoring for data aggregation

Franck Li

To cite this version:

Franck Li. Lane-level vehicle localization with integrity monitoring for data aggregation. Robotics [cs.RO]. Université de Technologie de Compiègne, 2018. English. NNT : 2018COMP2458 . tel-02186904

HAL Id: tel-02186904

https://theses.hal.science/tel-02186904v1

Submitted on 17 Jul 2019

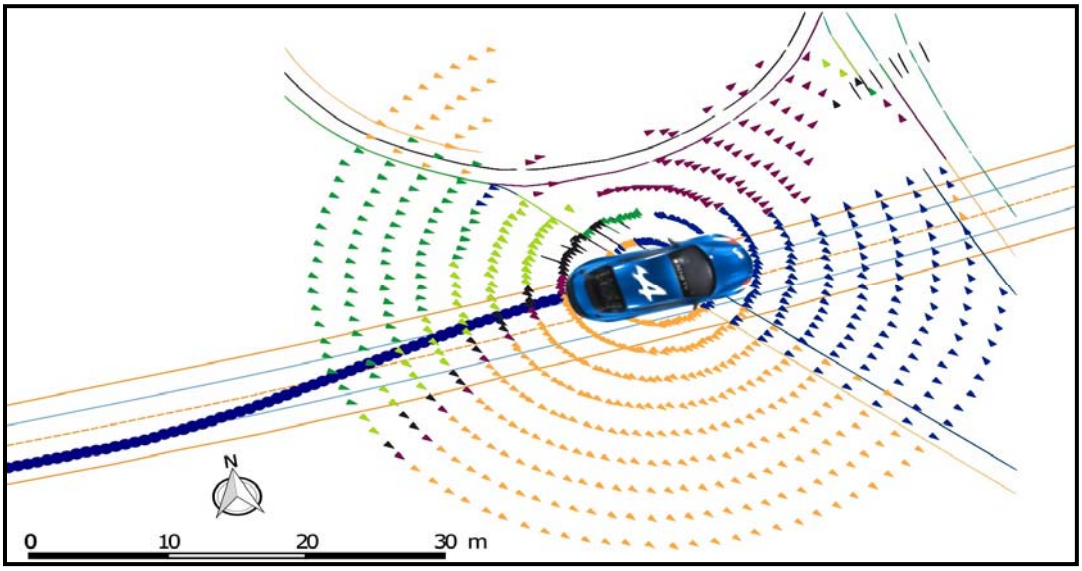
HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Par **Franck LI**

Lane-level vehicle localization with integrity monitoring for data aggregation

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



Soutenue le 18 décembre 2018
Spécialité : Robotique : Unité de recherche Heudyasic (UMR-7253)

D2458

University of Technology of Compiègne
Thesis
Submitted in partial fulfillment of the requirements for the degree of Doctor

Area of Specialization: Robotics

Spécialité : Robotique

By **Franck LI**

Lane-Level Vehicle Localization With Integrity Monitoring for Data Aggregation

Heudiasyc Laboratory, UMR UTC/CNRS 7253
In an industrial partnership (CIFRE) with Renault s.a.s.

Defended on the 18*th* of December 2018

Thesis Committee:

Reviewers:	David Bétaille	IFSTTAR Nantes
	Juliette Marais	IFSTTAR Lille
Examiners:	Tankut Acarman	Galatasaray University
	Véronique Cherfaoui	Univ. de Technologie de Compiègne
	Clément Zinoune	Renault s.a.s.
Supervisor:	Philippe Bonnifait	Univ. de Technologie de Compiègne
Industrial Supervisor:	Javier Ibanez-Guzman	Renault s.a.s.

Acknowledgments

"The time is gone, the song is over, thought I'd something more to say."

"I would never do a thesis", as I once said. But here I am, closing up my manuscript: life can be really surprising when you make bold choices. This thesis did open my mind to the research world, giving me the opportunity to work in the intelligent vehicle domain. But more importantly, it allowed me to meet extraordinary people.

First, I would like to thank Philippe Bonnifait and Javier Ibañez-Guzmán for convincing me to begin this thesis adventure, giving me the opportunity to work on the rising topic of digital maps for autonomous vehicles. Your constant support has been essential for me during these years. My gratitude goes also to the Heudiasyc laboratory and Renault for providing good working environments for this thesis. I would like to thank all the jury members for their interest in my research and for accepting to take the time to assess my work.

I was lucky to be working with amazing people at both of my offices. I would like to thank all the people at Heudiasyc, who made Compiègne a nice place to work at, especially Ahmet, Bérengère, Brigitte, Elwan, Freddy, Gildas, Nathalie, Philippe, Stéphane, Sabine, Thierry, Thomas, Yohan, Wendell. The coffee breaks were always a good place to learn a lot, on lots of topics. Outside of the laboratory, I would also like to thank Tes Lewis for the tea breaks, far away from the research world.

During my stay at the Technocentre, I had the chance to be in contact with several teams: I have been the adopted PhD student for the Research team, have pretended to be a normal engineer at the Amont team and have been the one who was desperately trying to finish writing his manuscript at the Fusion team. I have to thank Johann Mousain for welcoming me in his team during my thesis (and even a bit afterwards). Thanks also to all the members of the Amont ADAS team, especially Jacques, all the Freds, Guillaume, Marie-Anne, Pauline, Rémi. My gratitude then goes to Guillermo Pita-Gil for giving me the opportunity to continue my Renault adventure with the Fusion team.

A special thought goes to my fellow PhD student, companions of misfortune during this adventure. First, the predecessors, Clément and Alexandre, for making a thesis look like a good idea! Then, Romain for the coffee breaks during which science has greatly advanced; Mathieu, Yrvann (you see, not everything has been done 20 years ago!); and Iris, who has been a great support (even if it was cold outside) during the redaction, with its (hard) periods of stalling.

Last but not least, I would like to thank all my family members for their support during this period, that has not been the easiest personally.

Marc, Noemie, Pauline, Quentin and especially Julien, my dearest friends have always been there when I needed them most. Thank you for everything, for all the great moments during which I was able to take a break from the pressure of the thesis.

As a last note, I would also like to thank Dream Theater, Gojira, Karnivool, Marcus Miller, Miles Davis, Pink Floyd, Tool, among many others for their musicianship, helping me keep my sanity during this period of my life.

Thanks to everyone who supported me during this thesis.

Abstract

The information stored in digital road maps has become very important for intelligent vehicles. As intelligent vehicles address more complex environments, the accuracy requirements for this information have increased. Regarded as a geographic database, digital road maps contain contextual information about the road network, crucial for a good understanding of the environment. When combined with data acquired from on-board sensors, a better representation of the environment can be made, improving the vehicle's situation understanding.

Sensors performance can vary drastically depending on the location of the vehicle, mainly due to environmental factors. Comparatively, a map can provide prior information more reliably but to do so, it depends on another essential component: a localization system. Global Navigation Satellite Systems (GNSS) are commonly used in automotive to provide an absolute positioning of the vehicle, but its accuracy is not perfect: GNSS are prone to errors, also depending greatly on the environment (e.g., multipaths).

Perception and localization systems are two important components of an intelligent vehicle whose performances vary in function of the vehicle location. This research focuses on their common denominator, the digital road map, and its use as a tool to assess their performance. The idea developed during this thesis is to use the map as a learning canvas, to store georeferenced information about the performance of the sensors during repetitive travels. This requires a robust localization with respect to the map to be available, through a process of map-matching. The main problematic is the discrepancy between the accuracy of the map and of the GNSS, creating ambiguous situations. This thesis develops a map-matching algorithm designed to cope with these ambiguities by providing multiple hypotheses when necessary. The objective is to ensure the integrity of the result by returning a hypothesis set containing the correct matching with high probability. The method relies on proprioceptive sensors via a dead-reckoning approach aided by the map. A coherence checking procedure using GNSS redundant information is then applied to isolate a single map-matching result that can be used to write learning data with confidence in the map. The possibility to handle the digital map in read/write operation has been assessed and the whole writing procedure has been tested on data recorded by test vehicles on open roads.

Résumé

Les informations contenues dans les cartes routières numériques revêtent une importance grandissante dans le domaine des véhicules intelligents. La prise en compte d'environnements de plus en plus complexes a augmenté le niveau de précision exigé des informations cartographiques. Les cartes routières numériques, considérées ici comme des bases de données géographiques, contiennent des informations contextuelles sur le réseau routier, facilitant la compréhension correcte de l'environnement. En les combinant avec les données provenant des capteurs embarqués, une représentation plus fine de l'environnement peut être obtenue, améliorant grandement la compréhension de contexte du véhicule et la prise de décision.

La performance des différents capteurs peut varier grandement en fonction du lieu considéré, ceci étant principalement dû à des facteurs environnementaux. Au contraire, une carte peut fournir ses informations de manière fiable, sans être affectée par ces éléments extérieurs, mais pour cela, elle doit reposer sur un autre élément essentiel : une source de localisation. Le secteur automobile utilise les systèmes de localisation globale par satellite (GNSS) à des fins de localisation absolue, mais cette solution n'est pas parfaite, étant soumise à différentes sources d'erreur. Ces erreurs sont elles aussi dépendantes de l'environnent d'évolution du véhicule (par exemple, des multi-trajets causés par des bâtiments).

Nous sommes donc en présence de deux systèmes centraux, dont les performances sont dépendantes du lieu considéré. Cette étude se focalise sur leur dénominateur commun : la carte routière numérique, et son utilisation en tant qu'outil d'évaluation de leur performance. L'idée développée durant cette thèse est d'utiliser la carte en tant que canevas d'apprentissage, pour stocker des informations géoréférencées sur la performance des différents capteurs équipant le véhicule, au cours de trajets répétitifs. Pour cela, une localisation robuste, relative à la carte, est nécessaire au travers d'une méthode de *map-matching*. La problématique principale réside dans la différence de précision entre la carte et le positionnement GNSS, créant des situations ambigües. Durant cette thèse, un algorithme de map-matching a été conçu pour gérer ces ambigüités en fournissant des hypothèses multiples lorsque nécessaire. L'objectif est d'assurer l'intégrité de l'algorithme en retournant un ensemble d'hypothèses contenant l'hypothèse correcte avec une grande probabilité. Cet algorithme utilise les capteurs proprioceptifs dans une approche de navigation à l'estime aidée d'informations cartographiques. Une procédure d'évaluation de cohérence, utilisant le GNSS comme information redondante de positionnement est ensuite appliquée, visant à isoler une hypothèse cohérente unique qui pourra ainsi être utilisée avec confiance dans le processus d'écriture dans la carte. L'utilisation de la carte numérique en écriture/lecture a été évaluée et la procédure complète d'écriture a été testée sur des données réelles, enregistrées par des véhicules expérimentaux sur route ouverte.

Contents

1 Introduction **1**
1.1 Background . . . 3
1.1.1 Road Transportation . . . 3
1.1.2 Potential Solutions . . . 4
1.2 Intelligent Vehicles Functional Architecture . . . 10
1.2.1 Perception . . . 10
1.2.2 Navigation and Localization . . . 11
1.2.3 Control . . . 12
1.3 Problem Statement . . . 12
1.3.1 Purpose and Objectives . . . 13
1.3.2 Thesis Scope . . . 14
1.4 Contributions . . . 14
1.5 Thesis Contents . . . 15

2 Vehicle Localization on HD-Maps **17**
2.1 Introduction . . . 18
2.2 Global Navigation Satellite Systems . . . 18
2.2.1 GPS History . . . 19
2.2.2 Structural Components . . . 19
2.2.3 Reference Frames . . . 21
2.2.4 Operational Problems . . . 21
2.2.5 Position Calculation . . . 24
2.3 GNSS Protection Levels . . . 25
2.3.1 Aeronautic Integrity . . . 25
2.3.2 Road Transport Integrity . . . 27
2.3.3 HPL Availability . . . 29
2.4 Digital Road Maps . . . 29
2.4.1 Different Scales Available . . . 29
2.4.2 Existing Formalisms . . . 32
2.4.3 Semantic Information . . . 37
2.4.4 Map Accuracy Metrics . . . 38
2.4.5 Map Used in This Research . . . 40
2.5 Map-Matching Approaches . . . 43
2.5.1 Different Matching Principles . . . 44
2.5.2 Different Map Scales . . . 46
2.6 Experimental Setup and Functional Architecture . . . 47
2.6.1 Experimental Vehicles . . . 48
2.6.2 Sensors . . . 49

2.6.3 Middleware . . . 51
2.6.4 Functional Architecture . . . 52
2.7 Conclusion . . . 53

3 Simultaneous Map-Matching And Localization 55
3.1 Introduction . . . 56
3.2 Elements on Bayesian Sequential Estimation Theory . . . 57
3.2.1 Bayes' Theorem . . . 57
3.2.2 Bayesian Filtering . . . 58
3.2.3 Kalman Filtering . . . 60
3.2.4 Particle Filtering . . . 62
3.3 Simultaneous Map-Matching and Localization Using Particle Filtering 68
3.3.1 Particle Definition . . . 70
3.3.2 Inputs . . . 70
3.3.3 Output . . . 71
3.3.4 Initialization Step . . . 72
3.3.5 Main Loop: Updating State and Matching . . . 76
3.3.6 Likelihood Calculation . . . 78
3.3.7 Resampling . . . 84
3.3.8 Hypotheses Determination . . . 84
3.4 Experimental Results and Evaluation . . . 85
3.4.1 Experimental Setup . . . 85
3.4.2 System Initialization and Study of the PF Convergence . . . 88
3.4.3 System Performance in Tracking Mode . . . 91
3.4.4 Localization Results . . . 97
3.4.5 Map-Matching Consistency . . . 97
3.5 Conclusion . . . 99

4 Integrity and Decision Making 103
4.1 Introduction . . . 104
4.2 Integrity Monitoring Methods and Architectures . . . 104
4.2.1 Receiver Self Integrity Monitoring . . . 104
4.2.2 Integrity Monitoring Using GNSS and Additional Sensors . . . 107
4.3 Simultaneous Map-Matching and Localization Using Particle Filtering (SMMAL-PF) Coherence Checking Using Redundancy . . . 108
4.3.1 Working Hypotheses . . . 108
4.3.2 Proposed Architecture . . . 109
4.3.3 "Use/Don't Use" Decision . . . 109
4.3.4 Five Different "Use/Don't Use" Situations . . . 111
4.4 Comparison of Different Tests for Hypothesis Selection . . . 113
4.4.1 Deterministic Approach . . . 113
4.4.2 Covariance Estimation for Consistency Testing . . . 116
4.4.3 Consistency Test Based on a Chi-Squared Test . . . 118
4.4.4 Consistency Test Based on the Kullback-Liebler Divergence . 127
4.5 Graphical Illustration of the Results . . . 132
4.6 Conclusion . . . 134

5 Learning Map for Autonomous-Driving 135
5.1 Introduction . . . 136
5.2 Learning Strategy . . . 136
5.2.1 Online Learning . . . 136
5.2.2 Unsupervised Learning . . . 136
5.3 Adding Information in an Existing Map . . . 137
5.3.1 Need for Reliable Positioning . . . 137
5.3.2 Recording Structure . . . 137
5.3.3 Study of Possible Recorded Information . . . 138
5.3.4 Determination of Usability . . . 140
5.4 Repetitive Travels . . . 140
5.4.1 Accumulation of Data . . . 140
5.4.2 Processing Repetitive Information . . . 141
5.5 Efficient Writing In SQLite Database . . . 141
5.5.1 Direct Insertion Approach . . . 141
5.5.2 Direct Insertion With Two Tables . . . 142
5.5.3 Bulk Insertion . . . 145
5.5.4 Using Transactions . . . 145
5.5.5 SpatiaLite Performance Impact . . . 147
5.6 Results . . . 148
5.6.1 Positioning Usability . . . 148
5.6.2 Marking Detection . . . 149
5.6.3 Vehicle Speed . . . 149
5.6.4 Example of Functionality Usability Determination . . . 150
5.7 Conclusion . . . 151

6 Conclusion 153
6.1 Findings . . . 154
6.2 Perspectives . . . 156
6.3 Towards Autonomous Vehicles . . . 158

A Global Navigation Satellite Systems 161
A.1 GLONASS . . . 161
A.1.1 History . . . 161
A.1.2 Space Segment . . . 162
A.1.3 Control Segment . . . 162
A.1.4 User Segment . . . 162
A.2 Galileo . . . 163
A.2.1 History . . . 163
A.2.2 Space Segment . . . 164
A.2.3 Control Segment . . . 164
A.2.4 User Segment . . . 164
A.3 BeiDou-2 . . . 164
A.3.1 Space Segment . . . 165
A.3.2 Control Segment . . . 165
A.3.3 User Segment . . . 165

B Heudiasyc Datasets **167**
B.1 Acquisition Area 167
B.2 System Setup 167
B.3 Travel 1: "CI_Rocade" 167
B.3.1 Overview 167
B.3.2 Events of Interest 168
B.4 Travel 2: "Rocade" 170
B.4.1 Overview 170
B.4.2 Events of Interest 170

C MA-HUL Along-Track and Cross-Track Separation **173**
C.1 Raw Distance 173
C.2 Along and Cross-Track Separation 174
C.3 Results 175

List of Figures

1.1 Evolution of the number of road casualties (person who died within 30 days of a traffic accident) in France, from 1952 to 2016 (data from Observatoire National Interministériel de la Sécurité Routière ONISR [2018]). 4
1.2 Illustration of the Blind Spot Warning functionality (image from Renault [2018b]). 5
1.3 The Automatic Emergency Braking system can detect pedestrians (or obstacles) on the road and trigger an automatic braking maneuver (image from Renault [2018b]). 6
1.4 Renault Symbioz concept car [Renault, 2017]. A functional prototype exposes autonomous driving capabilities and Renault's vision of the automotive future. 8
1.5 Renault EZ-GO concept car [Renault, 2018a]. This concept exposes the concept of robot-taxi: the personal car is put aside for a shared mobility vision. 8
1.6 Autonomous vehicle simplified functional architecture as per Ibañez-Guzmán et al. [2012]. 10
1.7 The accuracy requirements for maps and localization for different applications (after Gilliéron et al. [2006]). 13

2.1 Map showing the different element forming the GPS control segment. (Image from `https://www.gps.gov/systems/gps/control/`). 20
2.2 WGS84 reference frame (image from International Civil Aviation Organization ICAO [2002]). 22
2.3 Example of multipath and NLOS conditions. In the first case, the direct signal reaches the receiver while for the latter, only the reflected signal does. (Image from [Groves et al., 2013]). 24
2.4 Geometric illustration of the different integrity metric with their signification (Image from [William J. Hughes Technical Center, 2017]). 27
2.5 Stanford diagram [Tossaint et al., 2007] illustrating the different integrity states. 28
2.6 An example of a macroscale map: roads in Compiègne, France. Image and data from Google Maps (© 2018 Google). 30
2.7 Point cloud from a lidar sensor can be considered as a microscale map. High density of information may not be practical for every use. After the data from *Paris-rue-Madame* database (MINES ParisTech 3D mobile laser scanner dataset from Madame street in Paris). 31

2.8 Mesoscale map: each lane is represented separately (in blue). Additional contextual information is included, such as lane markings (in orange). 32
2.9 Comparison between topological maps and geographic maps. Topological maps do not respect geographic reality, prioritizing readability over geographic fidelity. 33
2.10 Example of a shape point-based map. The discretization is clearly visible. More shape points are necessary to describe a curve with enough fidelity than a straight line. 34
2.11 Maps can contain outdated information: (a) Google Maps lacks a road that has been added by a user in (c) OpenStreetMap (in red). Both snapshots have been taken on the same date, along with (b) the Google Maps satellite view of the area. 35
2.12 Direction changes on highways are smoothed using clothoids. Example of the French N12 road, highlighted in blue. (© OpenStreetMap contributors). 35
2.13 Lanelets uses lane borders to represent the drivable lanes (image from [Ziegler et al., 2014]). 36
2.14 Lanelets provide a normal gradient to calculate *pseudo distances* to map elements to remove any discontinuity (images from [Ziegler et al., 2014]). 37
2.15 Two types of map accuracy errors are compared to the ground truth: the absolute error (in orange) is constant, while the relative error (in blue) varies. 39
2.16 Detail of the lane-level map of UTC *Centre de Transfert*, showing the *Seville* circuit in the top part. 40
2.17 Simplified UML diagram of the Link table, showing the essential elements. 41
2.18 Fuzzy rule describing the speed of the vehicle by three classes: zero, low and high. The definition of the class zero illustrates well the vagueness of the fuzzy logic (image from Quddus et al. [2006a]). . . . 45
2.19 Positioning likelihood field (right) generated from a road map (left) (image from [Gustafsson et al., 2012]). 46
2.20 IRIS test vehicle, based on a Renault Fluence Z.E. 48
2.21 Renault Fluence Z.E. 48
2.22 One APAChE test vehicle, based on a Renault Zoé. 49
2.23 Functional architecture presenting all the elements developed during this thesis and their relationships with each other. 52

3.1 Ambiguous matching situation caused by the use of an HPL for positioning. Some possible MM solutions are highlighted. 56
3.2 Using a set of particles, PF allows the representation of arbitrary distribution (a Gaussian distribution in this figure). The particles evolve in the state space and a weight w^i allows to give a likelihood for every point and thus shape the distribution. In this drawing, the particles are distributed along a grid. In practice, they are randomly spread in the state space. 63
3.3 Illustration of the resampling of a particle filter. 65

3.4 Graphical illustration of the low variance resampling. The rectangles represent the old particles, their width is proportional to their weight. Circles are the new particles drawn from the old set. For instance, particle 2 is duplicated 3 times and particle 3 is eliminated. 66
3.5 Flowchart of the map-matching process. The inputs are denoted by dashed lines/boxes. Based on a particle filter, the algorithm takes as inputs a GNSS fix with an HPL and the car dead-reckoning. The process is separated in two sections: the initialization which is heavy-computational and the main loop being highly efficient. Exteroceptive information is used during the likelihood calculation step. 69
3.6 The coordinates correspond to the Cartesian coordinates in (x^i, y^i), with axes pointing respectively to the east and north. ψ^i is the angle form with the east axis, respecting the trigonometric convention. . . . 70
3.7 Example of a particle initialization: colors denote the matched link. The initial heading corresponds to the matched link. Some particles are far from the links due to the 50 m HPL (this a high conservative value here), but will potentially be quickly eliminated during a resampling step. 73
3.8 Comparison between the two initialization methods. While it may be acceptable for small diameter, the first procedure (Fig. 3.8a) quickly loses efficiency when the diameter grows. The particle distribution is much more homogeneous using the second spiral generation procedure (Fig. 3.8b), independently of the diameter. 75
3.9 Particle at time $t = 1$ (M_1) remains on the current segment $[AB]$, while at time $t = 2$ (M_2) it has left it. This is determined by calculating the ratio r described by (3.58). 77
3.10 Lane bifurcation where particle cloning occurs: the particle M_{AB} is duplicated into M_{BC}, M_{BE} and M_{BD} in order to do a thorough search. 78
3.11 Trapezoidal likelihood function for orthogonal distance (based on the French regulations relating to the width of roads). The center of car is equally likely to be in any lateral position on the roadway in the interval [-1.75; 1.75]. 80
3.12 Reference frames used by the camera for describing lane markings. Dashed orange lines represent lane marking data from the map. Camera detections are represented in blue. The lane centerline is in black. Please note that $\overrightarrow{y_c}$ points forwards. 81
3.13 A lane change situation: the left lane particles get out of the road and are therefore eliminated by the filter, leaving only the originally on the right-side particles viable: a unique hypothesis is finally determined after the lane change. 83
3.14 The set of particles (on the left) is aggregated into hypotheses corresponding to the mean position (on the right, computed by (3.67)). The size of the hypothesis is proportional to its weight (computed by (3.68)). 84
3.15 The two test travels used to study the filter's performance in gray. The first (Fig. 3.15a) has open sky conditions during all the sequence while the second (Fig. 3.15b) is a more varied scenario, crossing a peri-urban area (images from © 2018 Google Maps). 86

3.16 Image from the contextual camera (a front-facing dashboard camera) showing clearly the position of the car on the roadway. It is used to determine manually the map-matching ground truth. 88
3.17 The filter initializes the particles to fill the area corresponding to the HPL (15 m in Fig. 3.17a). The particles off-road are immediately eliminated by the likelihood calculation (see Fig. 3.17b). 89
3.18 Clusters of particles forming after 7 s of execution. These are due to i) the absence of longitudinal information and ii) a too small a number of particles. 89
3.19 Starting positions (blue circles) used for the initialization position study. The brown stars correspond to the time $t = 106\,s$ and $t = 119\,s$. 90
3.20 Cluster formation study for different start times (i.e. different start position). HPL=15 m. In all instances, the number of clusters reach 1 around the same time (110 s): cluster elimination is independent from the start time. 90
3.21 Study of HPL influence on clustering. 92
3.22 Evolution of the hypothesis weights on the open sky dataset. Two majors hypotheses are prominent.The strong variations happening at $t = [140, 175]$ s are caused by a roundabout. The correct hypothesis is the blue one at $t = [0, 275]$ s and the red one from $t = 275$ s till the end of the travel. 93
3.23 Details of the test travel. 94
3.24 Evolution of the hypothesis weights on the second dataset. Multiple hypotheses are present, mainly due to the intersection encountered (including roundabout exits), but two majors hypotheses are also prominent in this recording. 95
3.25 The system travels through areas prone to GNSS multipath, but the filter is unaffected as the hypothesis weights stay stable. 95
3.26 Study of a lane forking event. 96
3.27 Results of a classical DR-only solution (in blue) compared to the map-aided odometry (in orange). 98
3.28 Histogram of the proportions of inconsistent filter for each execution. The majority stays completely consistent through their execution. The histogram below represents the same data but zoomed for better readability. 100
3.29 Cumulative histogram of the proportions of inconsistent filter for each execution. The ratio of consistency loss stay small, representing 95% of the time less than 10% of a single travel. 100

4.1 Illustration of the statistical test on the NSSE, using χ^2 distributions. It is characterized by a probability of false alarm p_{FA} and of miss-detection p_{MD}, set by determining the test threshold T and the non-centrality parameter δ_T (image from [Drevelle, 2011]). 106
4.2 Using redundancy is a key element for functional safety. Using different estimation models limits the common failure modes (image from [Nebot and Durrant-Whyte, 1999]). 108

4.3 Organization of the coherence checking procedure. It takes as input the multiple hypotheses from the map-matching and the GNSS fix. Each hypothesis is then compared to the GNSS to take a decision on the usability of the positioning system. 110
4.4 Illustration of the different cases that can occur with a correct classification. 112
4.5 Illustration of the different cases that can occur with an incorrect classification. 113
4.6 Biased GNSS position (the GNSS track over time is displayed by white triangles, a deviation is clearly visible) along with the associated matching hypotheses at the current time index (colored triangles). The MA-HUL is denoted by the red line, corresponding to the distance between the GNSS position and the farthest matching hypothesis. 114
4.7 The MA-HUL (in blue) appears as an upper bound of the GNSS error (in green, computed using data gathered by an IMU). 115
4.8 Flagging a correct matching hypothesis as inconsistent is possible if the GNSS position present a large bias. In the situation depicted here, both hypotheses (in purple and yellow) are eliminated as the GNSS (in white) present a strong error. This is an instance of the case depicted in Fig. 4.4b. 120
4.9 GNSS error evolution along the mixed travel, measured using an embedded IMU as reference. The two types of environment are easily distinguishable: peri-urban presents higher and more unstable error, while open sky conditions yields stable smaller error. 121
4.10 Statistics about the false alarm rate distribution for the mixed travel, eliminating epoch with high GNSS error. 122
4.11 Evolution of the hypothesis weights. Two hypotheses are prominent most of the time, reflecting the nature of the traveled roads (2-lane roads). The ground truth is drawn in dark blue, while the other colors denote different adjacent hypotheses. 123
4.12 Summing the two highest weights shows that the filter considers two main hypotheses most of the time. 124
4.13 Squared Mahalanobis distances for the matching hypotheses (the correct hypothesis shown in dark blue). The yellow line denotes the critical value for $p_{fa} = 0.01$. 124
4.14 Screenshot of the situation at $t = 127\,s$. A biased GNSS position causes the two matching hypotheses, including the correct one, to be rejected. 125
4.15 Rejection classification by cause, for the correct matching and the others. A point indicates that the corresponding hypothesis was rejected at this time step for the given reason. 126
4.16 "Use/Don't Use" decisions by the consistency test. It can be seen that misdetections ("Use Wrong") are rare. Positioning system availability corresponds to "Use Good". 127
4.17 Evolution of the number of coherent hypotheses contained in the final set after the coherence tests. 128

4.18 Three variations of the KLD on the same data. From top to bottom, the divergence is computed using, first the hypothesis as reference, then the GNSS. The third KLD is the half sum of the previous two. . 130
4.19 KLD computed with the GNSS as reference, along with the Mahalanobis Distance graph for reference. 131
4.20 Maps representing the results of the χ^2 usability test, on a single test travel (mixed conditions). 133

5.1 UML exposing the *Recording* classes. A composition with a map element is mandatory. Each learning recording type (implementing their own specific data fields) inherits from a base case *Recording*. . . 139
5.2 SQLite INSERT performance, writing the rows one by one in a single table. 143
5.3 SQLite INSERT performance, writing the rows one by one in two distinct tables. 144
5.4 SQLite INSERT performance, writing the rows in bulk, 500 at a time, in a single table. 146
5.5 Map representing the data from the "Use" SQLite table, showing the area where the positioning system has been determined usable. 148
5.6 Map representing the data from the lane marking detection table showing the areas with good quality markings detection and areas without. 149
5.7 Maps representing the recorded vehicle speed on the learning travel and its evolution after two passes. 150
5.8 Map representing the AD-compliant areas, determined using different requirements, in a logical AND operation. 151

A.1 Map showing the different element forming the GLONASS control segment (image from CGSIC Meeting). 163

B.1 Overview of the travel (ground truth): 4 km-long in Compiègne, France. It is representative of a peri-urban trip passing in areas with multipath conditions (near high-rise buildings) as well as areas with open sky (urban bypass): see Section B.3.2 for more details, the "Bx" labels refers to events described in the following corresponding figures (map data from OpenStreetMap). 168
B.2 Image from the contextual camera (front facing dashboard camera). . 169
B.3 Details of the GNSS fixes, during GNSS loss at $t = [93, 102]$ s, with multipath-induced errors. The car was traveling southward (right to left). 169
B.4 Details of the GNSS fixes, during multipath at $t = [165, 183]$ s. Note that the car stayed constantly on the right lane, the variations were caused by multipath. 169
B.5 Details of the GNSS fixes, during lane change on the bypass at $t = 304$ s. 170
B.6 Details of the GNSS fixes, reaching an exit ramp at $t = 455$ s, and taking it. 170
B.7 Overview of the travel (ground truth): 2 km-long on Compiègne bypass, France. This dataset is aimed to validate the behavior of the algorithm in nominal conditions. 170

B.8 Lane change event during travel 2. The car was traveling westward (right to left). 171

C.1 Biased GNSS position (the GNSS track over time is displayed by white triangles, a deviation is clearly visible) along with the associated matching hypotheses at the current time index (colored triangles). The MA-HUL is denoted by the red line, corresponding to the distance between the GNSS position and the farthest matching hypothesis. 174

C.2 Particles spread due to dead-reckoning drift in a PF. Depending on the situation, the spread can be more prominent in the *along-track* direction (C.2a) or in the *cross-track* direction (C.2b). 175

C.3 Separating the MA-HUL into cross-track and along-track components allows to characterize more finely the error. Here, the cross-track error is much smaller than the raw HCL. 176

C.4 The MA-HUL (in blue) appears as an upper bound of the GNSS error (in green). 177

C.5 Separating the MA-HUL in two: along-track (in cyan) and cross-track (in green). The camera detections are used, improving the cross-track uncertainty. 177

C.6 Histograms showing the distributions of the MA-HUL (on the left) and their cumulative density functions (on the right). The raw value is presented on top and the cross-track is at the bottom. 178

List of Tables

1.1 Identified causes of the recorded road casualties in France in 2017 [Observatoire National Interministériel de la Sécurité Routière ONISR, 2017]. A single accident may have multiple causes. 5
1.2 Levels of driving automation as defined by the SAE International J3016 standard [SAE On-Road Automated Driving (ORAD) committee, 2018] (Copyright © 2018 SAE International). 9

2.1 Frequencies used as of 2018 for GPS positioning signals, f_0 represents the master clock frequency (10.23 MHz). 20
2.2 List of sensors and integration status in the cars. 50

3.1 Typical values for the filter's parameters 87
3.2 Matching metrics on the set of matching hypotheses. 97
3.3 Matching metrics on the set of matching hypotheses after 400 executions. 99

4.1 Rejections based on χ^2 test and weight threshold. Percentages are calculated relatively to the total number of time steps. 125
4.2 "Use/Don't Use" classification rates, expressed in percentage of the total number of filter execution steps. 127
4.3 Rejections based on χ^2 test using the KLD measure. Percentages are calculated relatively to the total number of time steps. 132
4.4 "Use/Don't Use" classification rates, expressed in percentage of the total number of filter execution steps. 132

5.1 Mean insert time and corresponding performance for the different insert strategies: transactions outperform the two others. 147

Glossary

ABS Anti-lock braking system. 11, 70

ACC Adaptive Cruise Control. 6

AD Autonomous Driving. 136

ADAS Advanced Driving Assistance Systems. 3–7, 32, 70, 154

AEB Automatic Emergency Braking. 6

ASIL Automotive Safety Integrity Levels. 9

AVM Around View Monitor. 10

BSW Blind Spot Warning. 4, 5

CAN Controller Area Network. 49, 70, 85, 97, 155

DIA Detection, Identification, and Adaptation. 104, 107

DR Dead-Reckoning. xvi, 45, 56, 85, 97, 98, 109

ECEF Earth-Centered, Earth-Fixed. 21

EKF Extended Kalman Filter. 45, 62

ENU East-North-Up. 42

ESP Electronic Stability Program. 11, 70

FDE Fault Detection and Exclusion. 104, 107

FDW Forward Collision Warning/Mitigation. 5

FIS Fuzzy Inference System. 44

FOC Full Operational Capability. 19

GIS Geographic Information System. 29, 137

GMM Gaussian Mixture Model. 47

GNSS Global Navigation Satellite Systems. xv–xviii, 11, 12, 14, 18–27, 29, 38, 43–46, 50, 52, 53, 56, 68–70, 72, 73, 75, 78, 80, 81, 83, 85, 87, 88, 93–95, 99, 101, 104, 106–123, 125, 127, 129–134, 138, 140, 141, 154, 155

GPS Global Positioning System. xiii, xxi, 18–21, 50, 68, 133

GPST GPS Time. 21

HAL Horizontal Alert Limit. 25–27

HPE Horizontal Positioning Error. 26

HPL Horizontal Protection Level. xiv–xvi, 26, 27, 29, 52, 56, 69, 70, 72–75, 83, 84, 87–89, 91–93, 101, 108, 109, 115, 120

HUL Horizontal Uncertainty Level. 26, 27, 113–115

IBPL Isotropy-Based Protection Level. 28, 29

IMU Inertial Measurement Units. xvii, 11, 115, 121

INS Inertial Navigation Systems. 38, 50

ISO International Organization for Standardization. 9

KF Kalman Filter. 47, 60–62

KIPL Kalman Integrated Protection Level. 29, 87

KLD Kullback-Liebler Divergence. xviii, xxi, 127–132, 134

LDW Lane Departure Warning. 4–6, 51

LKA Lane Keeping Assist. 6, 140

LOS Line-Of-Sight. 23

MA-HUL Map-Aided-HUL. xvii, 114–116, 134

MCS Master Control Station. 20

MEO Medium Earth orbit. 19

MHMM Multi-Hypothesis Map-Matching. 47

MHT Multi-hypothesis Tracking. 62

MM Map-Matching. xiv, 56, 68, 69, 71, 72, 76, 78, 97, 99

MMSE Minimum Mean-Square Error. 60

NLOS Non-Line-Of-Sight. 23

NSSE Normalized Sum of Squared Error. xvi, 105, 106

PDF Probability Density Function. 57–59, 62, 64

PF Particle Filter. xiv, 52, 56, 57, 62–66, 68–72, 76, 78, 87, 88, 97, 99, 101, 113, 115, 157

POI Points Of Interest. 43

PPS Precise Positioning Service. 19

RAIM Receiver Autonomous Integrity Monitoring. 26, 104, 107, 111, 113–115

RTK Real-Time Kinematic. 38, 50

SA Selective Availability. 19

SAE American Society of Automotive Engineers. xxi, 7, 9

SBAS Satellite-Based Augmentation Systems. 28

SIR Sequential Importance Resampling. 78

SIS Sequential Importance Sampling. 64, 65

SLAM Simultaneous Localization and Mapping. 81, 97, 98, 136

SMMAL-PF Simultaneous Map-Matching and Localization Using Particle Filtering. x, 87, 88, 97, 99, 101, 103, 108, 109, 111, 119

SPS Standard Positioning Service. 19

SQL Structured Query Language. 41, 72, 138, 141, 145, 147, 151, 155

SV Space Vehicles. 19

TIR Target Integrity Risk. 25, 26, 29

TJP Traffic Jam Pilot. 6

TOA Time Of Arrival. 18, 21, 24

TTA Time To Alarm. 25

TTC Time To Collision. 51

UMAP Urban Mobility Advanced Platform. 40

UML Unified Modeling Language. xiv, xviii, 41, 139

UTC Coordinated Universal Time. 21

VAL Vertical Alert Limit. 25

WGS84 World Geodetic System 1984. xiii, 21, 22

Chapter 1

Introduction

Contents

1.1 Background 3
1.1.1 Road Transportation 3
1.1.2 Potential Solutions 4
1.2 Intelligent Vehicles Functional Architecture 10
1.2.1 Perception 10
1.2.2 Navigation and Localization 11
1.2.3 Control 12
1.3 Problem Statement 12
1.3.1 Purpose and Objectives 13
1.3.2 Thesis Scope 14
1.4 Contributions 14
1.5 Thesis Contents 15

Foreword

This thesis took place within the scope of SIVALab, a shared laboratory between Renault and Heudiasyc. It used experimental equipment provided by the Equipex Robotex (ANR-10-EQPX-44-0).

Mobility has always been an essential part of human development. Since ancient times our ancestors began to explore the world by foot, to become self-reliant, to start forming the first communities. Over the years, mankind has developed technologies enhancing mobility as populations grew and the need to travel further became part of daily life. Its ingenuity has resulted in the development of different means of assisted transport from animal-powered transportation (or wind-powered in a maritime context) to the invention of mechanical powered platforms following the industrial revolution. From this point, human development skyrocketed: railroads connected distant cities, in different countries, enabling further developments. Ships and planes later completed the transportation technologies to create an interconnected, globalized world.

The invention of the motorcar has fundamentally changed our way of life and our cities, while transforming our economic outlook. This has resulted in the development of large urban centers with thousands and thousands of vehicles converging daily to them. However, the limits to this relentless growth has resulted on saturated roads, traffic accidents, pollution, dense road networks, among other problems. Today, the automotive industry is undergoing a transformation triggered by the introduction of computer science as an enabler for multiple functions, which together with the availability of advanced perception sensors and high-performance computing is allowing to implement different navigation functions with an increasing degree of intelligence. This have led to progressing levels of autonomy in today's cars. The ultimate aim is to reach the development of connected autonomous vehicles: systems that could navigate safely under full computer control [Thrun, 2010].

Autonomous vehicles represent a very active field of research, both in industry and academia. These complex systems need to perceive their immediate environment, in order to understand their situation with respect to it, to finally being able to decide which action to follow and command the vehicle. A very important source of information is the road network map, storing prior information about the current and future surroundings of the vehicle. This provides contextual information to help the vehicle "making sense" of the perceived environment, improving situation understanding and thus decision-making. This has proven to be fundamental for autonomous vehicle navigation, hence the development of what is known as High-Definition maps (HD-maps) which include accurate information on the road geometry, together with attributes facilitating the navigation task [Filliat and Meyer, 2003]. However, to use properly these maps, it is fundamental to associate the current vehicle position to their projection on these maps without any ambiguity. Failure to do this would result in safety critical situations.

Moreover, prior maps need to be corrected or enriched on a regular basis, due to the manner they have been constructed and the dynamic nature of road networks (e.g., geometric errors, road works, changes in their structure [Zinoune et al., 2012]).

Autonomous vehicles benefit from advanced exteroceptive sensors, allowing data acquisition which can be transformed into features for navigation purposes. These can be annotated in a prior map, enriching or even correcting it. For this purpose, it is fundamental to localize the vehicle precisely and to associate it with the prior map: this process is called map-matching. In addition, it is important to detect discrepancies that might exist between the stored maps and the observed/estimated map features. One caveat of the perception system is its varying performance depending on various factors. Notably, a discrepancy can sometimes be observed in function of the location. Thus, it is important to assess this location-dependent performance variation.

This thesis addresses the localization problem, applied to the use of HD-maps for autonomous vehicle navigation. In addition, it examines the degree of confidence of these position estimations as well as the possibility of enhancing the prior map by annotating it. It is a fundamental problem for autonomous vehicle navigation and for what is known as actuating driving assistance systems. This chapter introduces the context of this thesis, presenting the elements around which autonomous vehicles are built. The research problem is then stated, focusing on the need for an accurate localization to use correctly an HD-map, the tenet of the thesis and rationale of the use of the map as a learning canvas is presented. Finally, the main contributions of this thesis are included.

1.1 Background

1.1.1 Road Transportation

Progress in transportation of people and goods has been a driving force to our increased mobility and move toward the globalization of trade. However, progress on the deployment of vehicles, building of roads, or new energy resources has been shown to not be unlimited [Meadows et al., 2004]. This progress, however, did not come without drawbacks, with the emergence of different issues, namely, road safety, congestion, pollution and scarce land to host more vehicles.

More kilometers are driven everyday, by a growing population: road safety became a major issue worldwide. In France alone, $61,224$ road accidents were reported in 2017, killing $3,684$ people and injuring $76,840$ [Observatoire National Interministériel de la Sécurité Routière ONISR, 2017]. The number of road casualties in France increased rapidly from 1952 to 1973 (see Fig. 1.1), following the expansion of the personal vehicle market. By combining regulation and new safety features in modern vehicles, the number of accidents have been successfully reduced over the past years, reaching an historical low in 2013 with $3,268$ casualties. However, this reduction, since then, has stalled and casualties have started to increase again. New societal factors influence this rise, Barrios et al. [2018] point out that the emergence of ride hailing companies and their popularity resulted in an increase of the number of traffic accidents in the USA.

Different approaches are being explored to inverse this trend, including the deployment of actuating Advanced Driving Assistance Systems (ADAS), wireless connectivity between vehicles or with the infrastructure, and ultimately autonomous vehicles to remove the human error from vehicle navigation. A common denominator of all these approaches is the use of localization information and maps. The

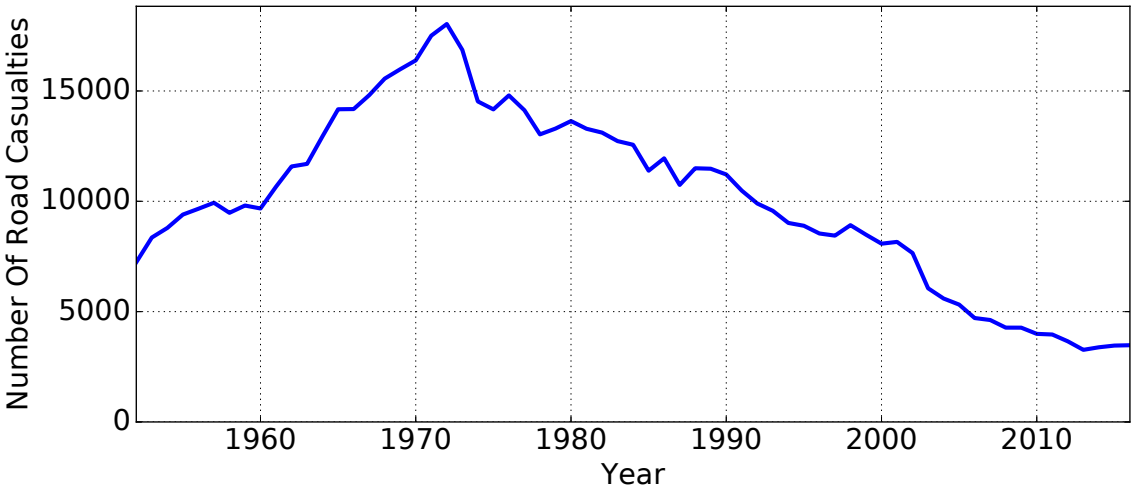


Figure 1.1 – Evolution of the number of road casualties (person who died within 30 days of a traffic accident) in France, from 1952 to 2016 (data from Observatoire National Interministériel de la Sécurité Routière ONISR [2018]).

deployment of autonomous vehicles may be considered as the ultimate solution that shall bring a major change to ground transportation. The tenet is that autonomous vehicles should provide accessibility, improve driver/passenger productivity and enhance safety.

1.1.2 Potential Solutions

Solutions reviewed within the scope of this thesis only include standalone systems, though the author is aware of those involving V2X (vehicle to vehicle & vehicle to infrastructure) communications technologies [Ibañez-Guzmán et al., 2010].

The potential solutions can be partitioned into three categories.

The first two relate to the conventional approach of developing separate driver assistance systems to either be informative, or to actuate directly on the vehicle. These are known as ADAS.

The third approach relates directly to the domain of intelligent vehicles: vehicles are designed as robotics systems whose behaviors are computer-controlled, substituting to the driver actions. These are known as Autonomous Vehicles.

In this section the main characteristics of these solutions are examined including some examples.

1.1.2.1 Informative ADAS

An informative ADAS provides information enhancing the driver's situation awareness in real-time. The system does not modify the car behavior, letting the driver take any action when needed. For example, Lane Departure Warning (LDW) informs the driver that the car is about to cross the border of the lane when there is no intention to change lanes (e.g., with a sound alert or steering wheel vibrations); Blind Spot Warning (BSW) monitors the blind spots for the presence of other vehicles, indicating danger in the lateral rear-view mirrors (see Fig. 1.2).

Informative ADAS consist mostly of systems monitoring the environment in order to improve the driver awareness of the car's surroundings. Some systems can also

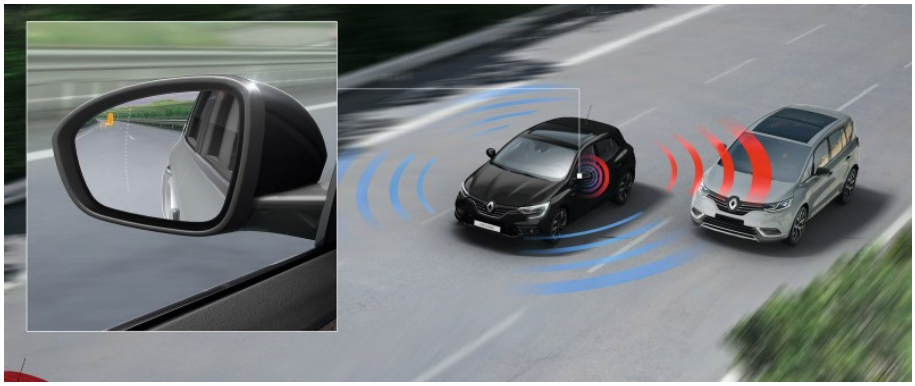


Figure 1.2 – Illustration of the Blind Spot Warning functionality (image from Renault [2018b]).

Table 1.1 – Identified causes of the recorded road casualties in France in 2017 [Observatoire National Interministériel de la Sécurité Routière ONISR, 2017]. A single accident may have multiple causes.

Accident Cause	%		%
Speeding	30	Driver fainting	4
Driving Under Influence (Alcohol)	20	Driver drowsiness	3
Right-of-way violation	12	Obstacle on the road	2
Other causes	12	Wrong-way driving	2
Undetermined cause	10	Lane changing	2
Driving Under Influence (Drug)	9	Insufficient safety inter-distance	1
Driver inattention	8	Vehicle-related fault	1
Unsafe overtaking	4	Use of phone	1

monitor the driver, watching signs of tiredness and loss of attention and encouraging the driver to take a break. Jermakian [2011] studied the impact of some ADAS on road casualties, namely BSW, LDW, Forward Collision Warning/Mitigation (FDW) and adaptive headlights. The author shows that one third of the recorded accidents in the US were relevant to at least one of the four considered ADAS. Statistically, these accidents could have been mitigated, if not completely avoided by cars equipped with such systems, showing the high potential of ADAS to increase road safety.

Table 1.1 presents a classification of the road casualties from data collected in 2016 by the French National Interdepartmental Road Safety Observatory [European Commission, 2016; Observatoire National Interministériel de la Sécurité Routière ONISR, 2017]. It appears that the human driver is responsible in a large portion of accidents [Petridou and Moustaki, 2000]. While the most frequent causes are direct consequences of human decisions (e.g., speeding, Driving Under Infuence), a misunderstanding or unawareness of the environment takes a significant part in the accidents (e.g., driver failing to see another car). This confirms the potential of ADAS for reducing road casualties by enhancing the driver's situation awareness, especially in difficult situations, for example at night [Plainis et al., 2006].

Figure 1.3 – The Automatic Emergency Braking system can detect pedestrians (or obstacles) on the road and trigger an automatic braking maneuver (image from Renault [2018b]).

1.1.2.2 Actuating ADAS

ADAS can also take a more prominent part of the driving task by directly actuating some of the car's elements without the intervention of the driver. It can range from switching on automatically the wipers and headlights when needed, to activating the braking system in case of emergency (Automatic Emergency Braking (AEB)). The latter are the ADAS that can improve road safety significantly: sudden events, such as a pedestrian unexpectedly crossing the road (see Fig. 1.3), can be managed more efficiently than by a human driver, especially considering the reaction time: ADAS can take over in dangerous situations in which an average human driver may not have the time to react fast enough.

Actuating ADAS frequently correspond to informative ADAS extended with an actuation. For example, Lane Keeping Assist (LKA) extends LDW with an action on the steering wheel to maintain the car inside its lane. They can also be an improvement of existing technology: Adaptive Cruise Control (ACC) extends the regular cruise control by automatically adjusting the vehicle speed in order to respect the inter-distance with the preceding vehicle. A technical separation is often made between lateral systems acting on the steering wheel (e.g., LKA), and the longitudinal systems acting on the throttle/braking systems (e.g., ACC). Finally, ADAS also helps in comfort functions such as automated parking systems, or Traffic Jam Pilot (TJP) that controls automatically the successive accelerations and braking in these situations.

All these functionalities are based on an advanced understanding of the road environment. This is only possible by equipping the vehicle with different types of sensors (see Section 1.2.1). ADAS shall address more and more complex road situations. For this purpose, a high level of contextual understanding of the environment is required. Experience has shown that even when using high-fidelity sensors such as stereo-cameras, LiDAR and Radar, their raw data or classified information (e.g., vehicles, pedestrians, etc.) are insufficient to facilitate machine understanding. This issue is addressed by introducing maps providing contextual information such as

the presence of certain road structures (e.g., lane markings, guard rails), see Section 2.4 for more details. This yields interesting results in the case of safety for road intersections [Armand et al., 2014].

Introducing more ADAS into a vehicle implies a cost that must be taken into account, especially for a generalist vehicle manufacturer. On the small to medium passenger car segment, ADAS can represent a more consequent part of the cost of a vehicle, making them less competitive compared to premium car manufacturers. The goal is therefore to balance the functionalities and the cost, without sacrificing performance.

1.1.2.3 Autonomous Vehicles

Anderson et al. [2016] studied the benefits and potential of Autonomous Vehicles and their power of transformation notably in terms of road safety. In the long-term it is envisaged that autonomous vehicles will enhance safety, the assumption being that most accidents are caused by human errors, automation would remove this major cause of safety problem.

The introduction of autonomous cars will be progressive as experience is acquired and solutions refined. Initially automation will be *driver-centric*, i.e. different levels of automation will facilitate the driver's tasks [Ibañez-Guzmán et al., 2012]. The driver, under different conditions, will remain as part of the vehicle control loop.

Once mature, the technology will have to be accepted by society. First, the market penetration will most likely take a long time as the automotive industry remains very conservative. Although, due to the pressure coming from the new entrants, often software companies (e.g., Tesla & Google), a rapid transformation is undergoing. But different concerns exist with regards to the deployment of autonomous vehicles and these need to be accepted by society in order to succeed [Schoettle and Sivak, 2014].

Car manufacturers take into account this issue by working on the Human-Machine Interface. For example, user experience is one of the main focuses in Renault's concept car Symbioz, showing autonomous driving capabilities in a functional prototype (see Fig. 1.4). User trust must be gained in order for the autonomous drive to spread. For this, every element of the system must be as robust as possible, from the sensors to the control algorithm.

Another vision of the autonomous car is the robot vehicle: in this use case, the driver disappears to become a simple passenger of the fully autonomous car. EZ-GO is Renault's concept car for this *passenger-centric* vision, emerging from the need to rationalize mobility, by using shared transportation means, eliminating the personal vehicle (see Fig. 1.5). While it imposes more trust from the user as no driver is present, the robot vehicle may be more easily accepted, as it is closer to existing autonomous public transportation (e.g., automatic subways). Even though it is not as controlled as a enclosed railway, robot vehicle experiments are currently conducted in secured areas, reinforcing the trust in the system.

1.1.2.4 Levels of Automation

The levels of automation for road vehicles have been defined by the American Society of Automotive Engineers (SAE) [SAE On-Road Automated Driving (ORAD)

Figure 1.4 – Renault Symbioz concept car [Renault, 2017]. A functional prototype exposes autonomous driving capabilities and Renault's vision of the automotive future.

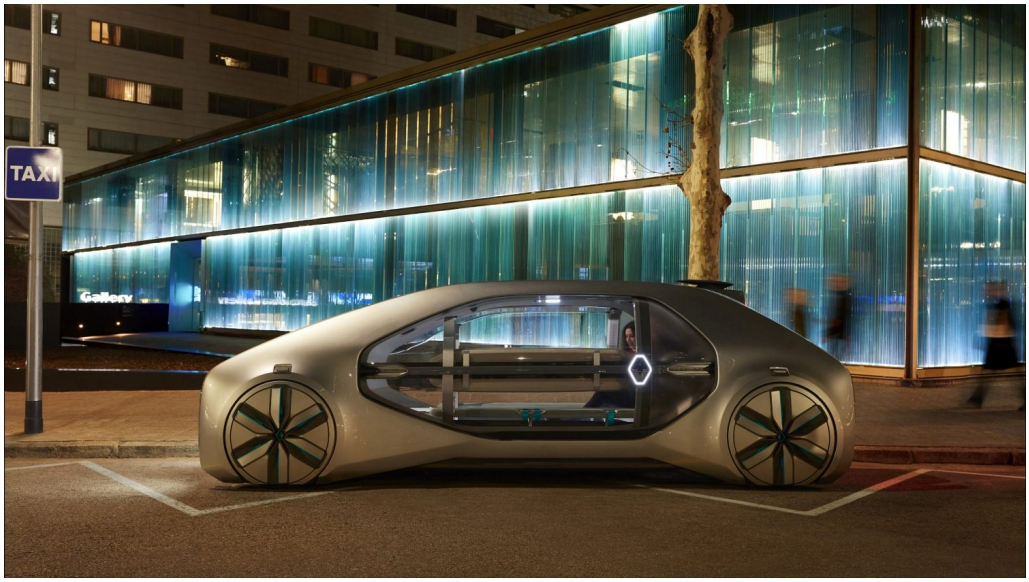


Figure 1.5 – Renault EZ-GO concept car [Renault, 2018a]. This concept exposes the concept of robot-taxi: the personal car is put aside for a shared mobility vision.

Table 1.2 – Levels of driving automation as defined by the SAE International J3016 standard [SAE On-Road Automated Driving (ORAD) committee, 2018] (Copyright © 2018 SAE International).

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

committee, 2018]. Six levels of automation are proposed and have been accepted worldwide. Table 1.2 presents a summary of these levels.

The lowest level is 0, where no autonomous functionality at all is active and the human driver is therefore responsible for all driving tasks. On the other side of the scale, level 5 qualifies a vehicle as fully autonomous, with no need for human intervention. The intermediate levels describe semi-autonomous steps, introducing progressively more functions while reducing the human intervention. The need for robustness increases along with the level of autonomy, as the system can rely less and less on the human driver taking over in case of problem. The system must be designed to respect an error threshold, determining nominal conditions. More importantly, it must be able to detect if this threshold is not respected: this is the principle of integrity monitoring.

Functional safety studies the system operation, especially its behavior facing errors. The International Organization for Standardization (ISO) published the ISO 26262 standard [ISO, 2011], defining the scope of functional safety for automotive applications. Terminology is defined, notably the *Functional safety* itself as the "absence of unreasonable risk due to hazards". Automotive Safety Integrity Levels (ASIL) are then used to characterize the system in terms of risk, scaling from A to D, D being the most critical.

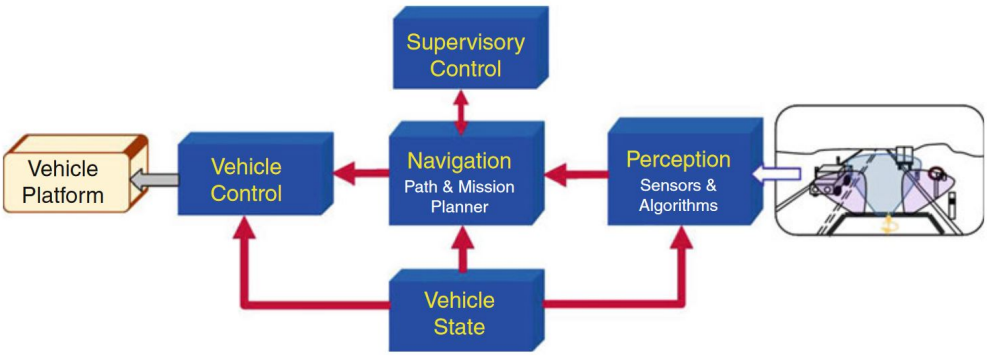


Figure 1.6 – Autonomous vehicle simplified functional architecture as per Ibañez-Guzmán et al. [2012].

1.2 Intelligent Vehicles Functional Architecture

An intelligent vehicle is composed of different components, essential for a safe operation. Ibañez-Guzmán et al. [2012] presents a simplified functional architecture for the autonomous vehicles (see Fig. 1.6).

1.2.1 Perception

An intelligent vehicle must pay constant attention to its immediate environment. This is the role of the perception system, composed of all the sensors embedded in the car. They can be grouped into two main categories.

1.2.1.1 Exteroceptive

Exteroceptive sensors are used to gather information on the surroundings of the vehicle. Different technologies are available, the most commonly used are:

- **Cameras**: a front facing camera is often used to monitor the area the car is heading to. Some systems also provide a 360° view around the vehicle (e.g., Around View Monitor (AVM)). Multiple cameras can be used, for example in stereovision, or with different focal distances and field of view, in order to cover a larger range. Different spectral ranges can also be used in the same manner (e.g., near infrared), completing the visible light camera, especially during adverse situations (e.g., night, dense fog).

- **Radars**: this technology uses radio waves to detect obstacles. Object ranging is the main goal of radars, different methods can be used: by measuring the time of flight of the radar pulse, the target distance can be calculated. Frequency modulation can also be used to this end. More advanced system can also use Doppler measurements in order to estimate the target's speed. Radars are an interesting supplement to cameras as they are not perturbed, for example by light sources by which a camera could be blinded.

- **Lidars**: similarly to radars, lidars (short for "light radar") use laser for ranging purposes. They usually perform in infrared light, at an energy level harmless to the human eye. The measurement is also based on time of flight calculation.

Using an array composed of numerous lasers, lidars usually return dense point clouds. Additionally, the reflectance of the objects can be measured by some sensors. A major drawback can be the computing power needed to process the dense point cloud that can be critical in an embedded environment.

This large panel of sensors is now available for automotive applications thanks to a miniaturization of the systems, enabling their integration on commercial vehicles. Using a variety of sensing principles is also interesting to be able to cope with different conditions: sensors operating with different physical principles can complete each other to provide a more complete view of the environment.

1.2.1.2 Proprioceptive

Knowledge about the car itself is also very important. Proprioceptive sensors are oriented towards the vehicle itself, providing information about the vehicle's dynamics (e.g., wheel rotation speed, acceleration, yaw rate). These sensors are already present in stock passenger vehicles, as they are needed for basic systems operations. For instance, Anti-lock braking system (ABS) requires knowledge of the vehicle's speed and Electronic Stability Program (ESP) requires the yaw rate and steering wheel angle. This odometry information is necessary to estimate the car's trajectory.

The stock on-board sensors are often low-cost, hence with a limited accuracy. It is sufficient for normal car operation, however the sensor uncertainty must be taken into account for advanced uses. High-accuracy Inertial Measurement Units (IMU) are also available but their high cost limits their use in the automotive domain to professional applications (e.g., road surveying, research).

1.2.1.3 Multi-sensors Data Fusion

The number of data sources keeps increasing. In order to process all this information, a data fusion [Bar-Shalom et al., 2011] step is necessary in order to eliminate the data imperfections by combining all sources. This research domain is extensively studied [Khaleghi et al., 2013] and solutions for eliminating the data uncertainty are based on a variety of theories, such as probabilistic methods [Mitchell, 2012] or evidential theory [Shafer, 1976]. Note that sensors are not the only sources of information available for fusion: digital road maps also contain valuable prior data (see Section 1.2.2.2).

1.2.2 Navigation and Localization

The goal of navigation systems in general is to reach a destination from the current position. This navigation task requires a situation understanding based on the information gathered from the sensors. This enables a decision making step, performing a path planning in the environment. Perception data are mostly useful for local path planning. At a larger scale (i.e. a complete travel), other sources of information must be used.

1.2.2.1 Global Navigation Satellite Systems

Global Navigation Satellite Systems (GNSS) use satellites in orbit around Earth in order to compute an absolute position, with coordinates expressed in terms of lon-

gitude, latitude and altitude (internally it is the ellipsoidal height that is computed and then transformed in altitude with respect to a geoid) [Kaplan and Hegarty, 2006]. This is in contrast to perception that only provides positioning relative to the system. The technology is widely available to the automotive market as GNSS receivers can be found for a low cost. Guidance systems rely on GNSS for positioning and thus embed such receivers. Positioning error can reach tens of meters, which is acceptable for this kind of application.

1.2.2.2 Digital Road Map

The absolute position of the vehicle on its own does not provide much information. To be meaningful, it has to be used in conjunction with a digital road map. This is the element containing contextual information about the road network, making the junction between the absolute positioning and the sensors data gathered locally by the car. The road map provides prior information about the road network without being limited by the range of perception sensors. It can then provide the system with information in advance spatially (and thus temporally) that would be unreachable by other sensors. It is therefore a very interesting source of information, enabling some anticipation.

Depending on its level of details (see Section 2.4.1), the map can be used exclusively for high-level route planning (low-definition maps, as embedded in driver guidance systems) or to provide rich details about the road network (high-definition maps). In the latter case, the accuracy of such maps is higher than the GNSS receiver's. To be usable, the absolute position (e.g., GNSS coordinates) must be transposed into the map coordinate system. This is the map-matching procedure. Problems can arise when a large discrepancy is present between the GNSS accuracy and the map definition.

1.2.3 Control

The vehicle control component is the last element that interfaces with the mechanical parts of the vehicle. With the data provided by the navigation system, the control system actuates the vehicle in order to respect the planned path. Different strategies can be applied, generally using close-loop algorithms (observing the vehicle state to correct the input command).

Another control component supervises the execution of the system, monitoring the system performance, notably in terms of uncertainty. It is in charge of the system safety and must act in a fail-safe way, that is to say interrupting the vehicle safely before it reaches dangerous situations. This system gets more and more important with the increase of the autonomy level reached by the cars, as direct supervision from a human driver tends to disappear.

1.3 Problem Statement

The challenge of accurate positioning rises inherently with the use of digital road maps for intelligent vehicles. It occurs in two main steps, the first being the creation of the map: a post-processing of raw data from road surveys has to be performed in order to eliminate potential errors and create an accurate map. This offline

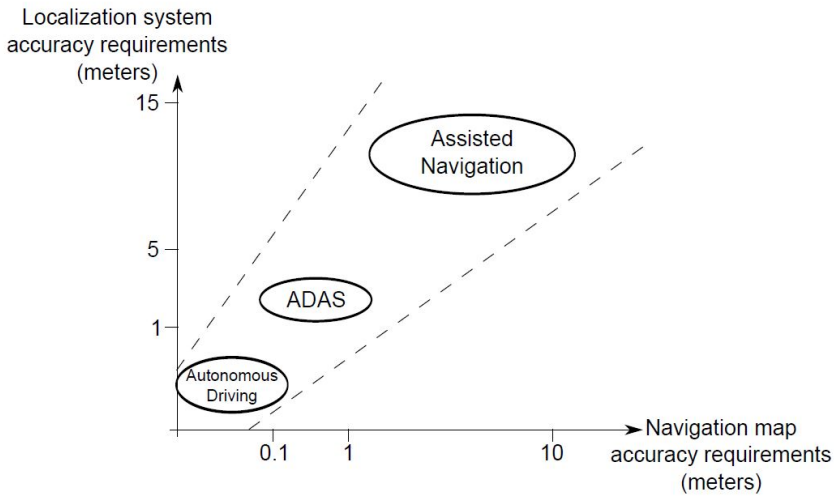


Figure 1.7 – The accuracy requirements for maps and localization for different applications (after Gilliéron et al. [2006]).

procedure is necessary to process the high volume of data and provide the best accuracy possible, often using external data sources such as corrections provided by base stations.

The processing time is less of a problem than for the second step: the actual use of the map by the vehicle. The system needs most of the time map information in real-time, post-processing is therefore not an option. This problem is in itself twofold: a first localization must be performed in order to delimit a zone of interest in the map. It can be relatively coarse but it is then refined in the second part to be accurate enough to process the map information. This level of accuracy must then be maintained, correcting the errors coming from the positioning system as well as the map. These two steps represent the basis of the map-matching. It links the map and the vehicle's pose, mitigating the effect of different sources of errors.

Contrarily to the map creation step, real-time use of the map cannot beneficiate from post-processed external information, the positioning system can only rely on embedded sensors. Car manufacturers consider map errors as an important issue, with problems such as inaccurate or even outdated data [Zinoune et al., 2012]. This causes a will to be as independent as possible from them. But the quantity of information that maps can provide is considerable, disregarding them totally is thus not viable. A robust method must therefore be developed to cope with the erroneous data that can be encountered both in the positioning system and the map, and even correct them. As such, map-matching appears as a crucial element for intelligent vehicles to make the maps usable, as the accuracy requirements for maps and localization are getting more and more demanding [Gilliéron et al., 2006] (see Fig. 1.7).

1.3.1 Purpose and Objectives

The purpose of this thesis is to match the subject vehicle position on the stored digital navigation map. This must be done with a high level of certainty to ensure the correct use of the map, in order to provide client applications with correct information (e.g., providing the decision-making function with the correct contextual

representation of the perceived world).

Research thus seeks to propose a map-matching method applicable to the use of HD-maps. It must be capable to detect ambiguous situations where map-matching cannot be solved with absolute certainty. Doubt is not acceptable when correcting or enhancing existing navigation maps and must be detected as the additional information is written into the map, based on the map-matching result. Any wrong association in the road map would lead to the introduction of critical errors caused by misplaced information.

This map writing procedure changes the classic "read-only" use of the map to a "read-write" vision, from which it becomes a powerful tool to use in the scope of machine learning, providing a learning support for storing geographic-based information.

The research question can then be formulated as:

How to determine with integrity a map-matching solution, in order to ensure the correct use of the map for data writing purposes?

1.3.2 Thesis Scope

This thesis focuses on the localization system of the intelligent vehicle architecture. The perception is not studied: sensors are considered as black boxes, returning already processed features as inputs. The vehicle actuation is also out of scope as the method aims to be used primarily on map-aided learning task, and thus performs on a manually driven vehicle.

Repetitive travels represent a large part of a vehicle use (e.g., daily commuting to work) and therefore is a good study case. Multiple acquisitions on the same locations would bring redundancy to the data, leading to a more robust processing. Typical environments range from moderately dense urban to peri-urban areas.

The considered map is a lane-level map with a high level of accuracy. It is considered correct and up-to-date. The GNSS receiver is representative to those embedded in the automotive market, i.e. low-cost with limited accuracy. The localization developed in this thesis is centered on the correct use of a map and may not correspond to the needs of navigation or control-oriented systems.

1.4 Contributions

The main contributions of this thesis are:

- A map-matching (MM) algorithm, developed focusing on the handling of matching ambiguities that can arise from the difference of accuracy between the map and the GNSS receiver.
- A multiple matching hypotheses management method, addressing these ambiguities, making an efficient use of the map architecture for hypothesis generation.
- A coherence metric, introduced using the previous map-matching algorithm results to assess the level of trust that may be put in each matching hypothesis. It is then used to express a "Use/Don't Use" characterization corresponding

to the usability of the positioning system given the current situation, notably the presence of ambiguity.

- A direct application of the map-matching coherence checking algorithm to write learning information into the map, as a proof-of-concept. Various information, gathered from different sensors is added to the map acting as a learning canvas.
- Experimental validation of the algorithms, performed on data acquired from a real vehicle.

1.5 Thesis Contents

This thesis is organized as follows:

Chapter 2 describes the positioning systems used in the automotive industry along with existing digital road maps formats. Focus is placed on the map-matching process and the position integrity, very important in the context of intelligent vehicles.

Chapter 3 presents the map-matching algorithm developed during this thesis, based on a particle filter. Details about its implementation are given, along with results using data gathered on open roads.

Chapter 4 introduces coherence metrics used to determine a "Use/Don't Use" classification of the positioning system, based on a multi-hypothesis matching approach.

Chapter 5 presents a case study of the use of such map-matching: a learning map scenario during which the correct localization on the map is required in order to add data gathered from different sensors to the map.

Chapter 6 concludes the thesis by summarizing what has been achieved, a critique on the proposed solution and a reflection on further work.

Publications

International Conferences

F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "*Using High Definition Maps to Estimate GNSS Positioning Uncertainty,*" in 2017 European Navigation Conference (ENC), 2017.

F. Li, P. Bonnifait, J. Ibañez-Guzmán, and C. Zinoune, "*Lane-level map-matching with integrity on high-definition maps,*" in 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 1176–1181.

F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "*Estimating localization uncertainty using multi-hypothesis map-matching on high-definition road maps,*" in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–6.

Journal

F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "*Map-Aided Dead-Reckoning With Lane-Level Maps and Integrity Monitoring,*" IEEE Trans. Intell. Veh., vol. 3, no. 1, pp. 81–91, Mar. 2018.

Patent Applications

F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "*Procédé de map-matching utilisant un filtre particulaire*", applied for in March 2018

F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "*Procédé de décision et diagnostic utilisant un calcul de cohérence*", applied for in March 2018

Distinctions

Best Presentation Award at the 2017 European Navigation Conference, taking place in Lausanne, Switzerland.

Chapter 2

Technologies for Vehicle Localization on High-Definition Maps

Contents

2.1 Introduction **18**
2.2 Global Navigation Satellite Systems **18**
2.2.1 GPS History 19
2.2.2 Structural Components 19
2.2.3 Reference Frames 21
2.2.4 Operational Problems 21
2.2.5 Position Calculation 24
2.3 GNSS Protection Levels **25**
2.3.1 Aeronautic Integrity 25
2.3.2 Road Transport Integrity 27
2.3.3 HPL Availability 29
2.4 Digital Road Maps **29**
2.4.1 Different Scales Available 29
2.4.2 Existing Formalisms 32
2.4.3 Semantic Information 37
2.4.4 Map Accuracy Metrics 38
2.4.5 Map Used in This Research 40
2.5 Map-Matching Approaches **43**
2.5.1 Different Matching Principles 44
2.5.2 Different Map Scales 46
2.6 Experimental Setup and Functional Architecture **47**
2.6.1 Experimental Vehicles 48
2.6.2 Sensors 49
2.6.3 Middleware 51
2.6.4 Functional Architecture 52
2.7 Conclusion **53**

2.1 Introduction

As intelligent vehicles capabilities improve, the requirements for the localization function increase. Two types of localization exist: the global and local localization. The former places the system in a general reference frame, allowing procedures such as high-level path planning. The latter aims to localize the vehicle with respect to its immediate environment. Both are important for autonomous vehicle navigation, the former allows the navigation on a large space and using/sharing information with other systems, whilst the latter is used for immediate tasks.

This thesis focuses on the global localization task as it enables contextual information gathering, crucial for local localization, through the use of digital road maps. GNSS is the main source of absolute positioning in the automotive domain, based on inferring the relationship between the vehicle and a constellation of satellites used as landmarks. Digital road maps are databases containing geographic information about the road network and its context. To be able to use this information, the vehicle's global position must be correlated to a position in the map: this procedure is called Map-Matching (MM).

On one hand, map accuracy is improving following the needs of intelligent vehicles, reaching a level of granularity allowing a clear understanding of the road configuration (e.g., the position of drivable lane, lane markings). On the other hand, GNSS suffers from error sources, diminishing its accuracy. This contrast is particularly strong in urban environment, where the map contains a high density of roads, while GNSS is affected by multipath problems (i.e. signal propagation perturbations caused by buildings), introducing positioning error. This can cause mismatches in the map-matching process, potentially leading to hazardous operations if incorrect map information is provided to the decision making process.

In this chapter, the principles of GNSS are summarized to understand the operational constraints acting upon them. Details about the digital road maps for intelligent vehicles are introduced. The issue of position integrity is formulated as well as the fundamentals and importance of the map-matching process, especially when using high-definition road maps. The chapter concludes with a presentation of the experimental setup used during this thesis.

2.2 Global Navigation Satellite Systems

Early explorers were able to localize themselves observing star constellations. GNSS positioning technology is based on the same principles, artificial satellites replace celestial objects and observations are now automated in GNSS receivers by processing signals transmitted from the satellite constellations. The operating principle resides in determining the position by measuring pseudoranges, metrics related to the Time Of Arrival (TOA)) at receivers of signals broadcast by the satellites [Kaplan and Hegarty, 2006]. To convey the issues related to GNSS positioning, this section centers on the use of the American GNSS known as Global Positioning System (GPS). Appendix A provides details about other GNSS: GLONASS, Galileo and BeiDou-2.

2.2.1 GPS History

The GPS is the American solution for worldwide satellite positioning. The US Department of Defense began its development in 1970, launching the first satellites in 1978. Full Operational Capability (FOC)) has been reached in April 1995. The US government initially reserved its use to military operation exclusively, but as early as 1983, the decision was made to open it for civil application with a voluntary degradation of accuracy named Selective Availability (SA). In 2000, former US President Bill Clinton decided to disable SA allowing a civilian accuracy of around ten meters [Zumberge and Gendt, 2001] following the Korean Air Lines Flight 007 tragedy.

Nevertheless, two types of service are offered: the *Standard Positioning Service (SPS)*, available to civilian uses, and the *Precise Positioning Service (PPS)*, reserved to military and governmental use (using an encrypted signal), providing greater accuracy.

2.2.2 Structural Components

GPS is composed of 3 segments, a structure common to every GNSS [Kaplan and Hegarty, 2006].

2.2.2.1 Space Segment

This segment regroups the satellites, also called Space Vehicles (SV), forming the GNSS constellation. The SV are spread across several orbital plans in order to optimize the worldwide coverage (or of the targeted local area for regional GNSS). Their orbital parameters differ for each GNSS. The GPS satellites orbit at an altitude of approximately 20,200 km, in Medium Earth orbit (MEO). As of March 2018, 31 satellites are operational [National Coordination Office for Space-Based Positioning Navigation and Timing PNT, 2018]. Each satellite makes two revolutions around Earth each sidereal day (11h 58m 2s period), following 6 different orbits. A minimum of 4 satellites are needed to compute a position (see Section 2.2.5). In practice, the constellations are designed to provide over-determination to the positioning problem. For example, the GPS constellation is designed to offer the best coverage of every point on Earth surface with, at least 6 satellites within line of sight at any given time and place. This number is however only theoretical, as it does not take into account possible masking from the environment (see Section 2.2.4).

Among the most critical elements of a satellite is its atomic clock[1], as highly-accurate timing is the basis of GNSS positioning. It is used to generate the master clock running at 10.23 MHz from which are generated the carrier waves used to broadcast positioning signals, see Table 2.1.

2.2.2.2 Control Segment

The control segment ensures the correct operation of the Space segment. Base stations on Earth communicate with the satellites for monitoring purposes. Operational updates are performed regularly to ensure a good performance of the system, such

[1]There are actually multiple redundant clocks embedded in a single satellite to cope with potential system failures

Table 2.1 – Frequencies used as of 2018 for GPS positioning signals, f_0 represents the master clock frequency (10.23 MHz).

Signal Carrier	L1	L2	L5
Frequency (MHz)	$154 \cdot f_0 = 1,575.42$	$120 \cdot f_0 = 1,227.60$	$115 \cdot f_0 = 1,176.45$

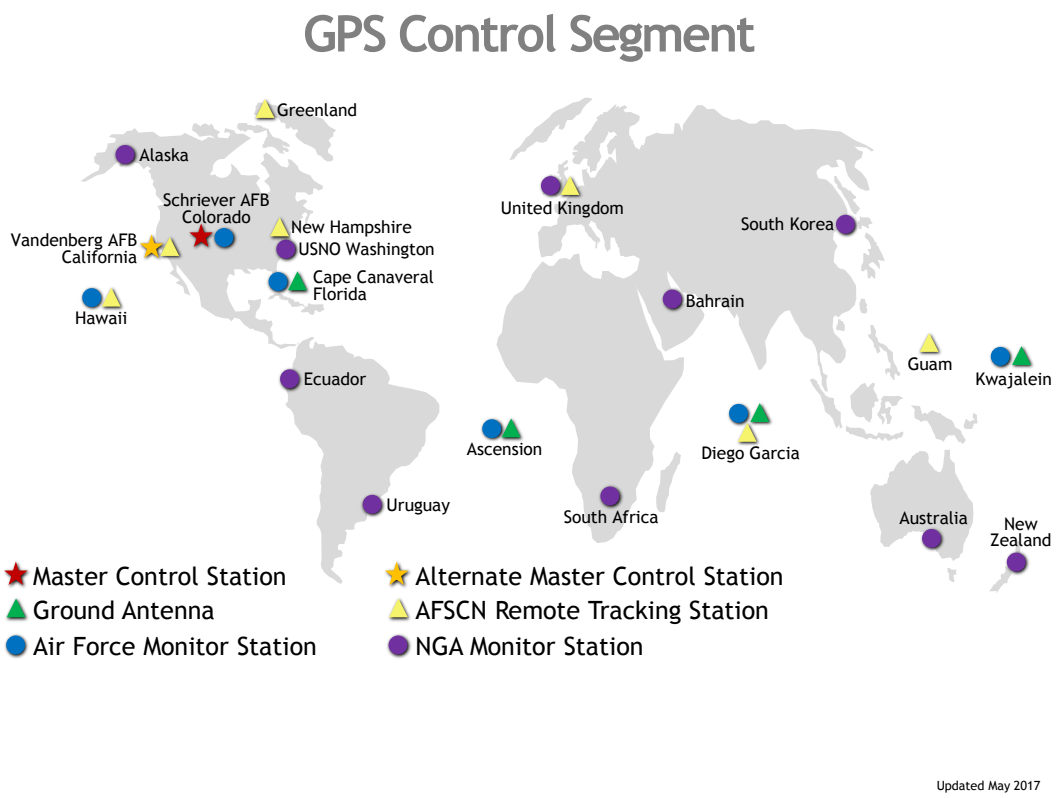


Figure 2.1 – Map showing the different element forming the GPS control segment. (Image from `https://www.gps.gov/systems/gps/control/`).

as clock synchronization or ephemeris corrections. The latter operation is crucial for the system to operate correctly: as presented in Section 2.2.5, GNSS requires to know accurately the position of the satellites. Ephemeris, broadcast by the satellites must then be corrected whenever it is necessary, based on the observations made by the control segment.

GPS control segment composed of 16 dedicated ground stations. The ground stations report to the Master Control Station (MCS), located in Colorado Springs where corrections are computed to be transmitted to the satellites using 4 dedicated communication antennas. Their worldwide distribution allows every satellite to be monitored by two stations simultaneously (see Fig. 2.1). This provides a greater availability in case communication with a specific satellite is needed.

2.2.2.3 User Segment

The GNSS receivers compose the last segment called user segment. They are in charge of determining the position of the system but also the current time, as both are intertwined for GNSS (see Section 2.2.5). These elements are passive in the sense

that there is no outgoing communication with the other segments, their number is thus unlimited. Modern receivers are often compatible with multiple constellations, increasing the number of visible satellites and thus the computation accuracy.

The positioning calculation is done by the user segment, estimating the distances (i.e. pseudoranges) of the receiver to the satellites.

2.2.3 Reference Frames

2.2.3.1 Time Reference Frame

The GPS being a worldwide system, a global time and space reference frame must be defined. GPS Time (GPST) is the time scale used in this system. The GPS epoch, the origin of the scale, is January 6, 1980 at midnight. GPS date is counted in weeks since this date, with a subdivision in seconds within the week. GPST is a continuous time scale, meaning that leap seconds[2] are not taken into account. As a consequence, GPST has a known offset with the Coordinated Universal Time (UTC), constant between leap second injection. Atomic clocks are used to measure time, both in the satellites and the ground stations. The control segment is in charge of synchronizing the system to respect the accuracy specifications of being within $1\,\mu$s of UTC (not removing the UTC offset). In practice, the synchronization has been kept around 25 ns [Parker and Matsakis, 2004].

2.2.3.2 Spatial Reference Frame

The chosen geodetic system is the World Geodetic System 1984 (WGS84) [Kaplan and Hegarty, 2006]. It is an Earth-Centered, Earth-Fixed (ECEF) frame: the origin is located at the center of the Earth and the axes follow its rotation. The axes are defined as shown in Figure 2.2: the zero meridian (through which passes the X axis) is located 102.5 meters east of the Greenwich meridian [International Civil Aviation Organization ICAO, 2002]. The Y and Z axis are then oriented to form a right-handed frame, with the Z axis passing through the geographic north pole.

The coordinates system used in GPS is longitude-latitude-altitude (altitude is different from ellipsoidal height), corresponding to the spherical coordinates of a point, although the altitude is the distance from the surface of an ellipsoid approximating the Earth's surface (rather than the distance to Earth's center of mass). A position on Earth can therefore being expressed by its longitude (angle with respect to the zero meridian, i.e. the XZ plan), latitude (angle with respect to the equatorial/XY plan) and its altitude from a reference geoid. These are commonly called GPS coordinates.

2.2.4 Operational Problems

GNSS are affected by perturbations, even under nominal conditions. As presented previously, positioning is based on satellite ranging using TOA. It makes the assumption that the GNSS signal travels at the speed of light in a straight path from the satellite to the receiver. This section presents phenomena impacting the signal propagation and thus adding errors to the TOA.

[2] added (or subtracted) periodically to synchronize the civil time with the Earth's rotational time

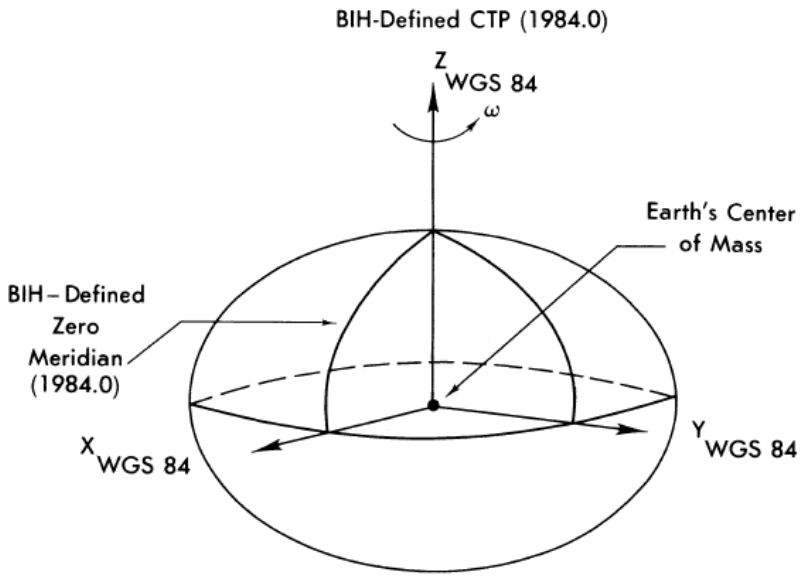


Figure 2.2 – WGS84 reference frame (image from International Civil Aviation Organization ICAO [2002]).

2.2.4.1 Systematic Sources of Error

The following errors are well known and are correctable (for most of them) using appropriate models, even if some residual error may subsist.

By design, a GNSS creates some known errors. For example, electronic delays are present during emission and reception of a signal in the on-board circuitry. The control segment is in charge of monitoring it on the satellite side and compute the corresponding corrections, while the issue is often disregarded on the receiver side as its impact on positioning accuracy is negligible [Kaplan and Hegarty, 2006]. However, it must be taken into account for high-accuracy timing applications.

Being on orbit around Earth, the satellites evolves at high speed and relativistic effects are present. The master clock frequency is adjusted to 10.22999999543 MHz to be observed at sea level as 10.23 MHz. A variable correction remains to be computed by the receiver caused by the satellite orbit's eccentricity. Seeber [2003] estimated the maximum delay to be around 70 ns, corresponding to a range of 21 m.

The position calculations are based on the knowledge of the accurate position of the different satellites. The control segment is in charge of correcting the ephemerides broadcast by the system (or even the satellite orbit if needed). While this correction is done frequently (every 2 hours), it remains possible (although rare) that a satellite deviates significantly from the planned orbit, skewing the measurement.

2.2.4.2 Signal Transmission Perturbations

GNSS are also prone to numerous external sources of error. One of the most prominent is the different transmission media encountered on the path from the satellite to the receiver. The signal crosses multiple layers of atmosphere before reaching the receiver, two of them have a more significant impact.

The ionosphere Ranging from about 60 km to 1,000 km altitude, the atmosphere is thin enough to be ionized by solar radiation [Klobuchar, 1996]. This phenomenon can affect greatly the signal propagation by adding a delay in the signal travel. The induced perturbation depends on the solar activity: the ionosphere's properties change between night and day, geographic location (latitude). Seasonality also impacts locally the ionosphere as well as solar activity. The consequence is an elongation of the pseudorange that can reach 45 m [Jorgensen, 1989]. Error models can be applied to correct roughly the delay. Another method consists on using a bi-frequency receiver, as the ionosphere is a dispersive medium (i.e. the delay is frequency-dependent), these two measurements allow the error to be isolated [Navstar, 2013].

The troposphere This is the lower part of the atmosphere, ranging from the ground level to around 20 km altitude. This layer is not dispersive but the induced refraction is not negligible [Spilker, 1996]. The amount of delay added depends on the distance traveled through the troposphere and thus depends on the satellite elevation. Moreover, weather (i.e. temperature, pressure, humidity) is also a factor. The delay can be estimated by various mathematical models [Hopfield, 1969; Saastamoinen, 1959]: it can range from 2 to 30 m, depending on the previous factors [Hofmann-Wellenhof et al., 2008].

2.2.4.3 Multipaths

Beside these natural sources of error, some are caused by the direct environment, often human-built structures. For example, buildings can cause *multipath* [Braasch, 1996]: the GNSS signal does not reach directly the receiver in a straight path, but instead is reflected by a building. This phenomenon is frequently observed, especially in modern districts where buildings with highly-reflective surfaces (e.g., glass facades) are common. Two cases exist (illustrated in Fig. 2.3):

- **Line-Of-Sight (LOS) Propagation**: the satellite remains directly in view: the receiver therefore gets both the direct signal and the reflected one.
- **Non-Line-Of-Sight (NLOS) Propagation**: in this situation, the satellite is occluded and the direct signal does not reach the receiver, only the reflected one does.

In both situations, measurements can be affected by a pseudorange bias of up to 20 m [Wells et al., 1986]. This amount of error is critical, especially when considering urban scenarios where multipath is a common problem. This source of error is of particular interest since, unlike the atmospheric perturbations, no generic model exists describing multipath delays, as the phenomenon is too specific to a given area. Nevertheless, if the immediate environment is well known, a local model can be made. For instance, using accurate local data such as 3D models of the buildings, multipath can be predicted and thus mitigated during the position calculation [Betaille, 2017; Hsu, 2017]. Such methods often rely on simulating the signal propagation and can use for example ray-tracing calculations [Lau and Cross, 2007] which is computation-intensive. Moreover, an accurate localization must also be known, as a few meters lateral shift can be enough to change the building satellite shadowing [Bressler et al., 2016].

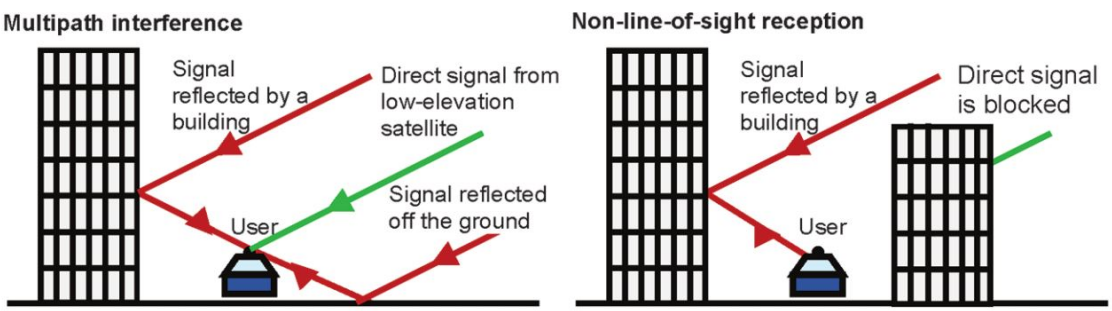


Figure 2.3 – Example of multipath and NLOS conditions. In the first case, the direct signal reaches the receiver while for the latter, only the reflected signal does. (Image from [Groves et al., 2013]).

Multipaths along with a smaller number of visible satellites are the main reasons that GNSS localization can be challenging in dense urban areas. In most cases, a GNSS receiver alone is not sufficient to yield a reliable accurate positioning.

2.2.5 Position Calculation

This section presents an overview of the position calculation using pseudorange measurements. Using GNSS for positioning is resolving a system of equations with 4 unknowns: 3 coordinates and a time component. The time unknown is more precisely the receiver-side clock offset: as the low-accuracy systems are used (quartz-based), the receiver's clock drifts and an unknown offset must be determined.

2.2.5.1 Measurement Equations

A minimum of 4 equations are thus needed to solve the problem. These are obtained by measuring pseudoranges from at least 4 different satellites. The term *pseudorange* denotes the fact that the distance to the satellite is computed based on the TOA and is therefore affected by the clock offset. If the system was not affected by any noise, exactly 4 measurements would be needed to compute an exact positioning. But as developed in Section 2.2.4, pseudorange measurements are affected by errors, noted e_n. A correct method to mitigate them is to consider more than 4 measurements in order to be able to resolve the system using a non-linear least squares method.

Considering that n measurements are obtained (with $n \geq 4$), the system of equations is:

$$\begin{cases} \rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + c \cdot t_u + e_1 \\ \rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + c \cdot t_u + e_2 \\ \rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + c \cdot t_u + e_3 \\ \rho_4 &= \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + c \cdot t_u + e_4 \\ & \vdots \\ \rho_n &= \sqrt{(x_n - x_u)^2 + (y_n - y_u)^2 + (z_n - z_u)^2} + c \cdot t_u + e_n \end{cases} \tag{2.1}$$

where c is the speed of light and (x_i, y_i, z_i) are the coordinates of the ith satellite ($i = \{1, ..., n\}$) (known for the satellite ephemeris). The 4 unknowns are the coordinates of the receiver (x_u, y_u, z_u) and its clock offset t_u.

2.2.5.2 Multi-Constellation Positioning

Every GNSS relies on the computation of pseudoranges (with some technical differences), measurements from different systems are therefore compatible with each other. It is then possible for a receiver to use satellites from multiple constellations, potentially increasing the number of visible satellites (thus the number of available measurements) in order to compute a more accurate position. This procedure is of course limited by the hardware capabilities of the receiver, that may not be compatible with all constellations.

2.3 GNSS Protection Levels

The previous section showed that GNSS positioning is inherently prone to errors at varying level, depending on the environment. In the context of positioning for intelligent vehicles navigation, some characterization of its uncertainty is mandatory. This section presents the notion of positioning integrity, corresponding to such requirement.

2.3.1 Aeronautic Integrity

GNSS integrity stems from the aeronautic domain, where positioning accuracy is critical, especially in some critical flight steps (e.g., airport approach). It has the advantage to perform in a controlled environment, simple for GNSS in terms of perturbations (i.e. open sky). The International Civil Aviation Organization ICAO [2005] defines the positioning integrity as "*a measure of the trust which can be placed in the correctness of the information supplied by the [positioning] system*". Moreover the users must be alerted in a timely manner in case this trust level reaches a too low level, indicating that the positioning system should not be used.

The aeronautic positioning integrity is based on the joint use of 2 metrics: the alert limit and the protection level, described as follows.

2.3.1.1 Alert Limit

Every system has a margin of acceptable positioning error, that can vary, depending on the current navigation task. The alert limit is then defined as this margin corresponding to the largest level of error that will not interfere with the correct system operation. As its name implies, the system must trigger an alert to the user in the event of the level of error exceeding the limit.

Note that it is composed of 2 values, as all metrics presented in this section: the Horizontal Alert Limit (HAL) and Vertical Alert Limit (VAL). For simplicity, the following will only consider the horizontal component, as it is the most relevant dimension in ground transportation.

Linked to this notion of HAL is the Time To Alarm (TTA): this specifies the maximum acceptable duration between the positioning failure and the alert raising. This effectively represents the maximum duration during which the system may be using a faulty position.

The probability of an error larger than the HAL is also specified by the Target Integrity Risk (TIR) of the system. This value is application-dependent, representing its integrity requirements and must be specified by the system.

2.3.1.2 Protection Level

The Horizontal Protection Level (HPL) corresponds to an upper bound to the positioning error, considering the given TIR. In other words, the positioning error is guaranteed to be smaller than the HPL with a probability of $1 - \text{TIR}$. The HPL is related to a specific integrity monitoring method, such as the Receiver Autonomous Integrity Monitoring (RAIM) [Brown, 1996] (see Section 4.2.1) and is independent from the actual measurements. As a consequence, knowing the states of the GNSS constellations through their ephemeris, the HPL can be computed in advance, creating a HPL prediction map, that can be used for path planning, in order to stay in low-HPL areas. This prior computation is only valid in open environments as encountered in the aeronautics (open sky conditions). In the automotive domain, other calculation methods are available. Some of them are presented in Section 2.3.2.

2.3.1.3 Uncertainty Level

The Horizontal Uncertainty Level (HUL) is used to characterize the uncertainty that a GNSS position bears. It is an estimation of the Horizontal Positioning Error (HPE), which is not accessible in real conditions (i.e. no ground truth data).

The HUL can be seen as an extension of the HPL. It is based on additional information from the current configuration and measurements of the positioning system (e.g., geometrical configuration of the satellites, positioning calculation least-squares residuals), the actual estimation method depending on what data are available (e.g., raw GNSS measurements). Note that HUL is not used in the aeronautic integrity norms but is frequently used in positioning performance assessment [William J. Hughes Technical Center, 2017].

2.3.1.4 Different Integrity Cases

An illustration of the HUL, HPL and HAL is presented in Fig. 2.4, each is represented by a circle, centered on the estimated position.

The HUL circle represents the error contour, given the current measurements. By definition, the real position is contained in this circle, considering the integrity risk and a correct uncertainty estimation.

The HPL circle represents the error contour given the information on the GNSS constellations and integrity method. This error contour is computed with purely a priori information.

The HAL represents the level that, if exceeded, raises an alert.

4 situations are then possible:

- Normal situation HUL $<$ HPL $<$ HAL: In this situation the HUL is smaller than the HPL so the actual conditions respect the a priori model of the GNSS positioning integrity. The alert limit is greater than the HPL, meaning that the required level of accuracy can be met a priori. Note that this does not guarantee the respect of integrity during time, but means that a fault would be detected and an alert triggered if necessary.

- Integrity Monitoring Unavailable HUL $<$ HAL $<$ HPL: In this case, the HAL is lower than the HPL. The integrity monitoring does not meet the alert requirement in the sense that some errors e risk to pass unnoticed when

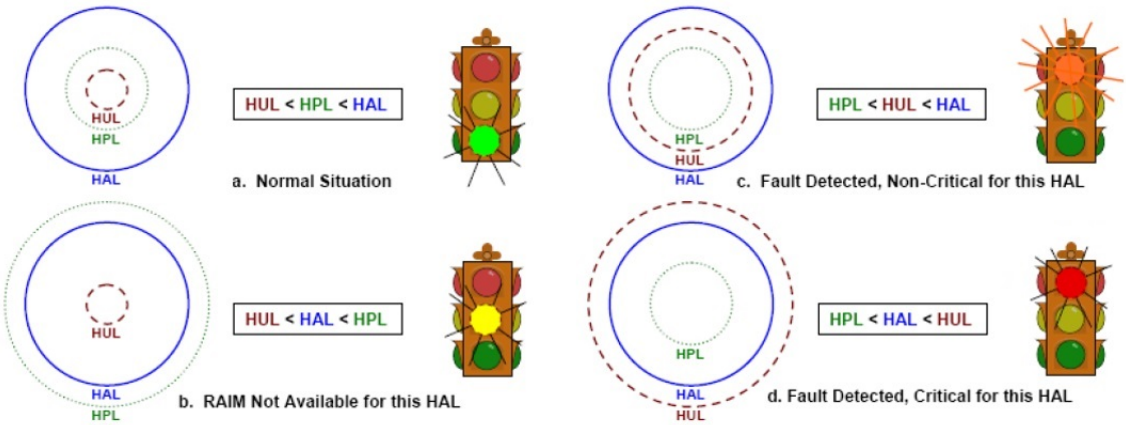


Figure 2.4 – Geometric illustration of the different integrity metric with their signification (Image from [William J. Hughes Technical Center, 2017]).

HAL $< e <$ HPL: the error is not big enough to be detected by the integrity monitoring but is out of acceptable range for the application. The positioning integrity cannot be guaranteed, and its monitoring is thus unavailable.

- Non-critical fault detected HPL $<$ HUL $<$ HAL: As the HPL is exceeded, the a priori model is therefore false, meaning that a fault is detected. This situation is the opposite of the previous one, as a fault is not consequent enough to alert the user. The system requirements classifies it as non-critical so no alert must be raised. The positioning system is marked as providing misleading information.

- Critical fault detected HPL $<$ HAL $<$ HUL: In this case, the HUL exceeds the HAL, an alert is raised as the fault is critical to the system. The positioning system is marked as providing hazardously misleading information.

Another representation of these different situations is the Stanford diagram, shown in Fig. 2.5.

2.3.2 Road Transport Integrity

The previous metrics are well adapted to the aeronautic domain but is hardly transposable to ground transportation, especially in dense urban environment. For example, most integrity monitoring systems make the assumption of a single faulty GNSS source: as presented in Section 2.2.4, multipaths are common in urban areas and may affect multiple signals.

These methods are thus too optimistic to be applicable directly to the automotive industry. Moreover, the scales must be adapted: aeronautics specifications, for an integrity risk of 10^{-7}, yields HAL reaching values in the order of magnitude of several kilometers, and several tens of meters for the smallest values. These of course, cannot be usable in a road environment.

A study of the integrity for road applications, especially for urban environment is therefore needed and is of course already a main topic of the automotive research [Drevelle and Bonnifait, 2011; Le Marchand et al., 2009; Zhu et al., 2018]. The European project Satellite Positioning Performance Assessment for Road Transport

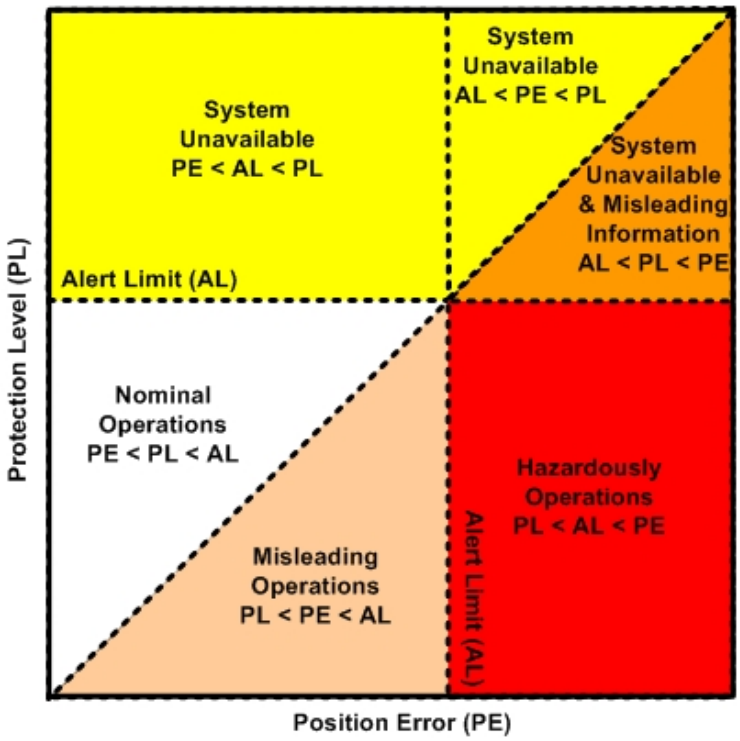


Figure 2.5 – Stanford diagram [Tossaint et al., 2007] illustrating the different integrity states.

(SaPPART, COST action TU1302) has conducted a tentative at proposing some road integrity specifications [SaPPART, 2017]. Due to the high variety of situations and of use cases, a consensus on a unique normative definition is difficult, and is still under discussion.

Different methods have been developed in order to implement such measurements, as presented below.

2.3.2.1 Isotropy-Based Protection Level

The Isotropy-Based Protection Level (IBPL) [Cosmen-Schortmann et al., 2008], developed by the company GMV [GMV Aerospace and Defence S.A., 2009], studies the least square estimation used for positioning calculation (see Section 2.2.5) using its residual vector and the least square Jacobian matrix. The calculation of the protection level is based on the isotropy hypothesis, stating that the error vector can point in any direction with the same probability.

$$\text{IBPL} = k\|r\|\sqrt{\lambda_{\max}} \tag{2.2}$$

with $\lambda_{\max}$, the largest eigenvalue of $(H^T H)^{-1}$, r the vector of residuals and k, a factor depending on the target integrity risk and the number of observation that can therefore be pre-computed [GMV Aerospace and Defence S.A., 2009].

This method is remarkable as it can be applied to any standalone receiver providing access to raw measurements, converging to results obtained using Satellite-Based Augmentation Systems (SBAS) enabled receivers (i.e. receiving corrections computed by ground stations via specific satellite broadcasts).

2.3.2.2 Kalman Integrated Protection Level

The Kalman Integrated Protection Level (KIPL) [GMV Aerospace and Defence S.A., 2013], also developed by GMV, acts on the level of the Kalman Filter (see Section 3.2.3), often implemented by receivers for position filtering.

KIPL is used in when multiple sources of measurements are available, for example combining pseudorange and Doppler measurements. The method then assumes that each source can be bound in error and that these partial bounds are additive to compute an overall error bound: the KIPL. The errors are modeled using Student's t-distributions, more robust to outliers than Gaussian distributions thanks to their heavier tails. Each error bound B_m can then be computed by using the integral of the corresponding t-distribution probability function f and the Target Integrity Risk α. B_m verifies:

$$\alpha = \int_{B_m}^{\infty} f(r)dr \tag{2.3}$$

Details about the calculation can be found in [GMV Aerospace and Defence S.A., 2013]. The final protection level is then the sum of all error bounds:

$$\text{KIPL} = \sum_m B_m \tag{2.4}$$

2.3.3 HPL Availability

The growing interest for road positioning integrity is shown by the availability of protection level metrics, such as IBPL and KIPL, that begin to be computed directly in GNSS receivers. Research is still ongoing in order to develop metrics compatible with ground transportation especially in difficult conditions such as urban environment, along with the normalization of the integrity levels needed for intelligent vehicle applications.

In this thesis, an HPL (for a given integrity risk) is considered to be available as an output of the GNSS receiver, indifferently of the computation method used. The absolute position of the car can thus be considered known, within a radius of error (the HPL), corresponding to a given risk (the TIR).

2.4 Digital Road Maps

The underlying goal of the localization for an intelligent vehicle is to gather contextual information about its environment. A digital map can be considered as a Geographic Information System (GIS), i.e. a georeferenced database. It is therefore a crucial element when evolving in a structured environment such as the road network. This section gives an overview of the maps available for the automotive industry.

2.4.1 Different Scales Available

The scale is one of the main characteristic used to classify the different existing road maps. In classical cartography, the scale is the ratio between a distance represented

Figure 2.6 – An example of a macroscale map: roads in Compiègne, France. Image and data from Google Maps (© 2018 Google).

on the map and its real value on the ground. For example, considering a $1/10,000$ scale, a centimeter on the map would represent 100 m.

In the context of intelligent vehicle, the scale can bear another meaning: it describes the level of detail at which the road network is represented and is thus a direct indicator of the map's accuracy. Du and Barth [2008] define three scales that are used to categorize the digital road maps commonly found in the automotive industry.

2.4.1.1 Macroscale

A *macroscale* map describes a roadway network with an accuracy in the order of magnitude of meters [Toledo-Moreo et al., 2010] (see Fig. 2.6). Aside of the geometric description of the road network, these maps can contain additional information such as speed limitations or the number of lanes present on a given road. The road geometry is often simplified to a minimum level of detail, keeping an acceptable road geometry (often at the roadway level). This allows the representation of large areas with a relatively small memory footprint. The main purpose of these maps are to be informative for a human user, providing for example a high-level guidance. It is thus commonly found in commercial car navigation systems [Betaille and Toledo-Moreo, 2010], as this accuracy is sufficient for applications involving a human driver in the decision process (e.g., turn-by-turn driver guidance).

2.4.1.2 Microscale

Microscale corresponds to very accurate maps, representing the road with the maximum details available. These maps provides very dense information about the environment surrounding the car, causing them to be costly in terms of memory. Data are therefore most of the time coming directly from embedded sensors, forming a dynamic map. This also responds to the need of such maps to be as up-to-date as

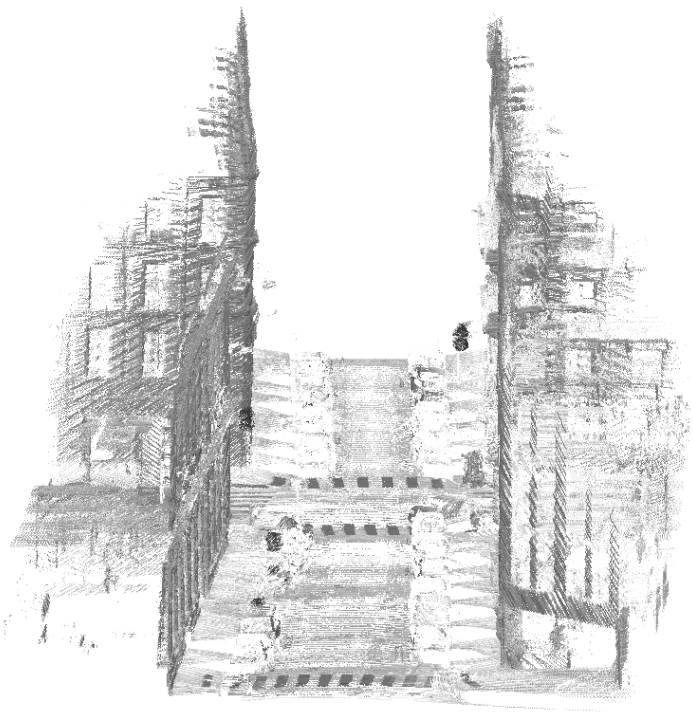


Figure 2.7 – Point cloud from a lidar sensor can be considered as a microscale map. High density of information may not be practical for every use. After the data from *Paris-rue-Madame* database (MINES ParisTech 3D mobile laser scanner dataset from Madame street in Paris).

possible: the purpose of microscale maps being to survey the immediate surroundings of the car (for applications such as obstacle detection), using outdated data is dangerous.

Different sensors (exteroceptive, see Section 1.2.1) can be used for this purpose. For instance, lidars are commonly used to gather dense point clouds of the environment (see Fig. 2.7[3]). The advantage of microscale maps is the completeness of information but this is also the main cause of their drawbacks: the density of information makes handling these maps difficult as computing power is necessary to detect and isolate points of interest. Moreover, data being gathered from sensors, the effective range is limited to a local area.

Occupancy grid [Thrun et al., 2002] is a good compromise to have a fine description of the local environment without being too memory-costly: the environment is represented by a discrete grid in which every cell contains the information about the presence of an obstacle. This forms the notion of free space used in robot navigation problems. This discretization allows a more efficient way to store the data than raw information from the sensors. Data fusion is required when using multiple sensors in order to compute the occupancy of a given cell.

[3]MINES ParisTech created this special set of 3D MLS data for the purpose of detection-segmentation-classification research activities, but does not endorse the way they are used in this project or the conclusions put forward.

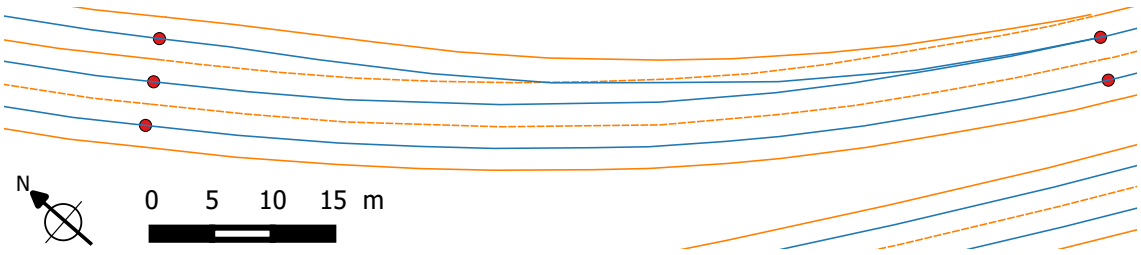


Figure 2.8 – Mesoscale map: each lane is represented separately (in blue). Additional contextual information is included, such as lane markings (in orange).

2.4.1.3 Mesoscale

The map generalization problem [McMaster, Robert and Shea, 1992] states that a map has to provide enough details about the environment without overwhelming the user (whether human or machine) with unnecessary information. Applied to the previous scales, superfluous data have to be removed from a microscale map by extracting exclusively important features that will be integrated into a macroscale map. This is the schematically concept of a *mesoscale* map.

Mesoscale is situated in-between the two previously described scales. It can be described as an enhanced macroscale map in which detailed information on the roadway is added. Structural information such as the nature of the road (e.g., regular driving lane, turn pocket or off-ramp) improves the description and thus the understanding of the network. A finer geometric definition of the road is used, describing for example all drivable lanes separately (see Fig. 2.8), instead of a single roadway. The geometry is also more accurately represented.

The additional information and the better geometrical accuracy allow the use of mesoscale maps for advanced applications (i.e. ADAS). Mesoscale is the most suitable scale for intelligent vehicles, as it carries accurate information without being too dense, maintaining an ease of use [Betaille and Toledo-Moreo, 2010]. It represents a good compromise between macroscale (with a moderate memory footprint) and microscale (with detailed information), providing lightweight maps with a good range without sacrificing information quality. Note that this scale loses the dynamic property of microscale maps, as pre-processing has to be performed.

An example of mesoscale map is described in details in Section 2.4.5.

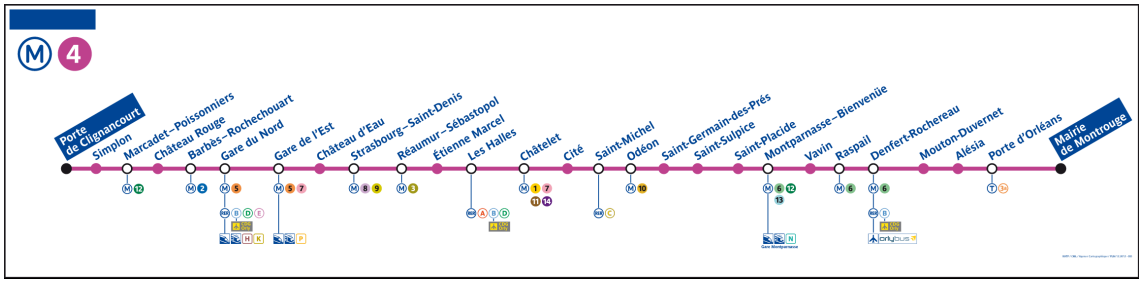
2.4.2 Existing Formalisms

The geometrical description of the road network is the central component of any road map and can be expressed by different means. This section presents the map formalisms commonly used in automotive research.

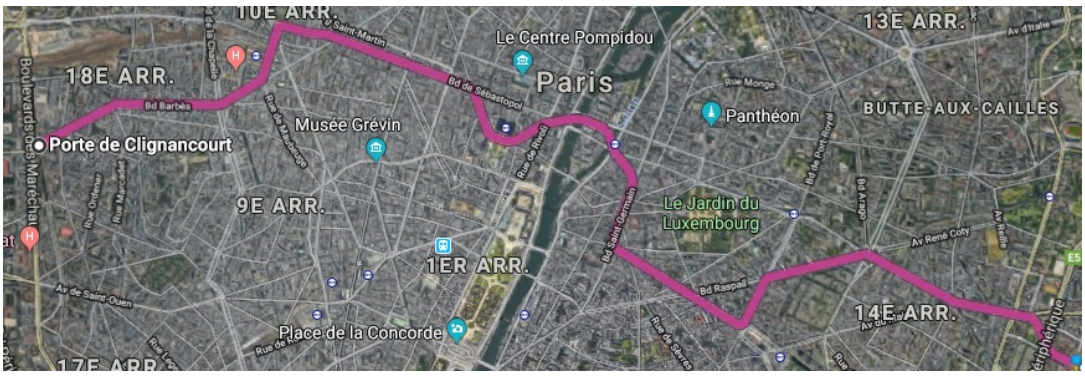
2.4.2.1 Common Notes

The following points apply to every formalism described next:

- Most maps for vehicles only require 2D coordinates, although more advanced uses benefit greatly from the presence of the altitude data [Drevelle and Bonnifait, 2011]. In this research, 2D maps are used, without loss of generality.



(a) Topological map of Paris subway line 4 (source RATP [2018]).



(b) Satellite view with the actual path of the subway.

Figure 2.9 – Comparison between topological maps and geographic maps. Topological maps do not respect geographic reality, prioritizing readability over geographic fidelity.

- Depending on the choice of description level, the smallest map geometry element can represent either a complete roadway or a single lane in the case of lane-level maps.
- Road mapping is based on discretization. This process inevitably introduces approximation errors of the actual geometry of the road [Toledo-Moreo et al., 2010].

These remarks address the geometrical properties of maps which is the most prominent data, providing the shape of the roads. Another feature is the topological information: topography does not focus on geometrical description but on the relationship between different elements. Examples of topological maps are the transportation network maps describing for example the links between subway stations in a convenient way without focusing on geographic accuracy (see Fig. 2.9a).

Advanced road maps often include topological information along with accurate geographic information, mixing the two types of map to provide the most complete information possible (a purely topological map would not be usable as the geographic information is mandatory for intelligent vehicles).

2.4.2.2 Shape Points Models

The most common and straight forward type of map model uses *shape points*: the road network is discretized so that roads are described by a set of points. Polylines (i.e. sets of joined segments) are used to represent the lane centerlines, formed by

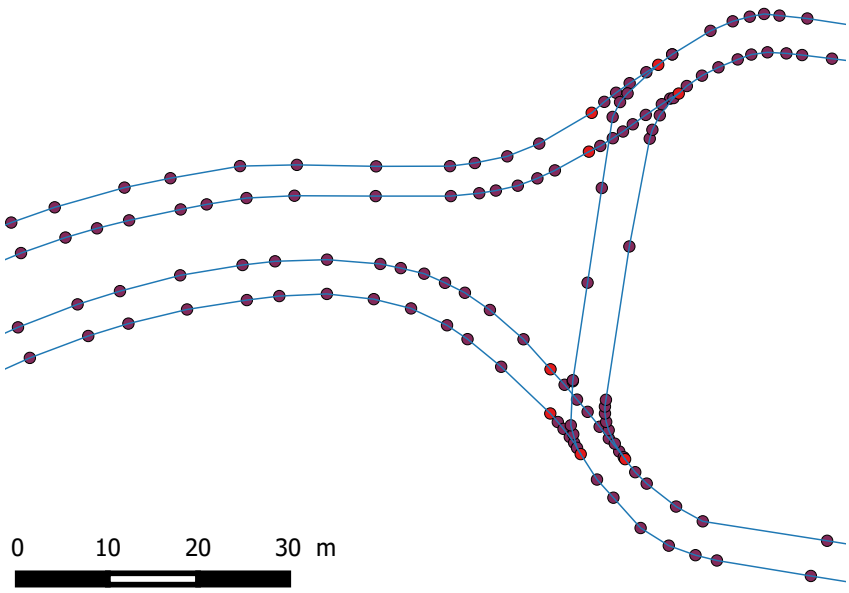


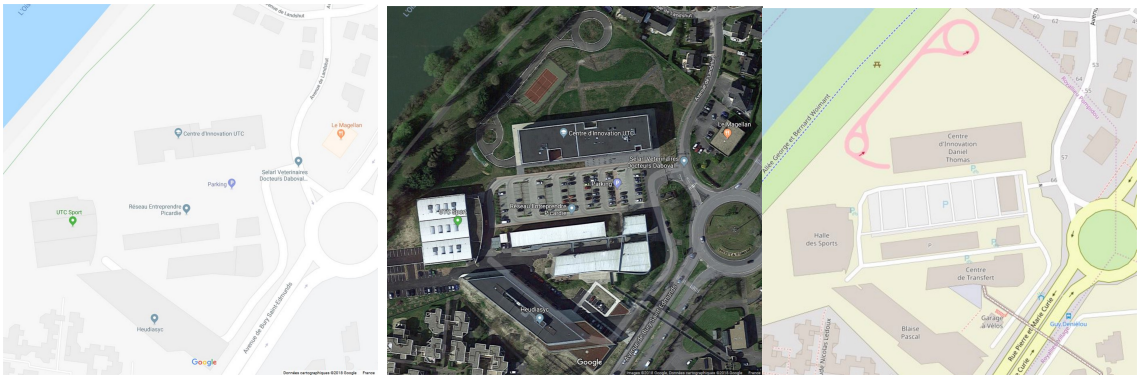
Figure 2.10 – Example of a shape point-based map. The discretization is clearly visible. More shape points are necessary to describe a curve with enough fidelity than a straight line.

linking together multiple shape points. An example is shown in Fig. 2.10 where the shape points are represented by purple dots, linked together to form the drivable lane. The shape points order within the polyline is important to form the correct polyline and may also be used to denote the authorized direction of travel.

The majority of available road maps are shape points maps, as they are the most simple formalism to use. The 2007 DARPA Urban Challenge provided participant with a shape point based map [DARPA, 2007]. OpenStreetMap is a notable project implementing shape point maps. Being a collaborative project with open data [Haklay and Weber, 2008], every user is allowed (even encouraged) to improve the database by adding or correcting elements. This reduces the risk of obsolete information (see Fig. 2.11), as the update period is potentially very short (a user could upload a road modification right after it is made).

This is in contrast with professional-made maps where a survey has to be performed to gather new information. Nevertheless, this functioning mode also has drawbacks: as the data is nearly freely modifiable (moderation exists), some user can, intentionally or not, introduce errors into the map. Professional-made maps do not encounter this problem as quality control is done before release, at the cost of longer update delays. Moreover, some remote areas may not be surveyed by mapmakers, collaborative work may be the only available source of accurate data for mapping [Goodchild, 2007].

This discretization into shape points creates errors caused by the estimation of a smooth curve (the road) by a broken line (the set of shape points). It also causes the need for irregular steps between shape points: to keep the same level of discretization error, more points are needed to fit accurately a bend than a straight line [Kamijo et al., 1989]. This phenomenon is clearly visible on Fig. 2.10 where a higher concentration of points is found on the bends.



(a) Google Maps (© Google). (b) Google Maps satellite view. (c) OpenStreetMap (© OSM contributors).

Figure 2.11 – Maps can contain outdated information: (a) Google Maps lacks a road that has been added by a user in (c) OpenStreetMap (in red). Both snapshots have been taken on the same date, along with (b) the Google Maps satellite view of the area.

Figure 2.12 – Direction changes on highways are smoothed using clothoids. Example of the French N12 road, highlighted in blue. (© OpenStreetMap contributors).

Polyline splitting most of the time corresponds to actual roads, often delimited at intersections (see Section 2.4.3 for other possible reasons). Starting a new polyline at each intersection allows to logically link all relevant roads together without restriction on their number (i.e. 3-way, 4way intersections, etc.). Junctions between the different polylines are denoted by red dots in Fig. 2.10. These elements represent the connectedness of the different links and are thus important to assess the possible paths offered by a road, notably at intersections. Reaching such a point notify to the system that a change is happening in the road network.

2.4.2.3 Clothoidal Models

Road design follows standards setting strict geometrical rules [Sétra, 2006], notably for highways where the curvature variation must be progressive, in order to provide safe vehicle handling conditions during changes of direction. To meet this requirement, road bends are drawn using a succession of clothoidal models (see Fig. 2.12).

Clothoids are geometric objects that have their curvature evolves linearly with the curvilinear abscissa. This property makes them fit well to do transitions between two sections with different curvatures [Meek and Walton, 1992]. Clothoids are expressed mathematically using Fresnel integrals by the parametric equation:

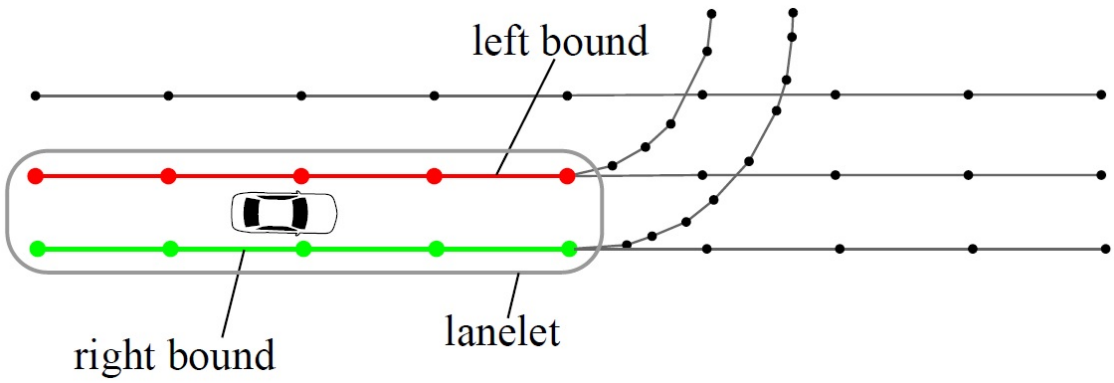


Figure 2.13 – Lanelets uses lane borders to represent the drivable lanes (image from [Ziegler et al., 2014]).

$$\begin{cases} x(L) & = \int_0^L \cos s^2 ds \\ y(L) & = \int_0^L \sin s^2 ds \end{cases} \tag{2.5}$$

with L representing the curvilinear abscissa.

The clothoidal model fits the road design more accurately than the shape point model [Gackstatter et al., 2010], especially in curved sections. Betaille and Toledo-Moreo [2010] use clothoids to define their Enhanced Map (Emap) model in order to benefit from this higher fidelity. Moreover, the number of points needed to represent a road can also be drastically reduced, as a curve can be represented by a single starting point along with the clothoid parameters, reducing the memory footprint of an accurate road map. The clothoid is often simplified to a polynomial expression (see (2.6)). A third degree estimation is accurate enough for roads that have less than 15° change in heading [Dickmanns and Mysliwetz, 1992].

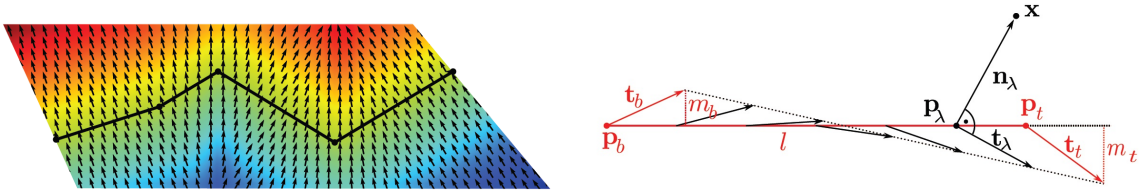
$$y(l) = y_0 + t_0 \cdot l + \frac{1}{2} c_0 \cdot l^2 + \frac{1}{6} c_1 \cdot l^3 \tag{2.6}$$

where y is the lateral offset (y_0 is the initial value), l is the curvilinear abscissa (in m), t_0 is the initial heading (in rad), c_0 is the curvature (in m^{-1}) and c_1 is the curvature derivative with respect to l (in m^{-2}).

The main drawback of the clothoidal model is the need to compute the road geometry, as the map only contains the clothoid coefficients and not the actual coordinates of the points. This problem may be also become a strength for certain applications. For example, the curvature is directly available within a clothoid model: the choice of a model must be done considering the final application.

2.4.2.4 Lanelets

The notion of lanelet has been introduced by Bender et al. [2014]: this type of map is based on a polyline representation of the road. The originality of this map type is that the road lanes are described by their borders and not their centerline. The drivable lane is thus defined as a surface delimited by its left and right bounds (see Fig. 2.13). This gives a 2D view of the road, compared to the single dimension of a polyline.



(a) (b)

Figure 2.14 – Lanelets provide a normal gradient to calculate *pseudo distances* to map elements to remove any discontinuity (images from [Ziegler et al., 2014]).

Topological information about adjacent lanes is also present to provide an overview of the surrounding environment. Previous and following lanelets are also linked together. Additional traffic management elements are included such as traffic lights or stop lines, providing richer information to the system.

Distance calculation is a frequent operation for map data handling. The trivial measurement is to calculate the orthogonal distance between the considered point and the nearest map element (e.g., a segment). But by nature, polylines can cause problems where the junction between two segments causes a discontinuity in the polyline normal vector. Lanelets address this problem by providing a gradient describing the normal vector evolution (see Fig. 2.14a), so that it progresses without discontinuity along the polyline.

Fig. 2.14b illustrates the principle: a tangent vector is affected to every shape point forming the polyline (t_b for point P_b and t_t for P_t, considering a single segment of the polyline). Then, to obtain the tangent vector (and thus deducing the normal) corresponding to any point P_λ on the $[P_b P_t]$ segment, a linear combination of t_b and t_t is computed using. This creates a continuous and smooth evolution of the polyline tangent removing any discontinuity caused by a segment transition.

This formalism is suitable for lane-level positioning [Rabe et al., 2017], crucial for advanced intelligent vehicles applications. It has been notably used in 2013 during the Bertha Benz memorial route autonomous journey [Ziegler et al., 2014].

2.4.3 Semantic Information

Beside the geometric description of the road network, the strength of a digital map resides in the additional information available. An intelligent vehicle needs more than the geometry of the road it is driving on but also contextual information that is a key element to understand the road environment. These contextual properties can be added to the map to complete the geometrical description. Polylines can conveniently bear these properties as they often apply to stretch of road (e.g., speed limitation). If needed, in the case of very punctual information, the shape points can be used as information storing unit. It can even be the only choice, in the case, for example, of the clothoidal model.

In the case of polyline-based model, to obtain a finer description and facilitate data handling, the road network is split into multiple separate polylines. The splitting criteria is the properties uniformity, each polyline bearing the same properties. This way, the contextual properties are only stored once per polyline, instead of once per shape point, thus being more efficient. Note that a single road that seems

uniform (e.g., a stretch of highway) can then be split into multiple polylines if a property, for example, the speed limitation changes along it

This information storage method can of course be completed by dedicated map elements describing exclusively a given property. For example, a shape point can be created to denote the location of a traffic sign.

2.4.4 Map Accuracy Metrics

When considering a map containing geographic information about the road network (i.e. not purely topological), a major map characteristic is its accuracy level. Two types of accuracy are considered separately.

2.4.4.1 Absolute Accuracy Assessment

The first considers the map in its entirety: the absolute accuracy characterizes the map in the global reference frame. Each point (either present directly in the map, or calculated from the different models presented previously) is analyzed individually and the position error is evaluated with respect to reference data that serve as ground truth (see Fig. 2.15). Errors statistics can then be determined to characterize the map (e.g., mean square error and standard deviation). Commercially available road maps can bear absolute accuracy errors up to 10 m [Toledo-Moreo et al., 2010].

A problem raised to assess absolute accuracy is the need of ground truth data. High-accuracy Inertial Navigation Systems (INS) can be used but are expensive. Advanced positioning systems can also be used to minimize the error. For example, the GNSS signal post-processing can be used to remove some perturbations, but this is dependent on the presence of reference base stations in the vicinity of the survey. In France, the *Institut national de l'information géographique et forestière* (French for "National Institute of Geographic and Forest Information", abbreviated IGN) is in charge of maintaining a network of 494 correction stations [Institut national de l'information géographique et forestière IGN, 2018]. Although the bases are spread across French territory, some areas may not be covered by any or outage can occur.

Another type of base station can also be used to complete the existing network or to simply replace it when in a zone without coverage: fixed bases can be deployed (temporarily or permanently) in order to improve the GNSS position calculation. This is possible by determining the position of the station accurately, that can then broadcast correction information to GNSS receivers in real-time: this is the principle of Real-Time Kinematic (RTK). A network connection is needed between the base station and the receiver that may incur fees. Otherwise, a simple recording of the base station data can allow a post-processing, as real-time correction is not needed for validation purposes.

Traditional topographic techniques can also be used to verify the map absolute accuracy. Reference geodesic markers (also managed by IGN in France) are landmarks whose the positions have been determined with great accuracy by a trusted agency. They can thus be used as absolute reference with specialized topology tools (e.g., theodolites). This requires an expertise and thus can be expensive. Reference markers share the same problem of coverage with GNSS stations. Additionally they are sometimes destroyed, for example during road work.

For these reasons, absolute accuracy assessment task is often let to metrology experts (e.g., surveyors).

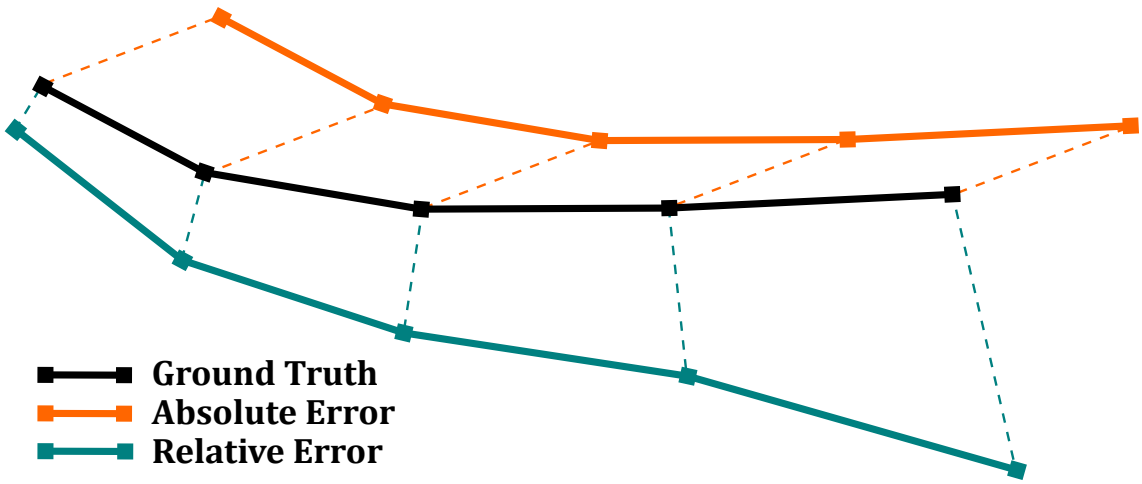


Figure 2.15 – Two types of map accuracy errors are compared to the ground truth: the absolute error (in orange) is constant, while the relative error (in blue) varies.

2.4.4.2 Relative Accuracy Assessment

The second accuracy is the relative one. It studies the relative position of map elements to each other. This characterizes the fidelity of the relationship between the points (see Fig. 2.15). For example, the shape of a road is taken into account, verifying that no divergence appears when going along it. This accuracy focuses more on deviations that cannot be expressed as an homothetic transformation which can be considered as absolute accuracy errors, being correctable by a rotation, translation and/or scale factor (scale errors are uncommon). In other words, a relative accuracy issue corresponds to an error that is not constant throughout the map. It therefore has to be assessed on a relatively extended area.

Problems during field surveying can cause this kind of errors (e.g., incorrect sensor calibration inducing biases). Another source of error appears when using maps of a large area: as the projection model used to "flatten" the map data is only valid locally, a very extended area mapping will present problems of relative accuracy if a single model is used.

Relative accuracy errors are harder to manage than absolute ones, as correcting them implies more complex actions than removing a constant bias. Note that good absolute accuracy implies good relative one. The reciprocal is not true as a map may present a poor absolute accuracy (e.g., a 1 m bias in a given direction) and have a perfect relative accuracy.

Using a map with absolute accuracy is naturally the most desirable situation. But as presented previously, verifying such accuracy can be difficult and expensive. Relative accuracy is then a good fall back if no better accuracy is available, as it constitutes the most important characteristics (absolute errors have still to be kept a least reasonable). It can be made suitable for use thanks to a map-matching procedure, changing the positioning problem from an absolute to a relative one with respect to a map (see Section 2.5).

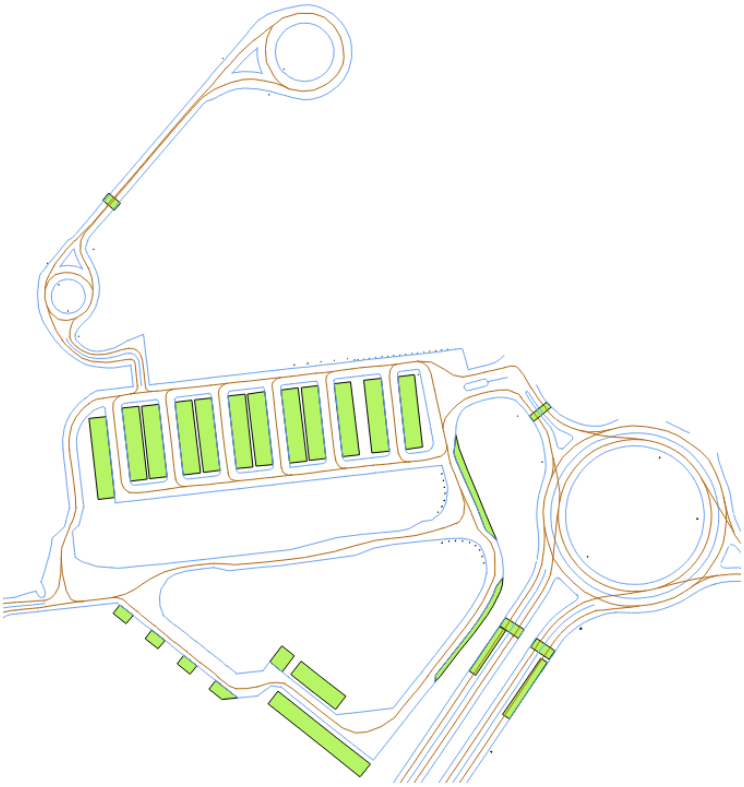


Figure 2.16 – Detail of the lane-level map of UTC *Centre de Transfert*, showing the *Seville* circuit in the top part.

2.4.5 Map Used in This Research

During this research, a proprietary map format has been used. As intelligent vehicles need geographically correct maps, topology-only maps do not meet their high requirements and a geometrical model is therefore used. A shape point mesoscale lane-level model has been chosen, originally designed for Renault's Urban Mobility Advanced Platform (UMAP) project [Renault, 2013]. The algorithms developed in the following chapters are based on this type of map but are not limited to it: they are format-independent and can be used on any map as long as it contains certain mandatory information, described in this section.

A shape point representation has been chosen for its ease of use, providing a direct access to the road geometry without having to compute road point coordinates. But any other representation could be used, although pre-processing may be needed.

A high-accuracy mapping has been realized by a professional topography company, with an announced absolute accuracy of 5 cm. It includes 5 km of public roads and the University of Technology of Compiègne *Centre de Transfert* site (notably the *Seville* test track, see Fig. 2.16). In this sense, the map used in this research is characterized as an High-Definition Map (HD-Map).

This mesoscale map contains a lane-level description of the road network with lane markings position, along with other semantic information (e.g., traffic sign, pedestrian crossing). The map information is stored in a SpatiaLite[4] database: an SQLite extension for spatial information, adding specific advanced queries useful in a spatial context. SQLite provides a lightweight database with a local architecture

[4]More information can be found at: http://www.gaia-gis.it/gaia-sins/

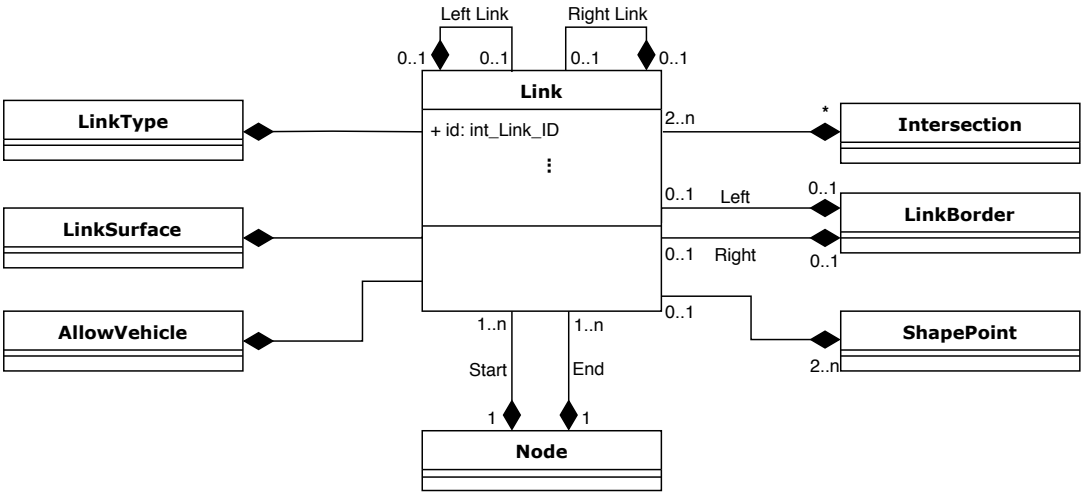


Figure 2.17 – Simplified UML diagram of the Link table, showing the essential elements.

(opposed to a client-server architecture). The Structured Query Language (SQL) engine is directly embedded in the application and the data structure is stored locally in a single SQLite file, along with the data. The database is therefore easy to use, allowing for example, a simple file copy to transfer the whole database to another system.

The road network data are organized in different tables to be able to profit from the relational nature of the database. While many additional tables storing contextual information are present, the following sections describe only the most important elements that are relevant to the algorithms developed in this thesis.

2.4.5.1 Links

Fig. 2.17 shows a simplified Unified Modeling Language (UML) diagram of the `Link` table. This table is the central element of the database and is related to all the other elements that will be described in this section. *Links* are defined as a polyline describing the centerline of the drivable lanes. They contain the geometric description of the roads on which the vehicle can drive, stored as polylines. Each link is given a unique identifier called `link_ID`.

The geometrical definition of links (drawn in brown in Fig. 2.16) is done by using polylines defined with SpatiaLite specific spatial type `linestring`, in which the polyline points are stored consecutively in a single variable. For ease of use (e.g., when isolated points are needed), the points are also redundantly stored as separated shape points in the `ShapePoint` table (see Section 2.4.5.3). The polyline is considered oriented, giving the drivable direction by following the order of the shape points in the `linestring`. With this consideration, two links are needed to define a two-way road even if they represent the same physical lane.

The `Link` table also contains relationship information:

- Connectedness information between lanes is described using references to *Nodes* (see Section 2.4.5.2).
- Surrounding lanes are specified as left and right links, if any exists, representing the adjacent lanes.

- The lane delimiters are given as right and left *link borders* elements referencing the corresponding table (see Section 2.4.5.4).
- Additional lane attributes such as the length, width, maximum speed allowed, are also stored in the `Link` table.

2.4.5.2 Nodes

As described in Section 2.4.2.2, a map is split into links at specific location, for instance, when a property changes or at an intersection. The junction between links is done using elements called *Nodes.* Every link is delimited by two nodes: a starting and ending one, corresponding to the first and last shape point of a link (a node is geometrically defined as a single point). They are denoted `Node_ID_Link_Start` and `Node_ID_Link_End` (`start_node` and `end_node` for short).

Nodes centralize the logical connections of the map: by definition, nodes can mark intersections, lane mergings or bifurcations. Arriving at the end of a link with an `end_node` N_E, the possibilities of successive links can be gathered by finding all the links with N_E as a `start_node`. Nodes are fundamental map elements, acting as connecting agents. For intelligent vehicles, nodes are of particular interest, as intersections are one of the most complex situations to handle [Lefevre et al., 2012].

Note that a link change caused by a change of road property (e.g., lane width) is also marked with a node. In this case, a unique link will begin with N_E as `start_node` and the situation is then more straightforward. Nodes are quite flexible concerning the cardinality of the links it manages. For example, multiple links can end at a given node while only a single one starts: that describes a lane merging. The opposite is also possible, being a lane bifurcation. The only constraint is that every link must be delimited by `start` and `end_nodes`.

2.4.5.3 Shape Points

This table contains the geometric description of every point of the road network contained in the map. The element described as the interval between two shape points is called a *segment.* As discussed in Section 2.4.2.2, the distance between two shape points depends on the curvature of the road. It is however limited to a maximum of 5 m to avoid creating too long segments (e.g., in long straight lines).

The coordinate system is based on a local reference frame with an origin chosen in the area covered by the map. For the Compiègne map, it uses a reference base station located on the university campus. The X and Y axes are defined respectively to the east and north direction (and upward for a 3D map, forming an East-North-Up (ENU) reference frame). The coordinates are expressed in meters.

2.4.5.4 Link Borders

The `LinkBorders` table contains additional geometries, providing a surface information on the drivable area as the lanelets do. These link borders represent the lane delimiters (shown in blue in Fig. 2.16). A *LinkBorder* is characterized by a unique ID (`LinkBorderID`). Its geometry is similar to the links and uses a polyline representation (i.e. `linestring`).

The link border stores the type of lane delimiter, as a link border can represent not only a lane marking but also any other kind of lane delimiter (e.g., pavement). A

LinkBorder is notably present even if no physical delimitation exists, with a "none" value as type. Each link is then affected two link borders, adjacent links always share one. Each shape point includes the distances to the left and right borders.

An a priori view of the roadway can thus be reconstructed, with the centerline surrounded by the lane borders of the specifying types. This can be used to compare experimental data, from the perception [Kim, 2008], with the expected markings stored in the map.

2.4.5.5 Points Of Interest (POI)

Other remarkable components of the road network are stored as `POI`s, corresponding to structures potentially of interest for a vehicle (shown in green in Fig. 2.16). They are commonly defined geometrically as areas (see Fig. 2.16). For example, pedestrian crossings, parking spots are represented by the surface they occupy on the ground. Traffic signs and posts are represented by punctual elements (small surface on the occupied ground space).

POIs often delimits areas that must be handled in a specific manner. For example, a pedestrian crossing indicates a zone on which pedestrians are likely to be present: the intelligent vehicle may then take extra caution near them. In the case of traffic signs, a traffic regulation modification may occur (e.g., speed limit change).

2.4.5.6 Minimal Subset Used in This Research

More tables exist in the database but are not described in this section. In the scope of this research, the only mandatory tables are:

- `Links`,
- `Nodes`,
- `LinkBorders`.

Any map format containing these data is thus compatible with the developed algorithms.

2.5 Map-Matching Approaches

This section introduces briefly the map-matching topic. It will be developed in more depth during Chapter 3.

To process the contextual information contained by a road map, the car's location must be determined with respect to it. In other words, the absolute positioning provided, for example, by GNSS receivers must be translated into relative positioning in the map. This is the process of map-matching.

In the ideal scenario, the localization system provides a position at the same level of accuracy as the map's. In this case, considering a correct map, the map-matching procedure is straight forward as the absolute position matches directly a map location. But in most cases, the actual situation is more difficult: the localization system is affected by errors and uncertainty, as well as the map. These elements often operate on different levels of accuracy; an HD-Map can reach centimeter-accuracy (see

Section 2.4.4) while the GNSS accuracy can bear several meters of errors, a value that moreover varies greatly depending on the environment (see Section 2.2.4).

Map-matching is thus in charge of mitigating this accuracy discrepancy that creates ambiguities between the localization system and the map [Ochieng et al., 2003]. This section presents different existing matching methods.

2.5.1 Different Matching Principles

Most map-matching algorithms use directly the computed GNSS position. This section focuses on this type of matching. Note that some methods use pseudorange measurements for map-matching, removing the need of a global position computation [Fouque and Bonnifait, 2012], particularly useful in harsh satellite reception conditions.

Various principles have been applied for matching purposes.

2.5.1.1 Purely Geometrical Matching

Basic matching techniques only take into account the geometric distance between the GNSS fix and the map elements, matching on the nearest one [White et al., 2000]. Different distance calculation methods are available: a *point-to-point* matching can be done on punctual map elements (e.g., shape points), directly using their Euclidean distance; it can be extended to a *point-to-curve* matching on element such as polylines, using an orthogonal distance. The *pose-to-curve* matching [Taylor et al., 2001] adds a criteria of heading difference (between the vehicle and the considered map segment) to the distance.

The previous methods act as *snapshot* procedure, only considering isolated positions. A temporal dimension can be introduced by using a *curve-to-curve* matching: the map-matching instead considers a trajectory [Joshi, 2001; White et al., 2000].

These methods are straight forward to apply but are highly susceptible to positioning and mapping errors, especially when considering HD-Maps in areas with a dense road network. For example, a lateral bias (with respect to the road direction) can cause the nearest map element to not correspond to the reality (a neighboring road) and thus rending the map-matching result incorrect.

2.5.1.2 Fuzzy Logic

Positioning alone may not be enough in dense areas especially when dealing with uncertainty. In an approach similar to data fusion, other sources of information can be used in a map-matching process. One reference method, developed by Quddus et al. [2006a], uses fuzzy logic [Zadeh, 1989] to represent these different data, describing them with the fuzzy set vagueness. For example the vehicle's speed is described by three classes: zero, low and high, each one having a degree of membership function, as shown in Fig. 2.18. The considered data for the map-matching include the distance of the estimate to the matching hypothesis (i.e. perpendicular distance), and the heading error, among others.

A Fuzzy Inference System (FIS) then defines a set of rules to associate these different fuzzy sets, in order to determine the likelihood of matching of a given link. Multiple candidates are tested, and the final matching is set to correspond to the most likely (see [Quddus et al., 2006a] for details about the FIS).

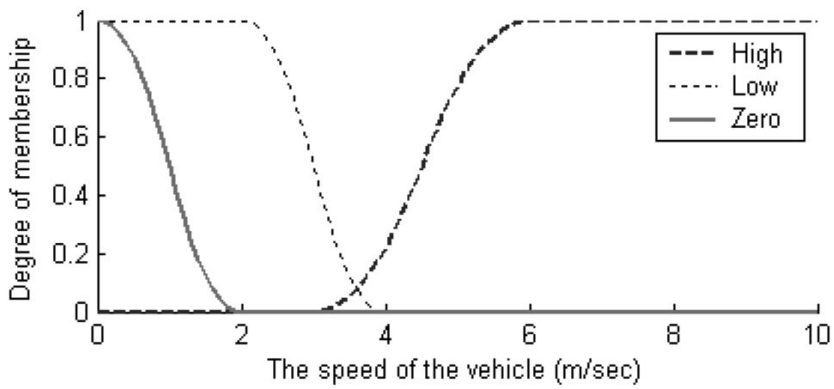


Figure 2.18 – Fuzzy rule describing the speed of the vehicle by three classes: zero, low and high. The definition of the class zero illustrates well the vagueness of the fuzzy logic (image from Quddus et al. [2006a]).

An interesting point raised by the authors is the separation of 3 different situations: the initial matching, the subsequent and the junctions. The initialization of the matching is similar to the kidnapped robot problem [Engelson and McDermott, 1992] that is a real challenging situation in robot localization problem. The management of junctions is also a difficulty for road vehicles, as they raise localization ambiguity, along with the level of risk [Lefevre et al., 2012].

2.5.1.3 Bayesian Information Fusion

Another approach uses Bayesian filtering in order to fuse the additional data. This kind of map-matching often consists of an integration of the map information into the chosen filter. For example, Kim et al. [2000] developed an Extended Kalman Filter (EKF)-based solution fusing GNSS positioning, Dead-Reckoning (DR) and map-matching result. The map-matching is performed by a point-to-curve approach and is used to estimate the lateral bias of the GNSS positioning, the perpendicular distance is the element integrated into the EKF. In this case, the map-matching result (i.e. the identification of the map link on which the car is) is secondary, as it is mainly used as bias estimator.

Extending the previous idea, the map as a whole can act as a constraint to represent the drivable space. Gustafsson et al. [2012] use their map data with this purpose: their map-aided positioning method consists of using a road map as a likelihood field (see Fig. 2.19). The field's value is maximum on the points belonging to the road map and quickly drop when getting away from any map element. This makes this assumption that the car is driving on-road, the possibility to drive off-road (for example when exiting a parking lot) is however taken into account during the final likelihood calculation.

This map-issued likelihood, combined with dead-reckoning information, can then be used to refine a position hypothesis, for example during the update phase of a Bayesian filter [Peker et al., 2011]. It is particularly efficient with trajectory including turns, offering discriminant information for the positioning.

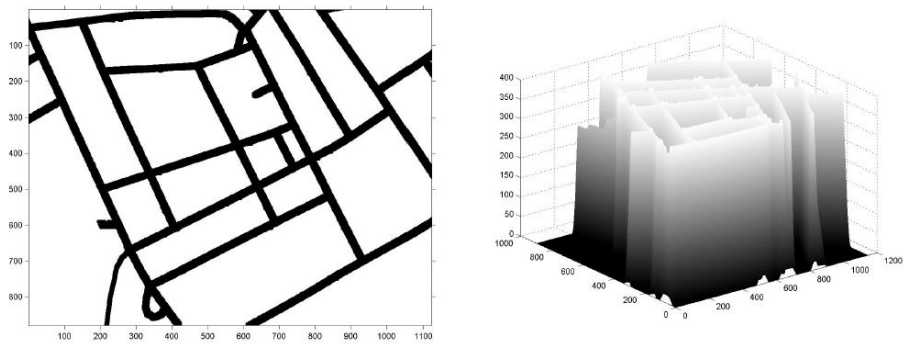


Figure 2.19 – Positioning likelihood field (right) generated from a road map (left) (image from [Gustafsson et al., 2012]).

2.5.1.4 Map as a Strong Constraint

To eliminate the need of determining a likelihood field, some map-matching methods choose to constrain hardly the localization state space to positions contained in the map. For example, the tightly-coupled GNSS position calculation developed by Fouque and Bonnifait [2012] is essentially based on constraining the possible car positions to the roads described by the map's polylines.

Methods based on Particle Filtering (PF, see Section 3.2.4 for a detailed description) are particularly affected by such constraints [Bonnifait et al., 2009; Szottka, 2013]: PF is based on the exploration of the state space in order to converge to the correct solution. By constraining all particles on the polylines contained in the map, the state space is greatly reduced, and the particle exploration can therefore be more efficient. This naturally relies on the correctness of the map, and the respect of the on-road driving assumption.

2.5.2 Different Map Scales

2.5.2.1 Macroscale vs Mesoscale

The majority of reference map-matching algorithms are applied to macroscale maps, with a road-level description. This is due to the prevalence of this scale at their development time. High-accuracy mesoscale maps are becoming more widely available, responding to the growing need of the intelligent vehicle. Map-matching methods adapted to this level of description are needed.

The main challenge stays the same as with macroscale maps: the lack of accuracy of the positioning system. It is only amplified by the map's high-accuracy, creating a large discrepancy. Additionally, a lane-level description of the road network creates new sources of problems: separating each roadway into multiple parallel lanes adds a lot of ambiguities, as these are very difficult to distinguish geometrically. In comparison, Quddus et al. [2007] considered highly dense areas where the average distance between roads is less than 100 m, two lanes side-by-side are separated at most by 4 m.

All these problems render the map-matching algorithms developed for macroscale map inefficient for lane-level maps. New considerations must be taken into account for mesoscale maps. For instance, using the map as a hard constraint is less interesting, as a lateral connectedness exists between adjacent lanes. A less constrained 2D

evolution is more adapted to these maps, letting the filter considering the entire 2D plan [Li et al., 2017a; Rabe et al., 2017], allowing maneuvers such as lane changing to be naturally taken into account.

2.5.2.2 Multi-Hypothesis Management

As the ambiguities caused by the lane-level description cannot simply be eliminated, the map-matching procedure has to manage them. Inspired by the positioning integrity (see Section 2.3), map-matching integrity can be defined as the level of trust that can be placed in the matching result returned by an algorithm [Quddus et al., 2006b].

As such, a sort of matching protection level can be defined: instead of being geometrical, as matching results are discrete, the protection level would be represented by a set of matching hypotheses that is guaranteed to contain the correct matching, with respect to a certain level of risk [Li et al., 2018a]. A solution to ensure this map-matching integrity in ambiguous situations is to manage multiple hypotheses that cover all ambiguities (e.g., on a 2-lane road, having one hypothesis per lane).

Multiple methods have been used in order to implement Multi-Hypothesis Map-Matching (MHMM). For example, Bayesian filtering can be applied to track the hypotheses by implementing a Kalman Filter (KF) for each one or a Gaussian Mixture Model (GMM) can be applied in order to regroup them under a single filter [Jabbour et al., 2008].

Another popular method is the PF: the particles are by nature multi-hypothesis and can cover multiple hypotheses [Bonnifait et al., 2009; Li et al., 2017c; Rabe et al., 2017; Selloum et al., 2009]. The main advantage is the flexibility of the filter, not limited to Gaussian distribution that can adapt to various conditions of the road network. It is of particular interest during intersection, where particle propagation can help for intersection management [Szottka, 2013].

For integrity purposes, being able to track multiple hypotheses is important but not sufficient. A method for hypotheses selection is crucial, i.e. eliminating or conserving the hypotheses depending on their likelihood: the method should not be overconfident, risking to eliminate the correct hypothesis, returning a set too reduced, nor should it be too pessimistic, returning a set too large. This is the core of this thesis problematic: the goal is to develop a lane-level map-matching method taking into account the solution integrity, in order to provide a set of hypothesis containing the correct solution with a high probability (corresponding to a risk level).

2.6 Experimental Setup and Functional Architecture

This section first presents the hardware and software used for data acquisition, algorithm testing and validation, and then introduces the functional architecture of the developed algorithms.

2.6.1 Experimental Vehicles

Two experimental vehicles have been used during this thesis. Both of them were financed by the ROBOTEX project[5] to equip the Heudiasyc laboratory with experimental platforms for the intelligent vehicle.

2.6.1.1 IRIS

IRIS[6] is a test vehicle used for numerous experiments in the field of intelligent vehicles. It is based on a fully electric Renault Fluence Z.E. (see Fig. 2.21). Instrumented with various sensors (see Section 2.6.2 for more details) connected to an embedded PC, IRIS can be operated as an acquisition vehicle to generate datasets corresponding to different scenarios, with various sources of information.

Figure 2.20 – IRIS test vehicle, based on a Renault Fluence Z.E.

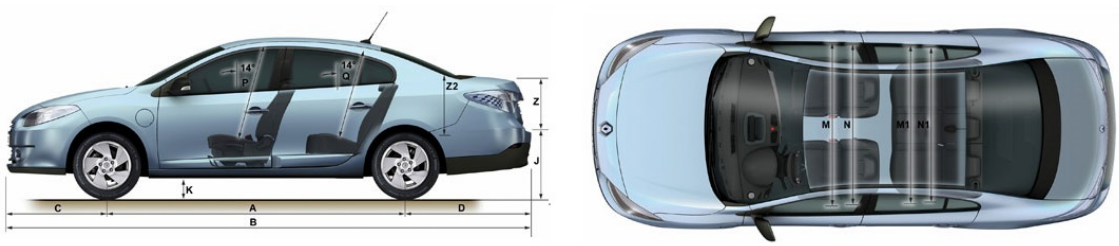


Figure 2.21 – Renault Fluence Z.E.

The computer embedded can run both Linux and Windows. This flexibility enables the use of numerous tools, depending on the need. For example, the Linux system is compatible with the ROS framework[7], which is becoming widespread in

[5]More information can be found at: http://equipex-robotex.fr/en/presentation/
[6]More information can be found at: https://pacpus.hds.utc.fr/vehicles/iris/
[7]More information can be found at: http://www.ros.org/

robotic research. The strength of ROS is its community: being open-source, lots of resources are available online including sensors handling packages.

The Heudiasyc-developed framework Pacpus (see Section 2.6.3) has been used during this thesis. As an in-house solution, Pacpus is well integrated into the laboratory's vehicles. It can be used in the same manner as ROS, to record data from the vehicle and offering replaying capability.

2.6.1.2 APAChE

APAChE[8] are two autonomous vehicles, based on fully electric Renault Zoé (see Fig. 2.22) cars. Used as platforms for experiments at Heudiasyc, they are robotized in order to be controllable by computer (which IRIS is not). The two cars have the same instrumentation and can be used for collaborative driving test scenarios.

Figure 2.22 – One APAChE test vehicle, based on a Renault Zoé.

They are also fitted with numerous sensors (see Section 2.6.2 for more details). The embedded computer is running Ubuntu and use the ROS framework. The APAChE car participated in the 2016 Grand Cooperative Driving Challenge (GCDC): details about the challenge and the vehicle architecture are described by Xu et al. [2018].

2.6.2 Sensors

This section describes succinctly the different sensors equipping the test vehicles. A summary of the sensor availability is presented in Table 2.2.

2.6.2.1 General

An access to the Controller Area Network (CAN) bus is available on both cars. This notably provides odometry information issued directly by the car's proprioceptive sensors, i.e. wheel ticks and yaw rate sensors. This information can then be integrated in a dead-reckoning positioning algorithm [Borenstein and Liqiang Feng, 1996].

[8]More information can be found at: https://pacpus.hds.utc.fr/vehicles/apache/

Table 2.2 – List of sensors and integration status in the cars.

	Sensor Name	IRIS	APAChE
General			
	CAN Information	√	√
	802.11p Communication		√
	Contextual Camera	√	√
Ground Truth			
	NovAtel SPAN-CPT	√	√
GNSS			
	Septentrio AsteRx	√	√
	Ublox 8T	√	√
Smart Camera			
	Mobileye EyeQ2	√	√
Lidar			
	Sick LMS511	√ (rear facing)	
	Sick LD-MRS	√ (front facing)	√ (front facing)
	Velodyne HDL-64E	√ (removable)	
Radar			
	24 GHz Radars		√ (3 front facing)

The APAChE cars are equipped with Cohda Wireless 802.11p modems, enabling automotive wireless communication for Vehicle-to-Everything (V2X) applications. This include Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. Although yielding interesting results [Hery et al., 2017], cooperative positioning is not considered in this thesis.

2.6.2.2 Ground Truth

A Novatel SPAN-CPT is installed on all vehicles to act as ground truth to the experiments. It is a GNSS with INS solution integrated in a single enclosure, compatible with positioning, featuring fiber optic gyros and MEMS accelerometers [NovAtel, 2018]. The announced performance level is 0.4 m of positioning accuracy using differential GPS (DGPS). An output frequency of up to 100 Hz is possible.

2.6.2.3 GNSS Receivers

Two receivers are used:

- Ublox 8T: this automotive grade, low-cost receiver is often used to gather data representative of the receiver found in production cars. It is GPS and GLONASS compatible and is most of the time fitted with a patch-type antenna.

- Septentrio AsteRx2eH PRO: the main feature of this receiver is its dual-frequency compatibility with both GPS and GLONASS. This receiver is used also for automotive tests as low-cost dual-frequency receivers are being introduced to the market and will eventually integrate the automotive world [Broadcom, 2017]. RTK corrections can be integrated to the position calculation. The

AsteRx2eH can also use a two-antenna configuration for more advanced applications, such as heading estimation. This thesis only considered the single antenna configuration, more representative to an automotive application.

2.6.2.4 Smart Camera

Smart cameras constitute the visual-based exteroceptive sensors. Mobileye EyeQ2 systems are fitted in the cars' windshield. These smart cameras provide rich post-processed information, for example, classified object detection, lane markings detection. These enable functionalities such as Time To Collision (TTC) estimation, collision alerts, LDW. The camera system is considered as black box, i.e. only the outputs of the system are known and available.

In this thesis, the main features used are lane marking detections. More details about the line definition system are given in Section 3.3.6.2.

2.6.2.5 Lidars/Radars

These exteroceptive sensors are mentioned for completeness, but are not used in the scope of this thesis. A Sick LD-MRS, 4-layer lidar is fitted in the front bumper of each car.

On IRIS, additional Lidars present: a Sick LMS511 1-layer lidar is installed in the rear bumper, faced backwards. Finally a Velodyne HDL-64E (64 layers) is mountable on the roof of the vehicle.

On APAChE, 3 front radars (24 GHz) complete the vehicle specifications.

2.6.3 Middleware

When coding a software aiming to interact with some external hardware (i.e. other than the computer's component), an interfacing element is necessary: this is the function of a middleware. To this end, Heudiasyc is working with the two previously cited frameworks: ROS and Pacpus frameworks.

Both of them provide a convenient interfacing with the sensors embedded in the cars, using C++. Packages must be developed to process the data coming from given sensors and can then be shared to anyone using the same middleware. The frameworks implement recording capabilities, replay but also "real-time" (i.e. on-the-fly) execution.

The algorithms developed during this thesis have been using primarily the Pacpus framework[9]. Their development has however been conducted with modularity in mind: the software is organized following Object-Oriented Programming (OOP [Stroustrup, 1991]). As such, the different algorithm are well encapsulated, providing a clear interface for input/output. This interface is then virtually adaptable to any framework able to handle the required data, and the algorithm is independent from the framework providing it with data as the interface makes it totally transparent, putting a data handling abstraction layer between the source and the processing.

[9]More information can be found at: https://pacpus.hds.utc.fr/

2.6.4 Functional Architecture

The thesis contributions can be grouped into three main algorithms: Fig. 2.23 shows the global functional architecture of the developed system.

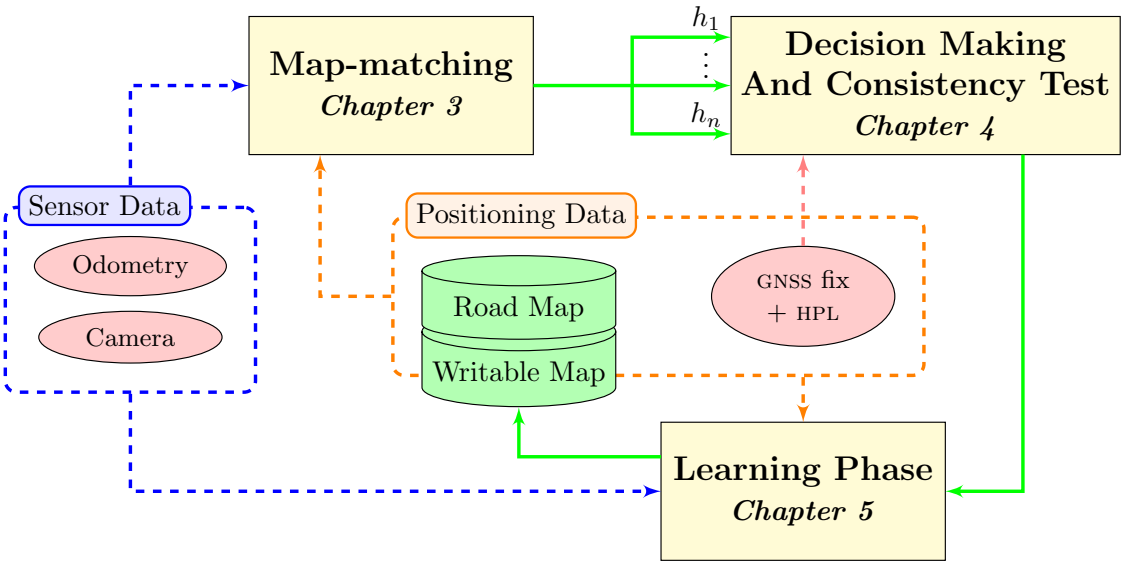


Figure 2.23 – Functional architecture presenting all the elements developed during this thesis and their relationships with each other.

2.6.4.1 Inputs

Three main sources of information are used, split into two groups. The first one is the sensor system, composed of:

- The car's odometry,
- Exteroceptive information from a smart camera (lane detection).

The second group is the localization system composed of:

- A GNSS receiver providing an HPL value,
- The digital road map.

Note that the road map is separated into two parts as it acts both as an information source and a data storing medium. The separation only means that the recording process must not modify the actual map and only refer to its elements.

2.6.4.2 Map-Matching

The first step is the map-matching algorithm. It is presented in Chapter 3. Based on a Particle Filter (PF), its specificity is its ability to output multiple hypotheses if the situation is ambiguous. It takes as input both the sensors and positioning system.

2.6.4.3 Decision Making and Consistency Test

Taking the multiple hypotheses of the map-matching algorithm as input, the consistency test aims to assess the usability of the positioning system by comparing them to the GNSS fix from the receiver. It then outputs a "Use/Don't Use" characterization, signifying if the positioning system result can be used without ambiguities. This test is developed in Chapter 4.

2.6.4.4 Learning Phase

The last step is the learning phase: if the previous test determines that the position is usable, data is recorded in the map. To do this, the learning algorithm takes the sensors and the positioning system as input and stores information about their performance in the map. This phase along with learning examples are presented in Chapter 5.

2.7 Conclusion

The different systems taking part in this thesis have been presented in this chapter. The central element of this research is the digital road map: different formalisms have been presented notably the specification of the chosen map format. Map must be used with an absolute positioning system: the GNSS complete the positioning system. But satellite-based positioning is tied to numerous sources of perturbation, degrading the system's accuracy: some metrics are needed to characterize the level of confidence that can be given to the positioning system. The presented integrity metrics may not always be sufficient for use in a terrestrial vehicle use case: map-matching is a solution processing map information in order to cope with some level of uncertainty.

Reviewing different map-matching methods, their weakness facing high-accuracy lane-level road maps has been raised. This observation leads to the development of the map-matching algorithm presented in this thesis: combining the uncertainty of the GNSS positioning and the high-accuracy of the used map, multiple map-matching hypotheses management is unavoidable. A mechanism is then needed to fall back to a unique solution when needed and possible.

Some assumptions are made during this thesis: the sensors are considered as "blackboxes", i.e. only returning already processed features, with potentially some uncertainty characterization. This includes the GNSS receiver that returns a computed positioning and not raw satellite measurement. A last important assumption is that the map is correct, accurate and up to date.

The following chapters will develop the different elements summarized by Fig. 2.23, beginning with the multi-hypothesis map-matching method.

Chapter 3

Simultaneous Map-Matching And Localization

Contents

3.1 Introduction 56
3.2 Elements on Bayesian Sequential Estimation Theory . . 57
3.2.1 Bayes' Theorem 57
3.2.2 Bayesian Filtering 58
3.2.3 Kalman Filtering 60
3.2.4 Particle Filtering 62
3.3 Simultaneous Map-Matching and Localization Using Particle Filtering 68
3.3.1 Particle Definition 70
3.3.2 Inputs 70
3.3.3 Output 71
3.3.4 Initialization Step 72
3.3.5 Main Loop: Updating State and Matching 76
3.3.6 Likelihood Calculation 78
3.3.7 Resampling 84
3.3.8 Hypotheses Determination 84
3.4 Experimental Results and Evaluation 85
3.4.1 Experimental Setup 85
3.4.2 System Initialization and Study of the PF Convergence . 88
3.4.3 System Performance in Tracking Mode 91
3.4.4 Localization Results 97
3.4.5 Map-Matching Consistency 97
3.5 Conclusion 99

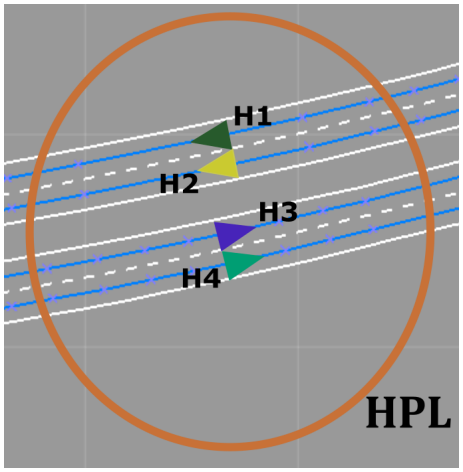


Figure 3.1 – Ambiguous matching situation caused by the use of an HPL for positioning. Some possible Map-Matching (MM) solutions are highlighted.

3.1 Introduction

Using high-accuracy lane-level road maps with automotive GNSS receivers creates most of the time ambiguous situations in which the correct map-matching solution cannot be determined using solely the GNSS fix. A critical situation for intelligent vehicles (a fortiori for autonomous vehicles) is a wrongly matched position: the system could take unsuitable decisions, being supplied with incorrect information about the environment (e.g., wrong speed limit).

Positioning integrity of a GNSS fix is characterized by an HPL (with a given integrity risk) which represents a bounded uncertain position domain. In this chapter, this domain is used as positioning information instead of a point estimate. This value frequently reaches tens of meters in urban environments when using standalone GNSS computation [Cosmen-Schortmann et al., 2008]. Used on lane-level maps, this area covers in most cases multiple lanes (see Fig. 3.1): a unique matching result cannot be determined with certainty. To get a correct matching at any time, multi-hypothesis management is an interesting approach. Classic mono-hypothesis methods choose the most probable one, discarding the others, including potentially the correct one. On the other hand, multi-hypothesis methods estimate consequently all likely hypotheses. This effectively prevents eliminating the correct matching when an ambiguous situation would force a quasi-arbitrary choice in the mono-hypothesis case (e.g., two parallel lanes). The ability to manage multiple matching hypotheses is therefore a key feature to ensure the correctness of the result in presence of ambiguities due to positioning uncertainty.

A MM method based on PF is designed in this chapter. It takes into account matching ambiguities and aims to avoid returning a wrong matching solution. Relying strongly on DR, the method solves simultaneously MM and localization problems. After having recalled some elements on Bayesian estimation theory, this chapter exposes the implementation of the filter, the efficient use of the map information during the MM process and the fusion of exteroceptive information. Experimental results complete the filter's description by focusing notably on the initialization phase. A study of the filter's performance in different situations (e.g., urban environment, lane forking) is presented.

3.2 Elements on Bayesian Sequential Estimation Theory

This section presents basic notions necessary to understand Bayesian estimation and therefore the PF used in the presented algorithm.

3.2.1 Bayes' Theorem

As its name suggests, Bayesian Sequential Estimation is based on Bayes' theorem. It is presented here with a description of how it can be applied to estimation problems [Durrant-Whyte et al., 2003].

3.2.1.1 Statement With Probability Density Functions

For two random variables x and y, the joint distribution noted is $p(x, y)$. Integrating the Probability Density Function (PDF) $p(x, y)$ over the variable x gives the marginal PDF $p(y)$:

$$p(y) = \int p(x, y)dx \tag{3.1}$$

The conditional PDF $p(x|y)$ is defined by:

$$p(x|y) \triangleq \frac{p(x, y)}{p(y)} \tag{3.2}$$

This can be written as:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \tag{3.3}$$

This is the Bayes Theorem. It can then be deduced:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{3.4}$$

Using the total probability theorem, the marginal PDF $p(y)$ can also be written as:

$$p(y) = \int p(y|x)p(x)dx \tag{3.5}$$

Bayes' formula can therefore be rewritten as:

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)dx} \tag{3.6}$$

The denominator can be considered as a normalizing constant such that $p(x|y)$ is a valid PDF (i.e. its integral over the support equals one). (3.6) can thus be rewritten:

$$p(x|y) = \eta \cdot p(y|x)p(x) \tag{3.7}$$

with η, a normalizing constant.

Bayes' formula is generalized with three random variables x, y, z as follows:

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)} \tag{3.8}$$

$$p(x|y,z) = \eta \cdot p(y|x,z)p(x|z) \tag{3.9}$$

3.2.1.2 Notation for Probabilistic Data Fusion

Bayes' theorem fits well the needs of data fusion by taking the following consideration.

First, let x be the unknown state that the fusion is trying to determine, and let y be the information gathered from sensors. The goal of data fusion is to use y in order to refine the knowledge we have on x. The meaning of the different PDFs are then:

- $p(x)$: the prior PDF, indicating the expected value of x before taking a measurement,
- $p(x|y)$: the posterior PDF, indicating the expected value of x after the measurement y (the probability to be in the state x, having observed y),
- $p(y|x)$: the conditional PDF.

The conditional distribution $p(y|x)$ can be seen in two ways:

- As a *Sensor Model*: the distribution is constructed by fixing a particular value $x = x_p$. The PDF $p(y|x = x_p)$ then depends only on the variable y. It translates to the probability of observing y, considering the system being in the state x_p.
- As a *Likelihood Function*: conversely, measurements are made once the sensor model exists yielding $y = y_m$. From this, $p(y = y_m|x)$ is now a distribution in x, translating the likelihood of a state x given the measurement y_m.

In a practical implementation, $p(y|x)$ is constructed as a function of both variables (or a matrix in the discrete form). For each fixed value of x, a distribution in y is defined. Therefore as x varies, a family of distributions in y is created.

3.2.2 Bayesian Filtering

The goal of Bayesian filtering is to estimate sequentially the posterior probability density $p(x|y)$ characterizing the distribution of the state x in the state space, knowing the measures y from a prior density. This probability density $p(x|y)$ is generally difficult to characterize while the likelihood $p(y|x)$ can be obtained easily using an observation model.

A Bayesian filter is divided into three phases: prediction, update and estimation [Chen, 2003; Ristic et al., 2004]. This sequential order is kept in the following to describe the theory but note that in practice, the prediction stage is executed last for real-time optimization reasons.

Consider the non-linear system represented by the following discrete time equations:

$$\begin{cases} x_{k+1} &= f(x_k, u_k) + \alpha_k \\ y_k &= g(x_k) + \beta_k \end{cases} \tag{3.10}$$

where α_k and β_k represent the model and observation noises with PDF p_α and p_β respectively. They are supposed additive, white and independent. u_k is the the input..

3.2.2.1 Prediction

Let $y_{1:k} = \{y_i\}_{i=1\cdots k}$ be all measurements up to time k. The prediction step corresponds to the calculation of $p(x_k|y_{1:k-1})$. This PDF can be considered as a marginal probability density with respect to x_{k-1}:

$$p\left(x_k|y_{1:k-1}\right) = \int p\left(x_k, x_{k-1}|y_{1:k-1}\right) dx_{k-1} \tag{3.11}$$

Using Bayes' formula as (3.3) in (3.11):

$$p\left(x_k, x_{k-1}|y_{1:k-1}\right) = p\left(x_k|x_{k-1}, y_{1:k-1}\right) p\left(x_{k-1}|y_{1:k-1}\right) \tag{3.12}$$

Let us suppose that the system is Markov. The Markov hypothesis states that any knowledge about the system is contained in the state x_{k-1}. This conducts to the conditional independence.

$$p\left(x_k|x_{k-1}, y_{1:k-1}\right) = p\left(x_k|x_{k-1}\right) \tag{3.13}$$

Integrating the previous results in (3.11), a new expression, known as Chapman-Kolmogorov equation [Jazwinski, 1970], is obtained:

$$p\left(x_k|y_{1:k-1}\right) = \int p\left(x_k|x_{k-1}\right) p\left(x_{k-1}|y_{1:k-1}\right) dx_{k-1} \tag{3.14}$$

3.2.2.2 Update

Now consider the nonlinear observation model $y_k = g\left(x_k\right) + \beta_k$.

Using Bayes' theorem, as in (3.4), allows to move from a prior distribution $p(x)$ to a posterior $p(x|y)$, using the observation likelihood $p(y|x)$. Considering all the measures $y_{1:k}$, we have:

$$p\left(x_k|y_{1:k}\right) = p\left(x_k|y_k, y_{1:k-1}\right) = \frac{p\left(y_k|x_k, y_{1:k-1}\right) \cdot p\left(x_k|y_{1:k-1}\right)}{p\left(y_k|y_{1:k-1}\right)} \tag{3.15}$$

Once again, under conditional independence assumption:

$$p\left(y_k|x_k, y_{1:k-1}\right) = p\left(y_k|x_k\right) \tag{3.16}$$

The likelihood function $p\left(y_k|x_k\right)$ is given by the model of the observation noise.

$$p\left(x_k|y_{1:k}\right) = \eta_k . p\left(y_k|x_k\right) . p\left(x_k|y_{1:k-1}\right) \tag{3.17}$$

where η_k is the normalization term.

3.2.2.3 Estimation

This step focuses on finding the first two statistical moments to characterize the result. From the estimation of the posterior distribution $p(x_k|y_{1:k})$, the Minimum Mean-Square Error (MMSE) can be used as an estimate. It is obtained by considering the following integral over the support of the density that represents mathematical expectation:

$$\hat{x}_k = \int x_k.p(x_k|y_{1:k})\,dx_k \tag{3.18}$$

The covariance matrix can be calculated from:

$$P_k = \int (x_k - \hat{x}_k)(x_k - \hat{x}_k)^T p(x_k|y_{1:k})\,dx_k \tag{3.19}$$

3.2.3 Kalman Filtering

A direct application of Bayesian sequential filtering is the KF [Kalman, 1960]. In a stochastic context, "filter" has the meaning of "state observer". The KF estimates the state with a second order model, limiting the calculation to the mean and covariance of the underlining distribution. This study is only complete in the Gaussian case, which is generally an assumption taken when using a KF.

3.2.3.1 Considered Problem and Notations

The system is modeled as a simplified discrete state space:

$$\begin{cases} x_{k+1} &= A \cdot x_k + B \cdot u_k + \alpha_k \\ y_k &= C \cdot x_k + \beta_k \end{cases} \tag{3.20}$$

Where dim(x_k)=n dim(y_k)=m and dim(u_k)=p. Model noise α_k and observation noise β_k are supposed centered (i.e. without bias) and white so that:

$$\begin{aligned} E(\alpha_k) &= 0 \\ E(\beta_k) &= 0 \end{aligned} \tag{3.21}$$

Their respective covariance matrices Q and R are known and considered constant here for simplification:

$$\begin{aligned} Var(\alpha_k) &= Q \\ Var(\beta_k) &= R \end{aligned} \tag{3.22}$$

Moreover, they are assumed to be uncorrelated so $E(\alpha_k \cdot \beta_k^T) = 0$.

The problem is to find a good estimate of the state x_k given observations y_k from the initial time to the instant l. The estimate of x_k is denoted $\hat{x}_{k|l}$ that is the estimate of x_k knowing everything that happened until the instant l. The covariance matrix of the estimation error will be noted:

$$P_{k|l} = E\left\{\left(\hat{x}_{k|l} - x_k\right) \cdot \left(\hat{x}_{k|l} - x_k\right)^T\right\} \tag{3.23}$$

The following notations will be used:

- $\hat{x}_{k|k}$, state estimation

- $e_{k|k} = \hat{x}_{k|k} - x_k$, estimation error
- $P_{k|k}$, covariance matrix of the estimation error
- $\hat{x}_{k+1|k}$, (one-step) prediction
- $e_{k+1|k} = \hat{x}_{k+1|k} - x_{k+1}$, prediction error
- $P_{k+1|k}$, covariance matrix of the prediction error

3.2.3.2 Estimator/Predictor State Observer

An observer is able to determine the system state by *observing* only the inputs and outputs of the system on a finite time interval. We consider observers in the form of:

$$\begin{cases} \hat{x}_{k|k} & = & \hat{x}_{k|k-1} & +K\left(y_k - C \cdot \hat{x}_{k|k-1}\right) \\ \hat{x}_{k+1|k} & = & A \cdot \hat{x}_{k|k} & +B \cdot u_k \end{cases} \tag{3.24}$$

where K is a gain matrix to define and not necessarily constant. Note the recursive nature of the observer: only the previous state and current inputs and outputs are needed. Two steps are described in (3.24): the first line represents the estimation of the current state. It is the element needed at time step k. The second line is the prediction step, during which the observer computes the following state given the evolution model provided and the inputs.

3.2.3.3 Recurrence of Covariance Matrices

A recurrence relationship exists to calculate the estimation error covariance matrix:

$$P_{k|k} = (I - KC) \cdot P_{k|k-1} \cdot (I - KC)^T + K \cdot R \cdot K^T \tag{3.25}$$

Likewise, the prediction error covariance matrix can be expressed as:

$$P_{k+1|k} \quad = A \cdot P_{k|k} \cdot A^T + Q \tag{3.26}$$

These last two formulas provide a recursive way to compute the second statistical moment of the model.

3.2.3.4 Application: Using Kalman Gain Matrix

The KF is an estimator/predictor observer as presented previously, using as K matrix the Kalman gain matrix, defined as:

$$K = P_{k|k-1} \cdot C^T \left(C \cdot P_{k|k-1} \cdot C^T + R\right)^{-1} \tag{3.27}$$

This gain yields an optimal estimation in the sense of the orthogonality principle [Maybeck, 1982].

The KF execution then compute sequentially the different steps described previously:

1. Estimation: from (3.24)(first line) and (3.25) for the associated covariance.
2. Prediction: from (3.24)(second line) and (3.26) for the associated covariance.

3. Kalman gain update, from (3.27).

The filter then loops executing the procedure at each time step with new measures. The KF is perfectly fit for data fusion. It provides a way to integrate optimally measurements into a model in order to improve the estimation of the system state. The gain matrix enables the modulation of the confidence given either to the previous estimate or the current measures.

But the method is not exempt of default. For instance, the filter is based on a centered Gaussian assumption that may not be respected in certain applications. Moreover, the implementation of the filter can bear heavy calculation in the case of high dimensional problem. This is caused by the need to invert matrices at each time step.

In case of non-linear systems, more computation are needed: one can use for example an EKF with Jacobian matrices or an Unscented Kalman Filter (UKF) with sigma points [Julier and Uhlmann, 1997].

Finally, KF are mono-hypothesis. This a major drawback for applications where multiple hypotheses may need to be tracked. Extensions have been developed to introduce a multi-hypothesis management in KF such as the Multi-hypothesis Tracking (MHT) algorithm [Blackman, 2004]. This algorithm relies on the use of Gaussian Mixtures as posterior belief, effectively creating multi-modal distributions. MHT does not fit the current study—map-matching on lane-level maps—as it is more natural to consider each mode (i.e. a single lane) separately, especially when multiple factors enter in the likelihood calculation.

3.2.4 Particle Filtering

The PF is an alternative to KF. It is a sub-optimal solution, also based on the equations of Bayesian filtering (3.14) and (3.17). It calculates sequentially $p(x_k|y_{1:k})$ from a given initial density. It has interesting characteristics such as being able to represent a variety of distributions, including non-Gaussian [Arulampalam et al., 2002], thus being highly suitable to multi-modal considerations. PF uses the Monte Carlo method to compute integrals.

3.2.4.1 Integral Calculation Using Monte Carlo Method

PF considers directly the Bayesian filtering estimation phase during which it is necessary to compute integrals (i.e. (3.18) and (3.19)). Monte Carlo methods provide a solution to the problem of Bayesian filtering by being able to estimate such calculation.

Principle Consider the following integral equation, generalizing (3.18) and (3.19):

$$I = \int h\left(x\right) p\left(x\right) dx \tag{3.28}$$

where $p\left(x\right)$ is a given PDF. If N samples x^i can be generated independently in accordance with the probability density $p\left(x\right)$ then an estimate of I is:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} h\left(x^i\right) \tag{3.29}$$

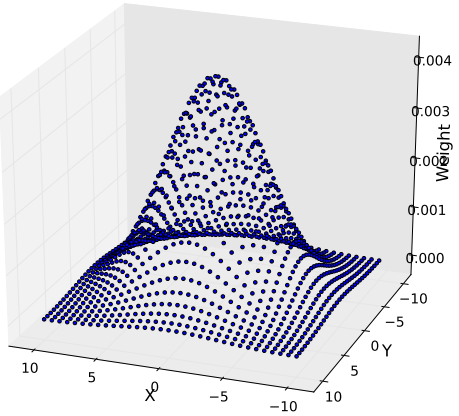


Figure 3.2 – Using a set of particles, PF allows the representation of arbitrary distribution (a Gaussian distribution in this figure). The particles evolve in the state space and a weight w^i allows to give a likelihood for every point and thus shape the distribution. In this drawing, the particles are distributed along a grid. In practice, they are randomly spread in the state space.

It is shown with the central limit theorem that there is convergence when $N \rightarrow \infty$.

Integration With an Instrumental Generator If the distribution $p(x)$ is unknown, drawing sample from it is not possible. If a random number generator $q(x)$ —often called instrumental generator or importance function in the literature—is available, (3.28) can then be rewritten:

$$I = \int h(x) \frac{p(x)}{q(x)} q(x)\, dx \tag{3.30}$$

If the instrumental law $q(x)$ is close enough to $p(x)$, then it is possible to accurately calculate the integral. This property is crucial when implementing such a method. It is also important when implementing a PF.

Application to Bayesian Estimation The main idea of PF is to approximate $p(x_k|y_{1:k})$ by a finite set of particles $\{x_k^i\}_{i=1\cdots N}$, each with an affected weight. The weight of each particle reflects the probability that the particle is at a relevant position of the state space. The weights are thus the elements shaping the distribution .They allow a representation of any arbitrary distribution [Shao et al., 2010]. For example, the particles in Fig. 3.6 describe a Gaussian distribution. It is possible to model more complex situations, such as multimodal distributions, which is one of the strengths of the PF.

Sequential Monte Carlo methods (another name given to PF) allow an implementation of a Bayesian filter through numerical approximation. Discrete samples are obtained by an instrumental generator to approximate a probability distribution that is unknown analytically. The greater the number of particles, the finer the approximation. With the evolution of computer technology, these filters have become popular and are applied in robotics applications, such as positioning problems [Gustafsson et al., 2002].

3.2.4.2 Bootstrap Particle Filter

In practice, a PF needs a random number generator from which samples can be generated. Let $q\left(x_k|x_{0:k-1}, y_{1:k}\right)$ be the PDF of the numbers generated. This approach conducts to manage a weight ω_k associated to any particle. This is called *importance sampling*.

Van Der Merwe et al. [2001] expose a recursive method for weight calculation. Under Markov assumptions the current weight can be computed by:

$$\omega_k = \omega_{k-1} \cdot \frac{p\left(y_k|x_k\right) p\left(x_k|x_{k-1}\right)}{q\left(x_k|x_{0:k-1}, y_{1:k}\right)} \tag{3.31}$$

This provides a sequential mechanism to estimate the weight of importance, given the instrumental law $q\left(x_k|x_{0:k-1}, y_{1:k}\right)$. Choosing this distribution correctly is a crucial point. The method called *bootstrap* considers:

$$q\left(x_k|x_{0:k-1}, y_{1:k}\right) = p\left(x_k|x_{k-1}\right) \tag{3.32}$$

It has the advantage to lead to a simple formulation of (3.31):

$$\omega_k = \omega_{k-1} \cdot p\left(y_k|x_k\right) \tag{3.33}$$

The particles are then drawn using the prediction model, i.e. the evolution model of the state and the model noise. Updating the weight of the particles simply consists in multiplying the weight of each particle by the likelihood. For these advantages, this is the method implemented during this thesis.

3.2.4.3 State and Covariance Estimation

Suppose we do a Bootstrap PF and we have a model of the measurement noise distribution (denoted p_β), for instance a Gaussian model.

For a given measurement y_k, the weights of the particles are updated as follows:

$$\omega^i_{k|k} = \omega^i_{k|k-1} \cdot p_\beta\left(y_k - g\left(x^i_{k|k-1}\right)\right) \tag{3.34}$$

We normalize the weights:

$$\tilde{\omega}^i_{k|k} = \frac{\omega^i_{k|k}}{\sum \omega^i_{k|k}} \tag{3.35}$$

Estimates of the state vector and of the covariance matrix are given by:

$$\begin{cases} \hat{x}_{k|k} & = \sum_{i=1}^{N} \tilde{\omega}^i_{k|k} \cdot x^i_k \\ P_{k|k} & = \sum_{i=1}^{N} \tilde{\omega}^i_{k|k} \cdot \left(x^i_k - \hat{x}_{k|k}\right) \cdot \left(x^i_k - \hat{x}_{k|k}\right)^T \end{cases} \tag{3.36}$$

3.2.4.4 SIS Algorithm

The PF execution depends on the knowledge of the noise models and needs a random number generator for particle drawing. The bootstrap particle filter algorithm with a Sequential Importance Sampling (SIS) implementation is summarized in Algorithm 3.1.

Algorithme 3.1 SIS implementation of the PF.

```
function PARTICLE FILTER SIS
    Initialization: generate N particles x^i from prior with associated weight w^i
    while getting inputs do
        y ← Get_Measures()
        WEIGHT_UPDATE                                   ▷ Eq. (3.34)
        WEIGHT_NORMALIZING                              ▷ Eq. (3.35)
        STATE_COV_ESTIMATION                            ▷ Eq. (3.36)

        PREDICTION        ▷ Using evolution model noise and random generator
    end while
end function
```

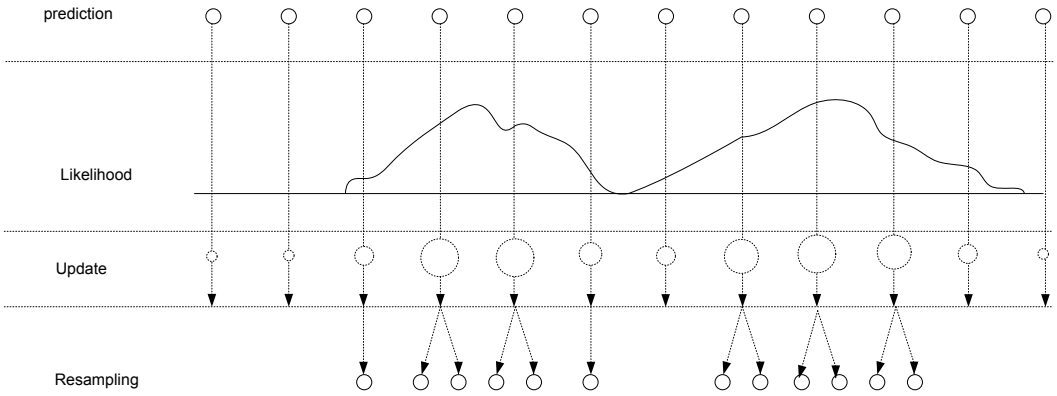


Figure 3.3 – Illustration of the resampling of a particle filter.

3.2.4.5 SIR Algorithm

Systematic Resampling The SIS particle filter presented above is not viable in practice because the weights quickly degenerate towards zero. This is because the particles are not limited in their evolution and can explore the state space entirely using the noisy model. When a particle go astray, its weight becomes infinitely small and rendering it insignificant. After a few samples, the number of significant particles becomes too low: the filter diverged.

To address this problem, a *resampling* procedure is introduced: particles with low weight are eliminated and the most significant are possibly duplicated several times (their clones will follow different paths in the state space as each model noise will be different). This procedure ensures that the set of particles only keeps the most likely particles, but still providing flexibility to other to evolve. Generally, after a resampling step, all weights are reset to be equal, as shown in Fig. 3.3.

Resampling is a fundamental step of particle filtering to prevent degeneration. A SIS-PF with resampling is called Sequential Importance Resampling PF (SIR-PF [Arulampalam et al., 2002]). The original steps of the algorithm stay identical (see Algorithm 3.1), only the resampling step is added just before the prediction step.

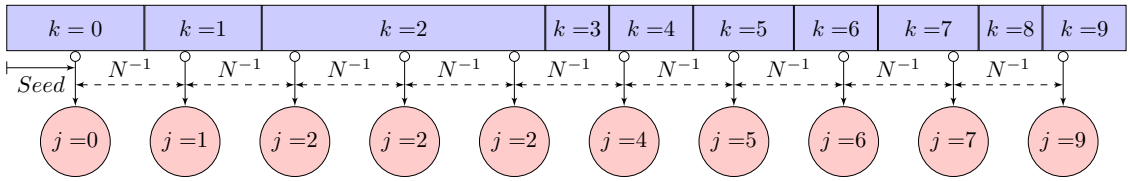


Figure 3.4 – Graphical illustration of the low variance resampling. The rectangles represent the old particles, their width is proportional to their weight. Circles are the new particles drawn from the old set. For instance, particle 2 is duplicated 3 times and particle 3 is eliminated.

Adaptive Resampling Frequency Resampling can be done systematically at each stage but, as this procedure requires a lot of computing, it may slow down the overall process. Moreover, a too frequent resampling can induce a particle set impoverishment.

For these reasons, a criteria can be defined in order to resample only when necessary. This can be done by taking into account the proportion of significant particles, i.e. the percentage of particles having a relatively high weight. An estimate of this number of effective particles N_{eff} can be computed using [Bergman, 1999]:

$$N_{eff} = \frac{1}{\sum_1^N \left(\omega_k^i\right)^2} \tag{3.37}$$

- The number of effective particles is maximized when all the weights are equal. In this case, $N_{eff} = N$ (all particles are significant, N in total).
- It is minimum when only a single weight is not zero. In this case, the weight of this particle is 1 and $N_{eff} = 1$.

A resampling is executed only when necessary, i.e. when N_{eff} falls below a set threshold. A common choice in the literature sets the threshold to a value of 2/3 of the total number of particles [Bergman, 1999; Gustafsson et al., 2002].

Resampling Strategy The classic resampling method used for PF draws new particles randomly from the original set with a probability proportional to the weight of the original particles [Bergman, 1999]. The Multinomial Resampling method is very popular and often used in many PF [Douc and Cappe, 2005]. This randomness can cause an impoverishment by drawing exclusively from a subset of the particles, as successive drawings are independent and uncontrolled.

A low variance resampling [Thrun et al., 2002] is an alternative often used in robotics.It avoids doing a repetition of N random draws and favors a good distribution of the retained particles, spread over the whole original set without risking to favor any interval. The complete procedure is described in Algorithm 3.2.

First, a correspondence between each particle is established with the interval $[0, 1]$ (each value in the interval is linked to a single particle). This is done by summing the particle weights consecutively to create an empirical cumulative density function, as the weights sum to 1. It is represented by the successive rectangles in Fig. 3.4, the difference in length denotes the different value the weight can take.

Then, the resampling procedure performs a single random draw to determine a seed s in the interval $[0; N^{-1}[$, N denoting the total number of particles and then

Algorithme 3.2 Low variance resampling.

```
function RESAMPLING
    N_eff ← (Σ_i (w_k^i)^2)^{-1}
    if N_eff < Threshold then
        Seed ← rand(0, N^{-1})
        w_sum ← w_k^0                          ▷ w_sum: cumulative sum of weights
        j ← 0
        for i ← 0, N − 1 do
            u ← Seed + iN^{-1}
            while w_sum < u do
                j ← j + 1
                w_sum ← w_sum + w_k^j
            end while
            new_particles[i] ← old_particles[j]
        end for
    end if
    NORMALIZE WEIGHTS                          ▷ always called, even if no resampling
end function
```

add repetitively N^{-1} to it. This generates a list of N cumulative sums of weights, from which the N resampled particles $\widehat{\text{Part}}^i$ can be chosen from the original set $Part$ by determining:

$$\widehat{\text{Part}}^i = \text{Part}^i \tag{3.38}$$

for $i = \{0, ..., N-1\}$ with

$$j = \underset{l}{\text{argmin}} \sum_{m=0}^{l} w^m \geq (s + \frac{i}{N}) \tag{3.39}$$

Graphically in Fig. 3.4, when a cumulative sum "falls" into a rectangle, the corresponding particle is added to the new particle set. This method takes into account the weight of each original particle: for example, the particle $k = 2$ bears a consequent weight, causing it to be drawn 3 times during the resampling procedure. On the other hand, particle $k = 3$ with a small weight does not get drawn and is eliminated from the filter.

As exposed in Section 3.3.5, the number of particle may vary during the execution of the filter. This is transparent during the resampling process, as the important point is that the total weight is normalized: the cumulative density is computed in the same manner. Nevertheless, it is during the resampling step that the number of particle is regulated: only N new particles are systematically drawn, naturally eliminating the exceeding ones, with the lowest likelihood.

Finally, in all cases the particle weights are normalized as they most likely do not sum up to 1 anymore. Even without resampling, the natural evolution of the filter causes weight changes needing a normalization at each iteration.

After following all theses step, the particle set is ready for the next iteration of the filter.

3.2.4.6 Strengths and Weaknesses

The PF theory is relatively simple to apply but the method also has some drawbacks. Monte-Carlo methods accuracy is based on the number of samples (the more particles the better): this causes a heaviness of calculation. Without an efficient implementation, some difficulties can arise when needing real-time implementation. For this reason, the number of particles is an important parameter: a compromise between computation load and filter performance must be found.

Another difficulty is the need of correct models: the method is sensitive to bias. To cope with this, a solution may be to increase the noise variance in the model. This in turn reduces the filter's performance.

On the other hand, the filter is simple to implement and the individual calculations are not complex. The PF is thus scalable to any computational power available (the limiting factor being the accuracy needed), by using a variable number of particles. The nature of the filter renders it highly parallelizable, being able to take advantage of the highly parallel architectures (e.g., Graphic Processing Units, GPUs). The possibility of representing arbitrary distributions is also a major strength, especially multi-modal distributions in the case of multiple hypotheses tracking.

The following section presents an application of PF to the problematic of map-matching on high-definition road maps in the context of intelligent vehicles.

3.3 Simultaneous Map-Matching and Localization Using Particle Filtering

Previous works have investigated PF for MM purposes. Oh et al. [2004] included map priors with GPS positioning in a PF-based pedestrian localization algorithm. The map data represents a probability distribution over the test area, defining the likelihood of the location. The map used in this research does not correspond to automotive digital road map, but the authors demonstrated the use of map data for positioning enhancement. The particles were used only to represent the position posterior density, and not to manage multiple hypotheses.

The same principle was applied to vehicles by Peker et al. [2011], who in addition merged odometry information. As the vehicle is assumed to be evolving on-road, the map is built using the topology of the road network, providing information on the probability of any given trajectory. The results focus not only on raw positioning but on the correctness of the MM, being oriented toward automotive applications. Once again, even though the algorithm uses the particles to consider multiple possible matchings, the output is still mono-hypothesis. Choosing the hypothesis with the highest weight can lead to errors.

Filters can also process information from exteroceptive sensors, as done by Gu et al. [2015], where lane marking detections from a camera were used. Focusing on dense urban environments prone to multipath problems, the authors processed the GNSS information using 3D maps of buildings to detect this kind of error. This provides a cleaner GNSS position to be integrated into the positioning system. It is then fused with lane markings using a PF and map information.

The goal of the method described here differs from the most common MM methods. For instance, in classical automotive systems used for turn-by-turn navigation,

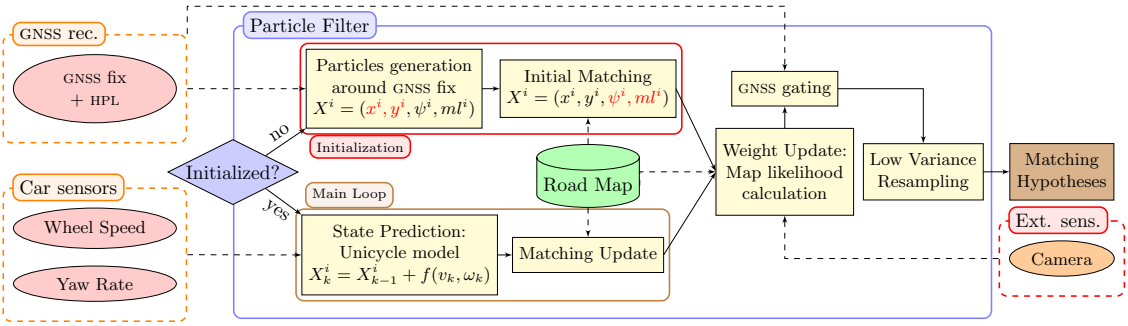


Figure 3.5 – Flowchart of the map-matching process. The inputs are denoted by dashed lines/boxes. Based on a particle filter, the algorithm takes as inputs a GNSS fix with an HPL and the car dead-reckoning. The process is separated in two sections: the initialization which is heavy-computational and the main loop being highly efficient. Exteroceptive information is used during the likelihood calculation step.

MM should give the user a *single* position estimate as a result. This corresponds to the usual requirement of these systems: a single position must be used to calculate a route and, in most cases, if this position is erroneous, the user is able to notice the error, disregard the given information and wait for a correct matching.

MM for autonomous navigation systems, on the other hand, must not be over-confident about its results: the worst-case scenario would be to provide an erroneous single position, since there may be no human to detect the error and correct it. If the matching is ambiguous, then the algorithm should retain the ambiguity and not make a decision. This illustrates the notion of *MM integrity* [Quddus et al., 2006b], which will be developed in Chapter 4. In the past, reference algorithms [Quddus et al., 2007] often addressed map-matching only via a roadway level map (i.e. where roads are described by a single link for each direction). More recent research tends to be oriented toward lane-level maps [Betaille and Toledo-Moreo, 2010; Toledo-Moreo et al., 2009], which present a lot of ambiguities that must be taken into consideration. For instance, lanes belonging to the same road have the same heading and are only separated by a few meters (3.5 m for the standard lane width). Selloum et al. [2009] proposed an evolution model adapted to such maps, taking into account the local curvature of the road, described in their clothoid-based map "Emap" (for Enhanced map).

In this context, the ability to track multiple MM hypotheses is relevant to take into account the ambiguities, hence the choice of a PF [Arulampalam et al., 2002]. This is a commonly used method for map-matching [Gustafsson et al., 2002; Levinson et al., 2007; Rabe et al., 2017; Szottka, 2013]. Its ability to manage multiple hypotheses is interesting in this context for providing a level of integrity, especially in ambiguous situations, where single hypothesis methods could lose track of the correct solution. This PF implement measures to avoid ambiguities. It explores and retains a number of likely hypotheses, which is made possible by an efficient use of the map. This chapter describes the different steps of the MM algorithm developed in this study (see Fig. 3.5).

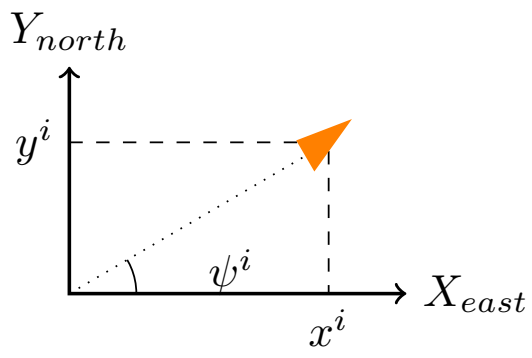


Figure 3.6 – The coordinates correspond to the Cartesian coordinates in (x^i, y^i), with axes pointing respectively to the east and north. ψ^i is the angle form with the east axis, respecting the trigonometric convention.

3.3.1 Particle Definition

The PF is used to estimate the posterior distribution of the car's position given its sensors inputs and the map. The particles model the car's 2D pose X_p^i (composed of the Cartesian 2D coordinates (x^i, y^i) and heading ψ^i). This spatial description is completed with ml^i, the ID of the map link it is matched to. It is the element from which the end result of the filter is inferred: the map link on which the car is currently driving. (3.40) describes the complete state X^i that each particle bears, representing a single position hypothesis with its map-matched solution. It is illustrated in Fig. 3.6.

$$X^i = (X_p^i, \mathrm{ml}^i) = (x^i, y^i, \psi^i, \mathrm{ml}^i) \tag{3.40}$$

Additionally, each particle possesses a weight w^i characterizing its likelihood as a positioning solution. The complete structure of a particle is then given by (3.41):

$$\mathrm{Part}^i = (X^i, w^i) \tag{3.41}$$

The following sections give details about the different steps depicted in Fig. 3.5.

3.3.2 Inputs

Algorithm 3.3 describes the interface of the filter, corresponding to the outer box labeled "Particle Filter". The inputs of the filter are data from sensors embedded in the car (lines 3 to 6):

- **Vehicle speed and yaw rate**, provided by the car's proprioceptive sensors needed for ADAS such as ABS or ESP. These information are available in CAN messages.
- **GNSS position**, computed from a GNSS receiver. The one used in this thesis is representative of the systems found in consumer vehicles (i.e. low-cost solution). It is used along with position covariance and HPL information.
- **Lane marking detections**, returned by an exteroceptive sensor. The algorithm uses feature-based inputs (i.e. the detection information directly). A

Algorithme 3.3 Filter's input interface.

```
function MAP MATCHING
    while getting inputs do
        v ← Speed from Odometry
        ψ ← Yaw Rate from Odometry
        P_GNSS ← GNSS Fix from GNSS receiver
        L_cam ← Lane Detection from Camera

        PARTICLE FILTER(v, ψ, P_GNSS, L_cam)          ▷ see Algorithm 3.4
    end while
end function
```

Algorithme 3.4 Main steps of the PF.

```
function PARTICLE FILTER(v, ψ, P_GNSS, L_cam)
    if first GNSS fix then
        INITIALIZATION(P_GNSS)          ▷ Executed only once, see Algorithm 3.5
    else
        MAIN LOOP(v, ψ)                 ▷ Subsequent iterations, see Algorithm 3.6
    end if
    LIKELIHOOD PROCESSING(P_GNSS, L_cam)          ▷ see Algorithm 3.7
    RESAMPLING                                    ▷ see Algorithm 3.2
    return Hypotheses Set
end function
```

smart camera system returning a description of the detected lane markings has been used. Other types of sensor, such as lidars, can be used as long as the required processing is done beforehand [Hata and Wolf, 2014; Zhang, 2010].

The filter is executed when the input data are available. As the inputs arrive at a fixed frequency, the execution rate is synchronized on the input rates. Algorithm 3.4 exposes the main steps of the PF. Each function does not need all the inputs and can be executed using only a subset. The different steps may then appear to run at different rates, but in practice, the execution follows the highest frequency, details are given in the following sections describing the different functions.

3.3.3 Output

The most important difference between the present work and the previously cited papers is that we retain the multi-hypothesis ability of the PF all the way up to the output of the system: where classic algorithms would make a choice, our filter outputs all likely hypotheses so as to avoid having to choose arbitrarily a hypothesis that is potentially wrong. If the filter has enough non-ambiguous information to provide a single hypothesis, then this hypothesis will be very reliable. But if ambiguities remain, the output will be the whole set of the hypotheses returned by the filter.

It is important to understand that the goal of this MM algorithm is not to determine a unique, although potentially wrong result, but to focus on the correctness of the matching even this implies multiple hypotheses as a result. The filter must

Algorithme 3.5 Initialization.

```
1: function INITIALIZATION(P_GNSS)
2:     if map not loaded then
3:         LOAD MAP(P_GNSS)
4:     end if
5:     GENERATE PARTICLES(P_GNSS)
6: end function
```

nonetheless restrain the size of the result set to a minimum in order to guarantee the best availability of the positioning system possible. The following sections describe the filter processing from initialization to the results output.

3.3.4 Initialization Step

The filter execution begins after the first GNSS fix is computed (see Algorithm 3.4 line 2). This event triggers the initialization of the particles, launching the MM on subsequent iterations (see Algorithm 3.5).

3.3.4.1 Map Loading

The first step is the map handling. SQL based map format are efficient when using large map coverage and database size due to the possibility to make spatial queries to a specific position. Although queries are generally fast to execute, query formatting and returned data parsing cause significant CPU load and thus response time. Their use should therefore be limited to punctual queries: using caching methods is then more efficient, as implemented by Bonnifait et al. [2009] where map information of the local area is loaded into memory as the car is moving from one position to another. Such method is mandatory in the case of large or extremely dense areas. The approach presented loads the whole map directly when initializing, as its size is relatively limited (4 km of road data representing around 1 MB). The SQL map is directly loaded into an internal C++ data structure (i.e. directly accessible), no SQL query is necessary to access the data anymore. C++ queries are of course still necessary but are far more efficient.

3.3.4.2 Spatial Initialization

Once the relevant map data loaded, the filter initializes its particles on the first valid GNSS position received (see Algorithm 3.5 line 5). This step is depicted in the red box on the top part the PF box in Fig. 3.5. Particles are generated around this position, to cover the full area corresponding to the associated HPL (see Fig. 3.7), circumventing any bias the GNSS fix could have (as the HPL denotes an area in which the true position is guaranteed to be, at a given risk level). Modern receivers begin to implement such calculation, as integrity becomes an important part of localization [Cosmen-Schortmann et al., 2008].

This step initializes the 2D coordinates (x^i, y^i) part of the particle state. The area to cover being circular, we chose to follow an arithmetic spiral pattern in order to spread uniformly the particles. For easy of calculation, a constant step initialization is used. The parameter describing the step between the turnings of the spiral has

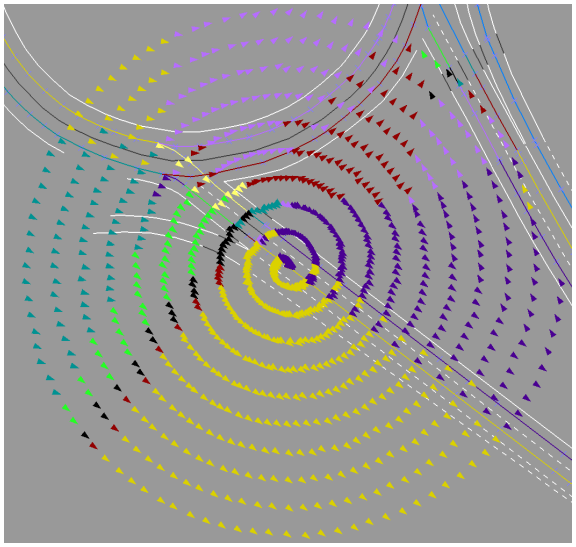


Figure 3.7 – Example of a particle initialization: colors denote the matched link. The initial heading corresponds to the matched link. Some particles are far from the links due to the 50 m HPL (this a high conservative value here), but will potentially be quickly eliminated during a resampling step.

to be set. This distance step impacts the angle step between two particles along the spiral. Choosing a suitable value is important to ensure a correct particle density for a good initialization of the filter, which depends greatly on N, defined as the number of particles. The coordinates of each particle can be simply computed as follow:

$$\begin{cases} x_0^i = & G_x + \frac{i}{n_{2\pi}} \Delta L \cdot \cos(2\pi \frac{i}{n_{2\pi}}) \\ y_0^i = & G_y + \frac{i}{n_{2\pi}} \Delta L \cdot \sin(2\pi \frac{i}{n_{2\pi}}) \end{cases} \quad i = 0, ..., N \tag{3.42}$$

with (G_x, G_y) the GNSS position, $n_{2\pi}$ the number of particles on each revolution, ΔL the distance between successive turnings. This formula is easy to apply, considering a constant angular step between the particles (from a polar coordinates point of view). The relation between two particles can then also be viewed as a rotation of the same angle (plus a change of module): it then suffices to compute a rotation matrix once and apply it for each particle. This reduces the trigonometric calculations to only the cosine and sine of the given angular step, rending the initialization highly efficient.

The drawback of this method is the decreasing particle density when the distance from the center increases, the phenomenon is particularly visible with large initialization diameter (see Fig. 3.8a). Moreover, ΔL needs to be manually set.

A solution to this problem is to consider a constant curvilinear distance d_{curv} between the particles. To have the most homogeneous set, ΔL is defined to be equal to d_{curv}. For ease of use, we work in the polar coordinate system centered on the GNSS position. A particle i has the coordinates (r^i, θ^i). An arithmetical spiral is defined by the equation:

$$r^i = b\theta^i \quad , \quad \forall \theta^i \in \mathbb{R}^+ \tag{3.43}$$

The angle of each particle has to be determined so as all particles are equidistant from each other. Let dL be the particle inter-distance, then the total length of the

spiral L_{tot} is:

$$L_{\text{tot}} = N \cdot dL \tag{3.44}$$

(3.43) also gives:

$$L_{\text{tot}} = \int_0^{\theta_{\text{tot}}} b \cdot \psi d\psi = b \cdot \frac{\theta_{\text{tot}}}{2} \tag{3.45}$$

θ_{tot} is the total revolution angle of the spiral (not modulo 2π). Combining (3.44) and (3.45), it can be isolated:

$$\begin{aligned} N \cdot dL &= b \cdot \frac{\theta_{\text{tot}}^2}{2} \\ \theta_{\text{tot}} &= \sqrt{\frac{2N \cdot dL}{b}} \end{aligned} \tag{3.46}$$

b is the coefficient controlling the spacing between the spiral turnings, this spacing is equal to:

$$sp = 2\pi b \tag{3.47}$$

We want $sp = dL$, therefore:

$$dL = 2\pi b \tag{3.48}$$

Incorporating in (3.46) gives:

$$\theta_{\text{tot}} = \sqrt{\frac{2Nb \cdot 2\pi}{b}} = 2\sqrt{N\pi} \tag{3.49}$$

The number of turns is given by:

$$n_{\text{turns}} = \frac{\theta_{\text{tot}}}{2\pi} = \sqrt{\frac{N}{\pi}} \tag{3.50}$$

The radius of the spiral corresponds to the HPL value, knowing the number of turns, dL can be determined:

$$dL = \frac{\text{HPL}}{n_{\text{turns}}} = \text{HPL}\sqrt{\frac{\pi}{N}} \tag{3.51}$$

Let us compute the angle coordinate of the particles by studying the angle step $d\theta$ between two elements. Integrating (3.44) between two successive points $P^{i-1} = (r^{i-1}, \theta^{i-1})$ and $P^i = (r^i, \theta^i)$ gives:

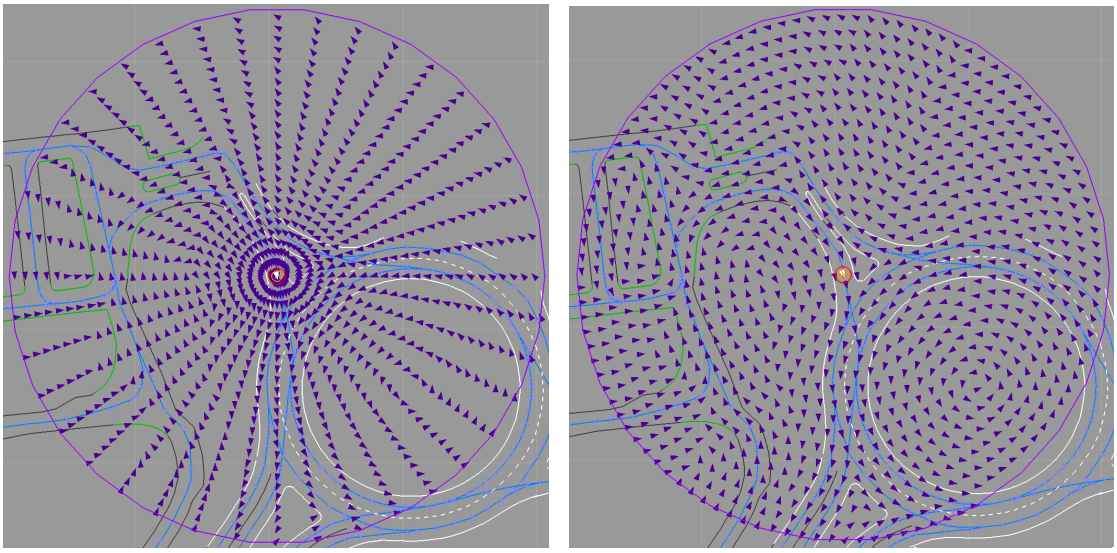
$$dL = \int_{\theta^{i-1}}^{\theta^i} b \cdot \psi d\psi = \frac{b}{2}\left(\theta^{i2} - \theta^{i-1^2}\right) \tag{3.52}$$

By only considering positive values for dL and θ_i:

$$\theta^i = \sqrt{\frac{2dL}{b} + \theta^{i-1^2}} \tag{3.53}$$

Using (3.48):

$$\theta^i = \sqrt{4\pi + \theta^{i-1^2}} \tag{3.54}$$



(a) (b)

Figure 3.8 – Comparison between the two initialization methods. While it may be acceptable for small diameter, the first procedure (Fig. 3.8a) quickly loses efficiency when the diameter grows. The particle distribution is much more homogeneous using the second spiral generation procedure (Fig. 3.8b), independently of the diameter.

The coordinates are thus given by the recurrence relation:

$$P^i = (r^i, \theta^i) \quad , \quad i = 0..N-1 \tag{3.55}$$

with

$$\begin{cases} \theta^0 & = 0 \\ \theta^i & = \sqrt{4\pi + \theta^{i-1}{}^2} \end{cases} \quad \text{and} \quad r^i = \frac{\text{HPL}}{2\sqrt{N\pi}} \cdot \theta^i$$

The Cartesian coordinates (the spiral being centered on the GNSS fix) are then given by:

$$\begin{cases} x^i = G_x + r^i \cos\theta^i & = G_x + \frac{\text{HPL}}{2\sqrt{N\pi}} \cdot \theta^i \cos\theta^i \\ y^i = G_y + r^i \sin\theta^i & = G_y + \frac{\text{HPL}}{2\sqrt{N\pi}} \cdot \theta^i \sin\theta^i \end{cases} \tag{3.56}$$

This method initializes the particles homogeneously in a circle (see Fig. 3.8b). Moreover, it does not require any parameter tuning: the only needed information is the number of particles (property of the filter) and the HPL (determined by the GNSS receiver). For these reasons, the homogeneous method is applied in practice during the developments of this thesis.

3.3.4.3 Matching and Heading Initialization

Each particle is then matched to the closest link, following a *point-to-curve* method which selects the link candidate with the lowest Euclidean distance to the particle. This process is responsible for most of the computational load during the initialization step as it needs to evaluate the distances of each particle to the links.

Concerning the vehicle heading, the assumption is made that the vehicle is driving on-road, and that this road is present in the map. It is thus reasonable to initialize the particles heading to correspond to their respective matched link's bearing. This process sets the values of the rest of the state (ψ^i, ml^i), finalizing the particle's initialization.

This initialization ensures that all the links present on the map around this position are considered, if enough particles are provided. This is the problem of particle density: the larger the initialization area, the more particles are needed in order to keep an acceptable particle density, essential to have an efficient PF. Some particles are clearly created outside the drivable area represented in the map. This illustrates the fact the particles are not strongly constrained onto the map and can evolve in the 2-dimensional space and not only on the centerlines. This provides a spatial flexibility to the filter, only limited by the decreasing likelihood of particles (for example one that gets too far away from a link). These particles will be quickly eliminated by the filter during the update steps, following the natural evolution of their likelihood.

3.3.5 Main Loop: Updating State and Matching

All the information needed for the matching to evolve is contained in the map structure, making the main loop filtering efficient for the purposes of MM. This section describes this second loop (see Algorithm 3.6), taking over after the filter has been initialized (see the bottom part of the PF box in Fig. 3.5).

Firstly, the 2D pose of each particle evolves using (3.57). It follows a unicycle evolution model, using as inputs the speed and yaw rate coming from the car proprioceptive sensors:

$$\begin{cases} x_k^i &= x_{k-1}^i + v_k^i \cdot \Delta t \cdot \cos \psi_{k-1}^i \\ y_k^i &= y_{k-1}^i + v_k^i \cdot \Delta t \cdot \sin \psi_{k-1}^i \\ \psi_k^i &= \psi_{k-1}^i + \omega_k^i \cdot \Delta t \end{cases} \tag{3.57}$$

This procedure corresponds to the state prediction step in Fig. 3.5. The sampling noise is added to the input, allowing each particle to evolve differently. $U_k^i = [v_k^i, \omega_k^i]^T$ is the input vector of the ith particle, with $v_k^i \sim \mathcal{N}(v_{\mathrm{raw}}, \sigma_v^2)$ and $\omega_k^i \sim \mathcal{N}(\omega_{\mathrm{raw}}, \sigma_\omega^2)$, based on raw measurements of the vehicle speed and yaw rate $(v_{\mathrm{raw}}, \omega_{\mathrm{raw}})$ with an added random noise. The noise, being added independently for each particle, allows them to spread during their evolution.

Secondly, the matching of each particle is evaluated to determine if an update is necessary. By design, the computations required during the matching update step are far fewer than during the initialization step. Further iterations do not require as many distance calculations: the matching evolves thanks to an efficient use of the map data, as explained next. The matched ID ml^i, being a discrete value, evolves differently, without a dynamic model per se. It follows the links logically using attributes stored in the lane-level map describing connectedness and adjacency, as described in the following. This evolution differs from the method used during initialization, being designed for efficiency (reaching real-time computation).

Algorithme 3.6 Main processing loop.

```
function MAIN LOOP(v, ψ)
    STATE PREDICTION(v, ψ)
    MATCHING UPDATE(Map)
end function
```

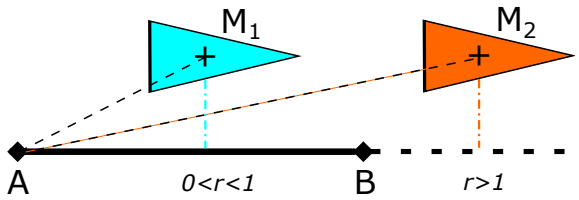


Figure 3.9 – Particle at time $t = 1$ (M_1) remains on the current segment $[AB]$, while at time $t = 2$ (M_2) it has left it. This is determined by calculating the ratio r described by (3.58).

3.3.5.1 Nominal Case

Let M_i be the position of a particle at time step i and $[AB]$ be the matched segment. M_i is considered to belong to $[AB]$ if its orthogonal projection falls on the segment. Thus, to determine if the matched segment needs to be updated, the ratio

$$r = (\overrightarrow{AB} \cdot \overrightarrow{AM})/||\overrightarrow{AB}||^2 \tag{3.58}$$

is computed: if $r > 1$ (respectively $r < 0$), the particle has left the current segment and has to be associated with the next one (respectively the previous one) by simply using the connectedness information of the map. This situation is illustrated in Fig. 3.9.

Based on this information, the ratio r becomes the only value requiring computation, which optimizes the execution of the main loop. Combined with the fact that map data are being stored in memory during the execution of the filter, the matching evolution is more efficient than a classic procedure that would require at each time step to search for the nearest link (resulting in N distance calculations, N being the number of links plus the cost of searching for the minimum). The chosen procedure reduces the need to a single distance calculation.

3.3.5.2 Case of Multiple Successors

Situations can arise where the current link has multiple successors. In this case, the filter follows a thorough exploration strategy: when leaving the current segment, a clone particle is created for each successor and matched accordingly (see Fig. 3.10). This effectively covers all hypotheses at the lane bifurcation. This differs from the common approach, choosing randomly a single successor given probabilities assigned to each candidate [Gustafsson et al., 2002; Szottka, 2013].

The cloned particles are exact copies of the original ones, and the evolution model together with the map data will allow a selection to be made, so that only the most likely hypotheses among the clones are retained. Note that this procedure increases the number of particles and can cause a computation complexity problem is if not controlled. A first preventive measure is to cap the maximum number of particles. For example, the maximum number of particles after cloning can be

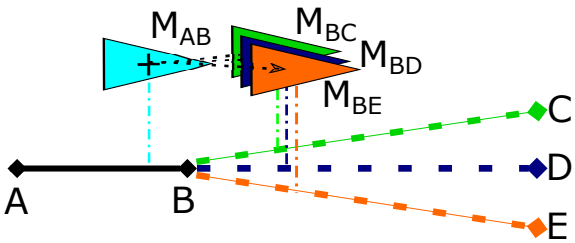


Figure 3.10 – Lane bifurcation where particle cloning occurs: the particle M_{AB} is duplicated into M_{BC}, M_{BE} and M_{BD} in order to do a thorough search.

limited to twice the initial size. This gives the possibility to clone every particle once simultaneously. The particles rarely need to be all cloned at the same time and one can then be clone more than once if needed (e.g., in a 4-way intersection). Then, the number of particles will be regulated by the normal course of the filter during the resampling step (see Section 3.3.7), in order to fall back to the initial number of particles.

3.3.5.3 Lateral Lane Change

Another possible evolution is a change to an adjacent lane. This is made possible as the particles are not constrained to evolve only on the links but also laterally. Using lane marking information from the map (see Section 2.4), the lane width can be determined. When a particle leaves the boundaries of its matched link, it will be transferred to the adjacent link, if one exists. Note that cloning the particle, as for the multiple successors case, is not a viable option because of the continuous aspect of the lateral lane change (as opposed to the punctual link change in the previous case). The worst case scenario would be a particle "swinging" between the right and left lane: this oscillation would create each time a clone and would unbalance the particle set. A simple matching change is thus preferred in this case.

3.3.6 Likelihood Calculation

After the particles evolution, the final step in the PF is the likelihood update and resampling step (see Algorithm 3.7). These are necessary to keep a consistent particle set. The likelihood of a particle is expressed through its weight: the higher the weight, the more likely the hypothesis. It is updated after each iteration of the filter. The MM algorithm is kept as independent from the GNSS as possible, the likelihood computation is then done by comparing the particle state to the prior map information. In this way, the particle evolution can be seen as a *map-aided dead-reckoning*.

3.3.6.1 Map Likelihood

A Sequential Importance Resampling (SIR) PF (see Section 3.2.4.5) is used combined with a recursive weight calculation such that:

$$w_k^i = w_{k-1}^i \cdot p(z_k|X_k^i) \tag{3.59}$$

where z_t represents the measurements used to assess the likelihood of the hypothesis, which corresponds to two metrics computed using the map. Note that every particle

Algorithme 3.7 Likelihood processing.

```
function LIKELIHOOD PROCESSING(P_GNSS, L_cam)
    if no marking detected then
        MAP LIKELIHOOD(Map)                          ▷ see Section 3.3.6.1
    else
        MARKINGS LIKELIHOOD(v, ψ)                    ▷ see Section 3.3.6.2
    end if
    GNSS GATING(P_GNSS)                              ▷ see Section 3.3.6.3
end function
```

uses the same z_t.

The first metric is the heading difference ($\Delta\psi^i$) between the particle and the link, as used by Merriaux et al. [2015]. This metric is a good likelihood indicator for map-matching, allowing the links whose heading evolution does not correspond to the dead-reckoned estimate to be eliminated quickly. A Gaussian distribution is assumed such as

$$f_\psi(\Delta\psi^i) \sim \mathcal{N}(0, \sigma_\psi) \tag{3.60}$$

to represent the particle likelihood with respect to its heading. It is centered on 0 (maximum likelihood if the particle heading is the same as its link) with a typical standard deviation of $\sigma_\psi = 15°$ (value set experimentally during the tuning stage of the method). Considering larger angles allows to take into account possible maneuvers (e.g., a positioning readjustment within the lane, or the initiation of a lane changing). Fouque et al. [2008] have modulated the standard deviation of the heading in function of the speed of the car to model this behavior (the higher the speed, the lower σ_ψ). Note that this heading metric is not sufficient to clear all ambiguities. For instance it cannot discriminate parallel links such as two lanes of the same road.

The second metric is the orthogonal distance ($d^\perp$) to the centerline: this takes into account that the farther a particle is from the roadway, the less likely it is. To reflect the fact that a car is equally likely to be located in any lateral position on the roadway, the likelihood function is defined as

$$f_d(d^\perp) = \begin{cases} 1 & \text{if } |d^\perp| < L \\ \max\{1 - \frac{|d^\perp| - L}{m}, 0\} & \text{otherwise} \end{cases} \tag{3.61}$$

where L is half the width of a driving lane, and m is a margin on either side of the lane where the likelihood decreases linearly, forming a trapezoidal distribution (see Fig. 3.11). We chose this shape in order to have efficient computation in real-time. When two lanes are adjacent, the two functions overlap so that there is no discontinuity when a particle changes lane: the top part of the adjacent lane's trapezoid begins where the current lane's trapezoid ends. In other words, the sloping parts only apply when there is no other lane adjacent to the current lane, because otherwise the particle would change its matching and apply the likelihood function of the new lane.

Given that the two metrics are considered conditionally independent, (3.59) can be computed as

$$w_k^i = w_{k-1}^i \cdot f_\psi(\Delta\psi^i) \cdot f_d(d^\perp) \tag{3.62}$$

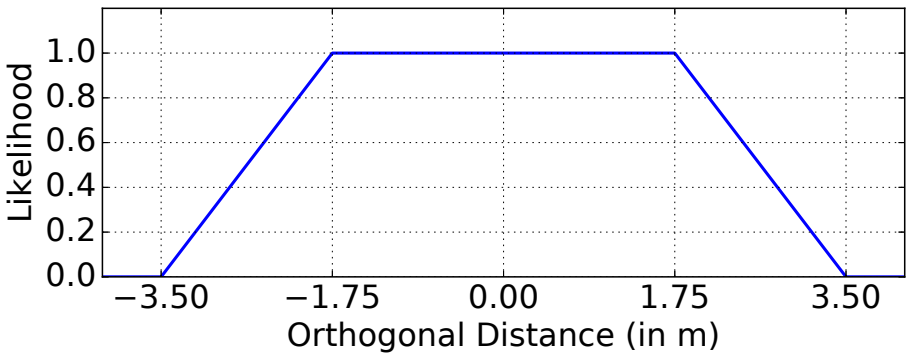


Figure 3.11 – Trapezoidal likelihood function for orthogonal distance (based on the French regulations relating to the width of roads). The center of car is equally likely to be in any lateral position on the roadway in the interval [-1.75; 1.75].

3.3.6.2 Refining Using Exteroceptive Sensors

The previous calculations take only into account the map data. Additional information can be used in order to refine the likelihood calculation. Information from exteroceptive sensors can be used to this end, by providing for example the relative heading of the vehicle with respect to the lane markings. In this study, we use a smart camera system able to detect the ego-lane markings and return them using polynomial models. For each detected lane marking, the camera returns the coefficients of a third-degree polynomial [Tao and Bonnifait, 2016] (see (3.63)) approximating the equation of a clothoid, where x and y are coordinates in the camera's working frame $(C, \overrightarrow{x_c}, \overrightarrow{y_c})$ (see Fig. 3.12).

$$y = \frac{C_3}{6} \cdot x^3 + \frac{C_2}{2} \cdot x^2 + C_1 \cdot x + C_0 \tag{3.63}$$

In practice, the highest coefficients (the curvature C_2 and curvature derivative C_3) are often very noisy and do not provide reliable information. Therefore, only C_0 (the lateral offset, in meters) and C_1 (the line heading, in radians) are used, the lane marking being considered as a straight segment. This approximation still gives a good estimate of the line position and orientation with respect to the car. Note that the algorithm would also work with other sources of information (e.g., lidar-based lane detection) that are able to provide lane detection in a similar manner [Joshi and James, 2015]. Road borders detected by a lidar can also be used as long as they are stored in the HD-map [Jabbour et al., 2008].

As presented in Section 2.4, lane markings are additional information included in the map. They follow the same geometric representation as the centerlines (i.e. polylines). Each driving lane directly references the lane markings (if they exist) that delimit it, and it is therefore possible to obtain the equation $y = C_1 \cdot x + C_0$ for the current left and right markings.

Using a camera as source of information to improve the localization performance is common practice. For example, Tao et al. [2013] use the lane detection measurements from a camera with map data (a preliminary mapping of lane markings must be performed) in a loosely-coupled solution to observe GNSS errors. With this method, only the lateral positioning (with respect to the road) error is observable. To observe the complete positioning error, a change of direction must occur. Data fusion (for instance, using a Kalman filter) with odometry is then able to reduce

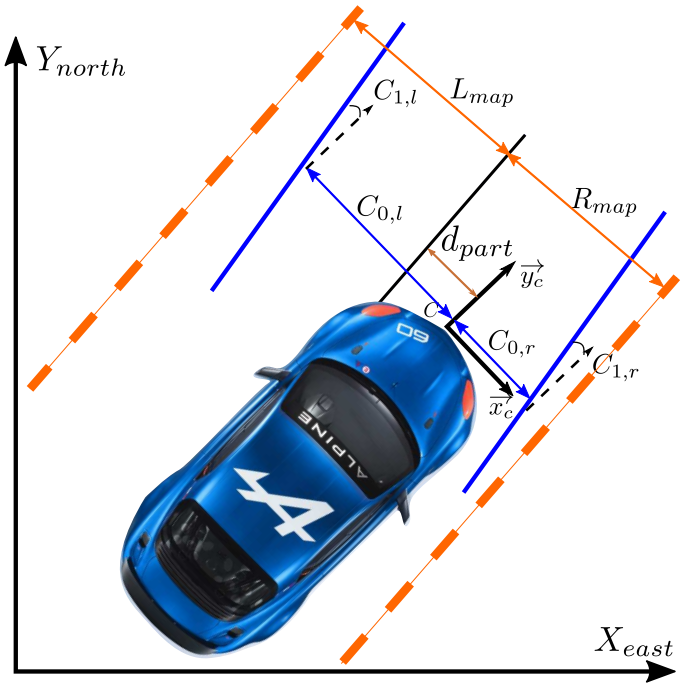


Figure 3.12 – Reference frames used by the camera for describing lane markings. Dashed orange lines represent lane marking data from the map. Camera detections are represented in blue. The lane centerline is in black. Please note that $\overrightarrow{y_c}$ points forwards.

the positioning error (down to 30 cm laterally for a low-cost u-blox 4T receiver), by integrating the difference between the measurement made by the camera and the expected value, given the GNSS position and map information. This shows that prior map information is a key component for localization. Note that the authors only use a single lane detection at a given time.

Other methods choose a Simultaneous Localization and Mapping (SLAM) [Durrant-Whyte and Bailey, 2006] approach: building their own map while at the same time localizing themselves. The position refinement emerges from back-propagating the error correction computed from a loop closing event, that can be based on the lane markings mapping [Hata and Wolf, 2014]. In the present work, a map is already available. The lane markings can be processed directly, assuming that prior information exists.

This exteroceptive information, if available, is thus used during the update step of the filter to calculate the likelihood of each particle based on richer information [Li et al., 2017a]. Note that the lane markings are not always detected by the camera. This can be caused by difficult conditions for the sensor, for instance, bad lighting, a wet road surface, faded markings, etc. The camera also performs best when moving in a straight line, since the marking tracking can be challenging in high-curvature roads [Kim, 2008]. In these cases, the camera indicates a low confidence level and the likelihood calculation reverts to the map-only likelihood described previously.

If marking detection is available, lateral positioning in the lane can be taken into account more accurately than previously. The likelihood calculation follows the same procedure as before, with a slight modification: with marking information, the likelihood will be centered on the relative lateral positioning according to the

detection [Rabe et al., 2017]. To do this, two position ratios are computed, describing the lateral position of the vehicle in its lane. A particle within the limits of the lane markings will have a ratio between 0 (when on the left-hand marking) and 1 (on the right-hand marking).

The first position ratio uses the marking detection from the camera, i.e. the signed distances $C_{0,l}$ and $C_{0,r}$ (respectively the C_0 coefficients of the left and the right lane marking detection, respectively with negative and positive values, see Fig. 3.12) and is defined as:

$$r_{\text{cam}}^{\text{lat}} = \frac{C_{0,l}}{C_{0,l} - C_{0,r}} \tag{3.64}$$

The second position ratio corresponds to the map data: for each particle (i.e. position hypothesis) the theoretical ratio is computed using the distances L_{map} and R_{map} of the left and right lane markings to the centerline (both unsigned), and the signed distance d_{part} of the particle to the centerline (positive to the right):

$$r_{\text{map}}^{\text{lat}} = \frac{L_{\text{map}} + d_{\text{part}}}{L_{\text{map}} + R_{\text{map}}} \tag{3.65}$$

Absolute lane marking distances are not used so as to address a possible scaling factor discrepancy between the lane detection provided by the camera and the map information. In other words, if an incorrect lane width is stored in the map or if the camera does not estimate well the width of the lane, the ratios are still valid. Moreover, it aggregates the two measurements into a unique metric that is more meaningful than, for example, the sum of the both detections, especially in the previously described situation.

A drawback of this method is that it imposes the detection of both right and left markings to be able to calculate the ratio. This increases the quality requirements, as the conditions must be good enough for both markings to be detectable, eliminating unclear situations prone to errors.

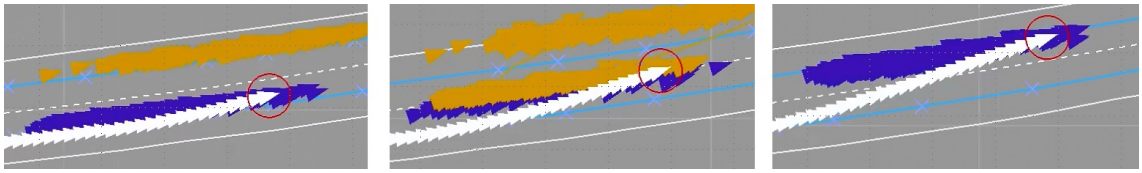
The physical meaning of these ratios is a percentage of the driving lane width. For example, a ratio of 0.1 is 10% of the width of a lane, which might for instance correspond to a distance of 0.35 m on a lane that is 3.5 m wide.

If the camera observations are in accordance with the particle position on the map, the two ratios will be approximately the same. To verify this, the difference between them is used to compute the likelihood:

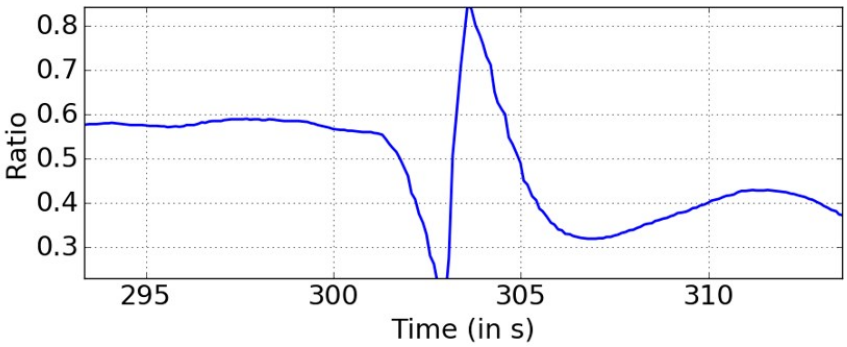
$$\Delta r = r_{\text{map}}^{\text{lat}} - r_{\text{cam}}^{\text{lat}} \tag{3.66}$$

The closer to zero the value is, the better the particle fits the camera observation. The likelihood function is thus set as a Gaussian distribution centered on 0 (i.e. a null difference meaning $r_{\text{map}}^{\text{lat}} = r_{\text{cam}}^{\text{lat}}$, a perfect fit), described by $\mathcal{N}(0, \sigma_{\text{dist}})$, with $\sigma_{\text{dist}} = 0.1$. This difference in ratio would correspond to a 10% error (0.35 m on a 3.5 m lane). The value has been set empirically, as the camera does not provide any quantitative indicator of accuracy.

In the same way, instead of considering the car being aligned with the road segment, the heading coefficients from the camera $C_{1,r}$ and $C_{1,l}$ are used to calculate the particle heading likelihood: the mean heading C_1 is computed and serves as a reference for the particle heading. The heading likelihood is described by $\mathcal{N}(0, \sigma_{\text{head}})$ with $\sigma_{\text{head}} = 15°$, set empirically.



(a) Three key moments of the lane change depicted chronologically from left the right. The blue and orange colors denote the lane assignment of the particles, the blue one denotes the actual lane in which the car is currently driving.



(b) The ratio evolution allows the detection of such event and indicates a change to the left.

Figure 3.13 – A lane change situation: the left lane particles get out of the road and are therefore eliminated by the filter, leaving only the originally on the right-side particles viable: a unique hypothesis is finally determined after the lane change.

With exteroceptive lateral information, the likelihood function follows the lateral movement of the vehicle, therefore containing the particle spread in this direction. This gives a much better lateral distribution (tighter) and even naturally allows lane changing maneuvers. Moreover, it allows a better description of the car's path in the carriageway. For instance, if the vehicle is one meter away from the center of the lane, the particles fitting this behavior will have the best likelihood score.

Consequently, maneuvers such as lane changes can be detected (see Fig. 3.13): a characteristic discontinuity of the camera ratio is observable (see Fig. 3.13b). For a left-hand lane change, the ratio tends toward 0 then jumps to 1 to finally take a value in accordance to the car position in the new lane. For a right-hand change, the opposite change is observed, with a ratio tending toward 1 and jumping to 0.

3.3.6.3 GNSS Gating

This last step of the likelihood calculation ensures that the filter stays consistent by killing particles that stray too far from the position returned by the GNSS receiver. It simply sets the weight of these diverging particles to 0 in order to eliminate them from the filter during a resampling step. To keep the influence of the GNSS fix to a minimum, the gate radius (after which a particle is eliminated) is set to the HPL returned by the GNSS receiver. Typically, the HPL can reach a value of 50 m for a 10^{-4} integrity risk in urban environments [Cosmen-Schortmann et al., 2008].

The situation where all the particles are eliminated by the gating is the worst case scenario. In this rare occurrence, the filter diverged as all particles left the HPL and resetting must be performed. The current particle set is discarded, as not being relevant anymore and the initialization procedure described in Section 3.3.4

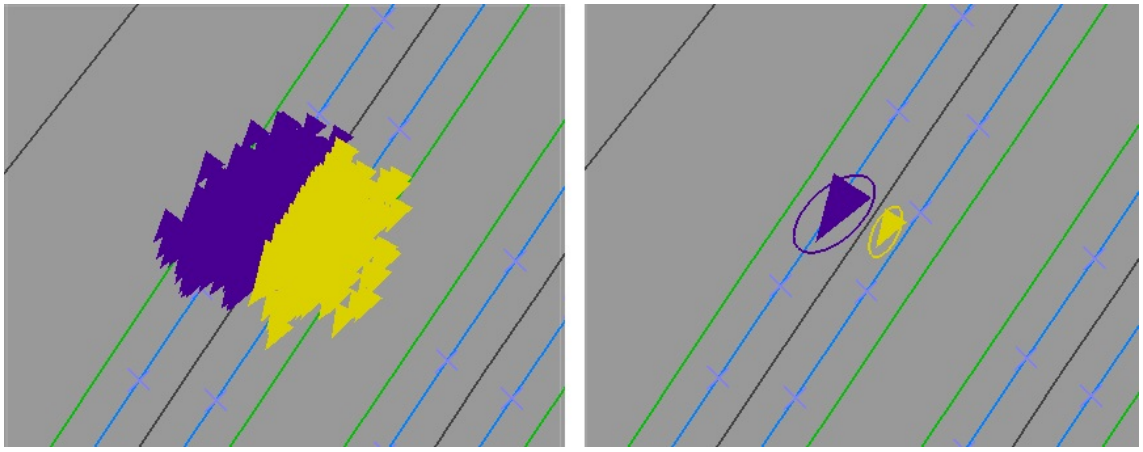


Figure 3.14 – The set of particles (on the left) is aggregated into hypotheses corresponding to the mean position (on the right, computed by (3.67)). The size of the hypothesis is proportional to its weight (computed by (3.68)).

is renewed, creating a particle set adapted to the current position and HPL.

3.3.7 Resampling

Finally, the resampling procedure presented in Section 3.2.4.5 is applied. It follows the low variance resampling, using an adaptive resampling with a 2/3 threshold, as described by Algorithm 3.2.

3.3.8 Hypotheses Determination

Determining the hypotheses is done by clustering the particles by matching element. For instance, in this research the level chosen is the link, representing a drivable lane. The estimated position of each hypothesis is computed using a weighted average of all the particles belonging to the corresponding lane:

$$X_{\mathrm{Hyp}_j} = \sum_{i \in P_j} w^i X^i \tag{3.67}$$

with P_j, the set containing the indexes of the particles belonging to the jth hypothesis.

The sum of the particle weights, computed by (3.68), estimates the relative likelihood of each hypothesis among the solution set and can be used to measure the confidence the filter gives to the given hypothesis.

$$w_{\mathrm{Hyp}_j} = \sum_{i \in P_j} w^i \tag{3.68}$$

Fig. 3.14 illustrates this computation in the case of two hypotheses: each color represents a different hypothesis. They are considered separately and the mean particle of each subset is computed using (3.67).

3.4 Experimental Results and Evaluation

3.4.1 Experimental Setup

This section describes briefly the experimental setup used to validate the algorithm. For a more in-depth description of the experimental setup, please refer to Section 2.6, and Appendix B for the datasets. A C++ implementation of the filter has been realized, running in real-time on a mid-range laptop (AMD A8-4500M CPU@1.90 GHz with 8 GB RAM). The Pacpus framework provides an interface to the car's sensors, allowing access in real-time, as well a replay "on desk" of the data. This allows easy validation of the algorithm by generating results, and notably by studying the repeatability of the filter.

3.4.1.1 Environmental Conditions

Two datasets have been used to study the performance of the algorithm.

1. **Open sky conditions** (see Fig. 3.15a): the first one follows a 2 km trajectory in open sky conditions. The car drives exclusively on a bypass with 2 one-way lanes. A roundabout is used to make a u-turn. Moreover, lane markings are present on a majority of the bypass.

2. **Peri-urban conditions** (see Fig. 3.15b): the second one (4 km-long) extends the first by adding travel in an moderately dense urban environment, i.e. areas prone to multipath and even GNSS signal loss.

3.4.1.2 System Setup

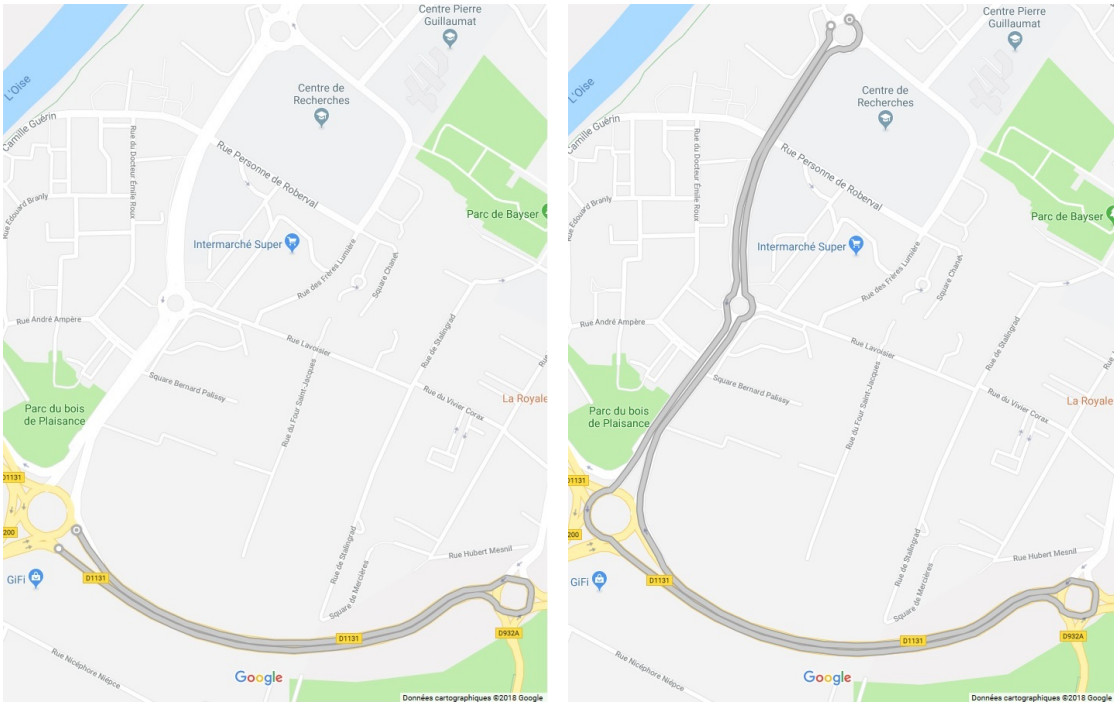
A C++ implementation of the algorithm was developed using the Pacpus framework[1], providing easy integration with the Heudiasyc Laboratory's test vehicles for data acquisition and offline replay. The experimental vehicle IRIS (see Section 2.6.1.1) was used for the acquisition of real road data. The car was equipped with a Septentrio AsteRx2 GNSS receiver (used as a standalone receiver with no differential corrections) and DR information was accessible directly from the vehicle CAN bus, as well as from a Mobileye EyeQ2 system used for its lane detection capabilities.

3.4.1.3 Filter Parameters

All the necessary parameters for the filter's execution are listed is Table 3.1. The following gives details on the determination of their values.

As the filter's results improve with size of the particle set, maximizing the number of particles (`NB_PARTICLES`) is important. It depends essentially on the processing power available: to do so, a benchmark is necessary to test the processing load with different quantities of particles. To respect real-time execution, the limiting factor is mostly related to the processor characteristics (i.e. number of cores and clock speed). The highest number of particle allowing a smooth execution is chosen.

[1] developed at Heudiasyc. More information at `https://pacpus.hds.utc.fr`



(a) (b)

Figure 3.15 – The two test travels used to study the filter's performance in gray. The first (Fig. 3.15a) has open sky conditions during all the sequence while the second (Fig. 3.15b) is a more varied scenario, crossing a peri-urban area (images from © 2018 Google Maps).

Table 3.1 – Typical values for the filter's parameters

Name	Description	Typical Value
NB_PARTICLES	Number of particles	$[500, 2000]$
CLONING_FACTOR	Max cloning factor	2
MEAN_SPEED	Speed sampling noise mean	$[1, 2]$ km/h
STD_SPEED	Speed sampling noise std	$[5, 10]$ km/h
MEAN_YAW_RATE	Yaw rate sampling noise mean	0.155 deg/s
STD_YAW_RATE	Yaw rate sampling noise std	5 deg/s
OFFSET_GNSS_COV	Offset GNSS covariance	1 m
HPL	Protection Level	$[5, 15]$ m
RESAMPLE_THRESH	Resampling threshold	0.66
SIGMA_HEAD	Weight sigma heading	15 deg
SIGMA_DIST	Weight trapezoidal margin	1.75 m

The CLONING_FACTOR, determining the maximum number of particles after cloning (NB_PARTICLES·CLONING_FACTOR) is also taken into account in these benchmarks.

The input noises characteristics (MEAN/STD_SPEED, MEAN/STD_YAW_RATE) are crucial parameters. PF performance is tied to a correct setting of the sampling noise. It must be based on the error model of the actual inputs, to which is added a random noise. The error models have been determined statistically using data recording from the sensors on a reference trajectory. The additional noise is chosen empirically to spread the particles during their evolution. Note that a too large value can cause the filter to diverge.

During this thesis, the HPL value was not available from the GNSS receivers during the dataset recordings. To replace such sensor information, a value has been calculated in post-processing, yielding a correct HPL: a constant value is applied for each dataset, corresponding to the largest GNSS error observed during the travel. Doing so yields a correct HPL in the sense of integrity, but it is sometimes over-evaluated: for instance, during a travel mixing open sky conditions and urban environment, the considered HPL will be the urban one, over-estimating it for the open sky conditions. With an actual HPL computed from a GNSS receiver (e.g., KIPL calculation, see Section 2.3.2.2), the filter is expected to perform better in mixed travels, using potentially smaller HPL in good reception conditions.

As noted in Section 3.2.4.5, the resampling threshold is a common value taken from the literature. The last parameters characterize the likelihood calculation: SIGMA_HEAD is the standard deviation used in the heading difference likelihood function. This value must be set to a permissive value, in order to allow events such as lane changes. SIGMA_DIST corresponds to the likelihood linked to the distance to the centerline: the typical value is the half width of a driving lane.

3.4.1.4 Map-Matching Ground Truth

To assess the performance of the Simultaneous Map-Matching and Localization Using Particle Filtering (SMMAL-PF), reference map-matching data must be available. This matching ground truth is not directly available from any sensor. To avoid using another matching algorithm with the related risk of incorrect matching, the map-matching ground truth has been determined manually during this thesis. This has

Figure 3.16 – Image from the contextual camera (a front-facing dashboard camera) showing clearly the position of the car on the roadway. It is used to determine manually the map-matching ground truth.

been made possible by the use of contextual camera filming the road scene during the dataset recording (see Fig. 3.16).

As the images are synchronized with the sensor data, labeling each recording step is done accurately with ease. This procedure is applicable as the studied test travels are relatively simple and short. The resulting map-matching ground truth is a table containing the link ID corresponding to the actual position of the vehicle for each time step of the travel. It will be used to assess the correctness of the SMMAL-PF during these tests.

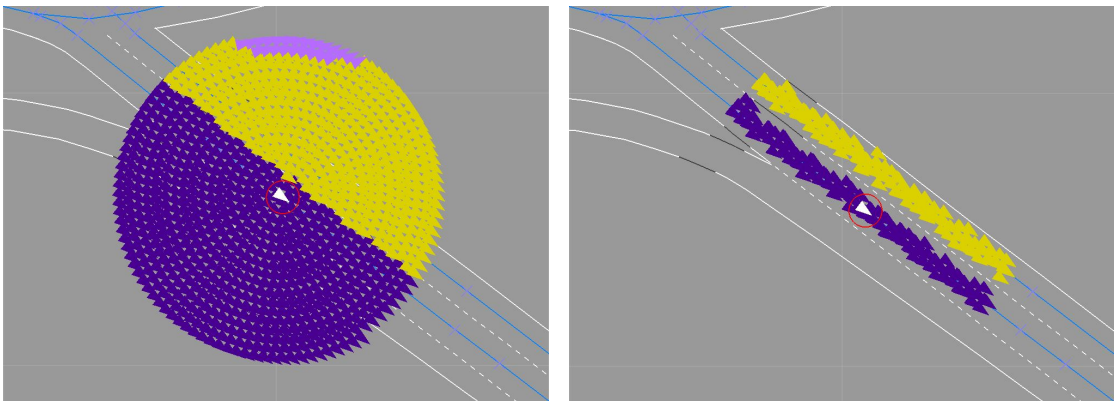
3.4.2 System Initialization and Study of the PF Convergence

This first study uses the open sky dataset to assess the filter's initialization in nominal conditions (i.e. without GNSS perturbations). The PF contains 1000 particles at initialization. This number gives a good compromise between performance and particle density.

This first part studies the filter around initialization time. The initial GNSS fix is made at the entrance of the bypass, around 20 m from the exit of a roundabout (see Fig. 3.17). This is a simple situation where the particles get approximately evenly spread between the two likely hypotheses, as shown in Fig. 3.17a. A small portion is matched to the near round about (in light purple on the top of the particle set) but the majority of the particles belongs to the left or right lane of the bypass (in yellow and dark purple).

The initial situation, in which the particles fill an area corresponding to the HPL, quickly evolves after computing the likelihood of the particle set. As shown in Fig. 3.17b, the map likelihood eliminated all off-road particles. Lane detection is available which provides an even greater lateral confidence. In this case, the filter converges immediately to the two likely hypotheses.

While the situation is favorable sensor-wise, the travel path is quite linear and shows one of the main ability of the PF: the filter bears a large uncertainty longitudinally (i.e. in the road axis) as no observation is made to characterize the likelihood in this direction. This is clearly shown in Fig. 3.17b as the particle sets are elongated along the road. This characteristic is common to any Bayesian filtering method that



(a) (b)

Figure 3.17 – The filter initializes the particles to fill the area corresponding to the HPL (15 m in Fig. 3.17a). The particles off-road are immediately eliminated by the likelihood calculation (see Fig. 3.17b).

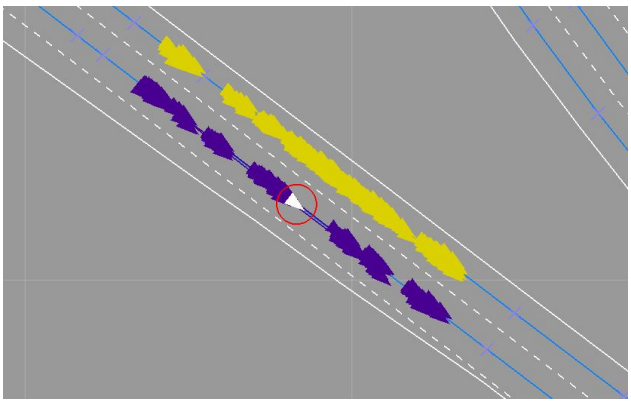


Figure 3.18 – Clusters of particles forming after 7 s of execution. These are due to i) the absence of longitudinal information and ii) a too small a number of particles.

cannot observe errors in a given direction.

At first, the correct hypotheses are identified and the particle distribution seems coherent. But after several iterations, the particle sets begin to create longitudinal clusters (see Fig. 3.18), giving multiple possible positions by hypothesis and bringing even more complexity to the situation. A solution would be using a higher number of particles that could cover more continuously the hypothesis, but this comes at a higher computational cost.

First, the influence of the initial position on this phenomenon is studied: the HPL is fixed to 15 m and the only varying element is the initial position. This is done by starting the algorithm at a posterior date, for instance every 14 seconds at $t_0 + \{0, 14, 28, 42, 56, 70\}\, s$, yielding the starting positions depicted as circles in Fig. 3.19.

Fig. 3.20 shows the evolution of the number of clusters over time for the given initialization times. For more readability, the data only cover the clusters for a single hypothesis. The convergence (defined as the moment when the particles corresponding to a hypothesis form a unique cluster) is thus reached when the number of clusters falls to 1.

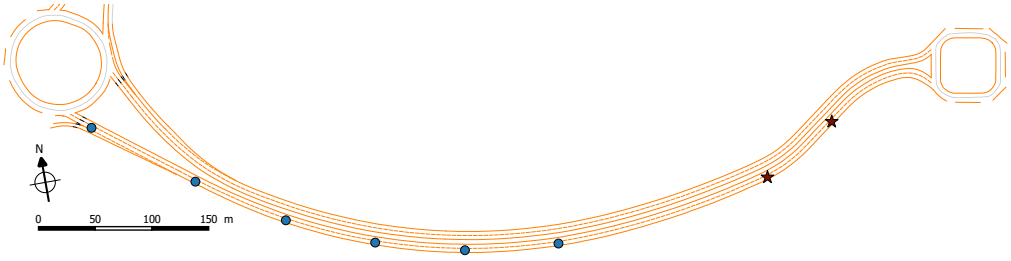
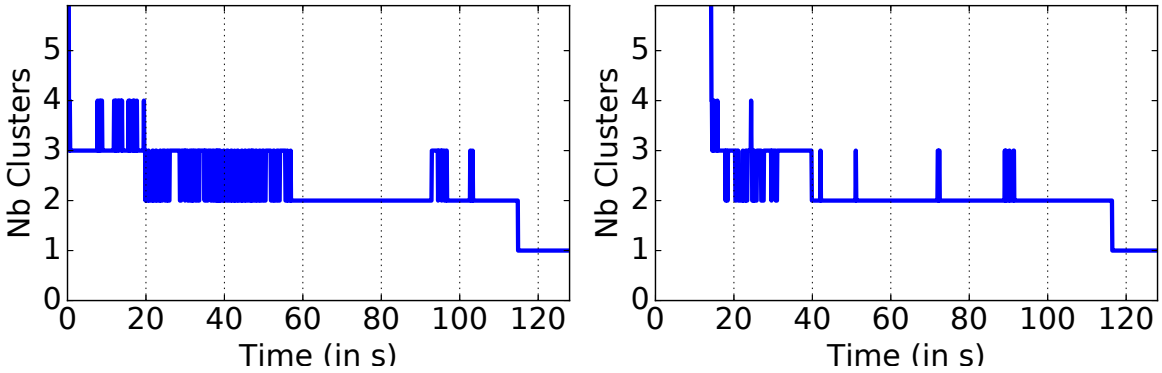
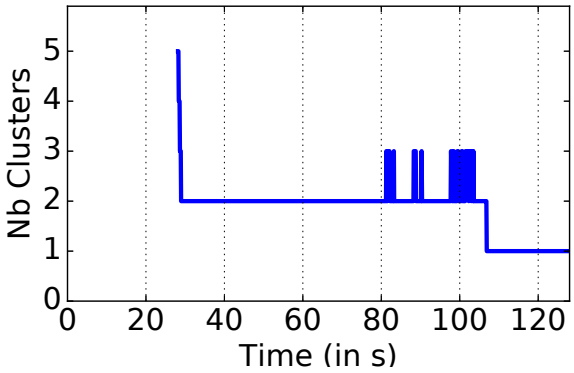


Figure 3.19 – Starting positions (blue circles) used for the initialization position study. The brown stars correspond to the time $t = 106\,s$ and $t = 119\,s$.

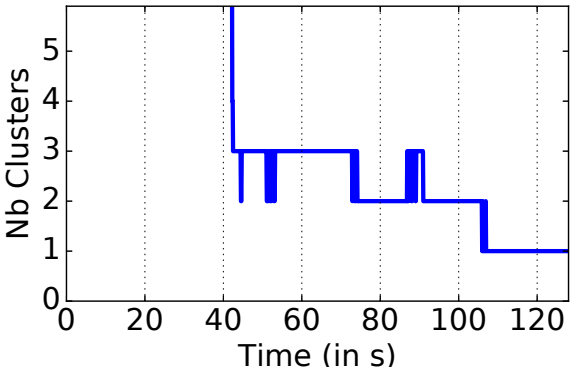


(a) Start time: t_0

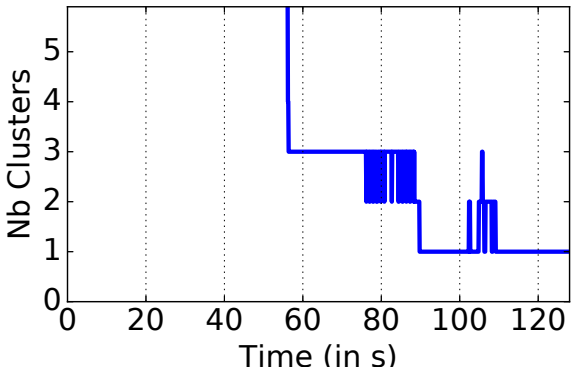
(b) Start time: $t_0 + 14\,s$



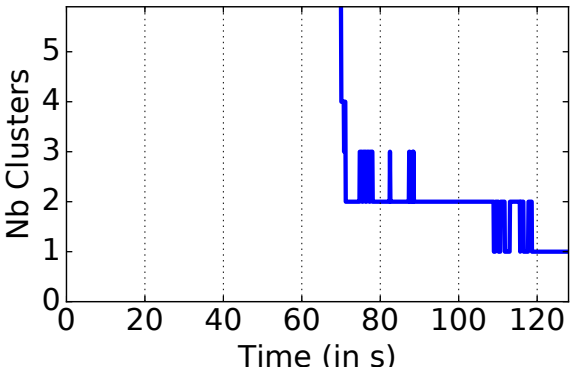
(c) Start time: $t_0 + 28\,s$



(d) Start time: $t_0 + 42\,s$



(e) Start time: $t_0 + 56\,s$



(f) Start time: $t_0 + 70\,s$

Figure 3.20 – Cluster formation study for different start times (i.e. different start position). HPL=15 m. In all instances, the number of clusters reach 1 around the same time (110 s): cluster elimination is independent from the start time.

For all starting positions, the filter converges in the same time frame, with $t \in [106, 119]\, s$ (the positions corresponding to these times are represented by brown stars in Fig. 3.19). Note that in Fig. 3.20e, the number of clusters falls a first time to 1 but this convergence is not stable as new clusters form thereafter. The convergence time taken into account is the one when the cluster remains stable.

This time frame corresponds to a more significant bend in the travel. It shows the need for changes of direction in order to be able to observe errors in all direction [Tao et al., 2013]. The starting time is therefore not related to the clustering phenomenon (for a given travel).

The following studies the influence of the initialization radius (i.e. HPL) on the convergence time.

Fig. 3.21 shows the convergence time of the filter for the same travel, with the same starting time but different HPL values. For a value greater than 15 m, the behavior is the same as previously described, with a convergence only happening at a change of direction. With a 10 m HPL (see Fig. 3.21b), the clustering is more contained, converging earlier at $t = 60\, s$. A unique cluster is obtained as soon as 5 s but is not stable. To obtain quickly a single cluster, the HPL has to be set to 5 m: the filter converges immediately and stably.

In most cases, an HPL of 5 m is obtainable in open sky conditions: this value is thus realistic. The influence of the HPL on the convergence of the filter has been exposed. It is important to not overestimate the value of the HPL and bringing the risk to hinder the convergence of the filter. It is the perfect illustration of the trade-off between integrity and availability: the system must not be too pessimistic so that it becomes unavailable too frequently.

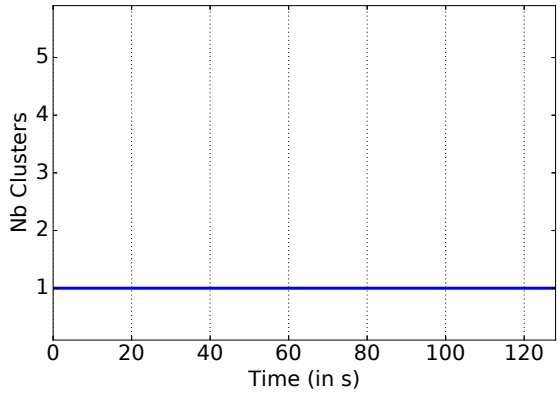
3.4.3 System Performance in Tracking Mode

This section studies the filter's behavior after convergence. Therefore, an HPL small enough to allow a quick convergence is chosen.

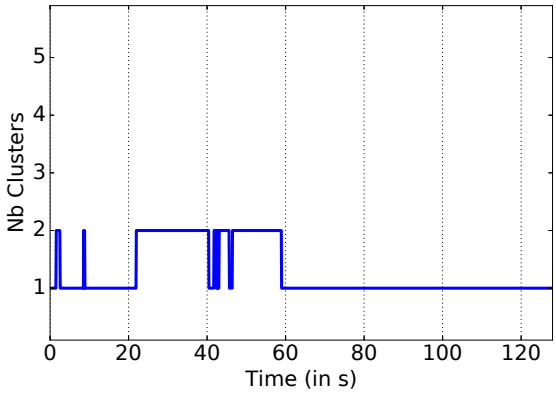
3.4.3.1 Nominal Conditions

In open sky conditions (first dataset, see Fig. 3.15a), the HPL is set to 5 m as seen previously. The tracking is straightforward as the car is traveling on the 2-lane bypass. The expected result is to get two hypotheses tracked (left and right lanes), which is the observed results depicted in Fig. 3.22. The blue line corresponds to the right-hand side lane and the red to the left-hand side. Two others hypotheses appear during the travel, as developed later. The correct hypothesis is the blue one in $t = [0, 275]$ s and the red one from $t = 275$ s till the end of the travel.

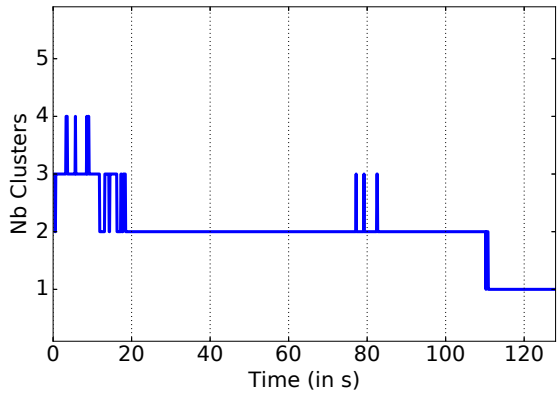
Two events of interest happen during this travel. The first one is the roundabout happening in the middle of the travel. Fig. 3.23a focuses on this event: the car enters the roundabout at $t = 125$ s and exits at $t = 165$ s. The car follows the outer lane that has a distinct profile (a sharper initial turn with longer distance traveled), the filter thus gives more weight to the right-side hypothesis (the blue curve rises) which matches the closest to the odometry data. This roundabout being almost square, the car takes another sharp turn. Due to the uncertainty on the yaw rate, some particles are directed to the inner lane, transferring weight to the other hypothesis: the red curve value increases. But in the end, factoring the longer distance covered, the most likely hypothesis emerges, being on the right-hand lane: the blue hypothesis



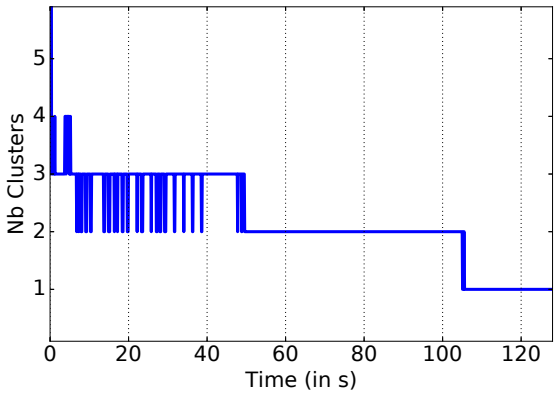
(a) HPL=5 m



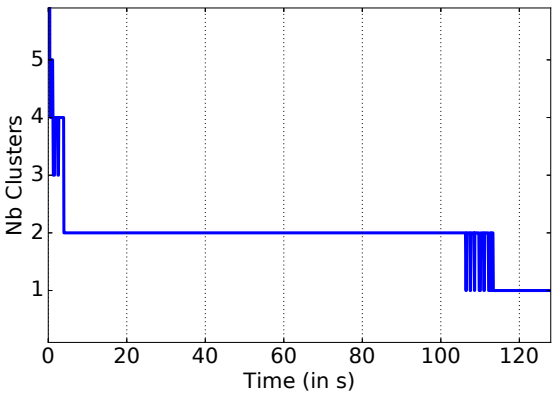
(b) HPL=10 m



(c) HPL=15 m



(d) HPL=20 m



(e) HPL=25 m

Figure 3.21 – Study of HPL influence on clustering.

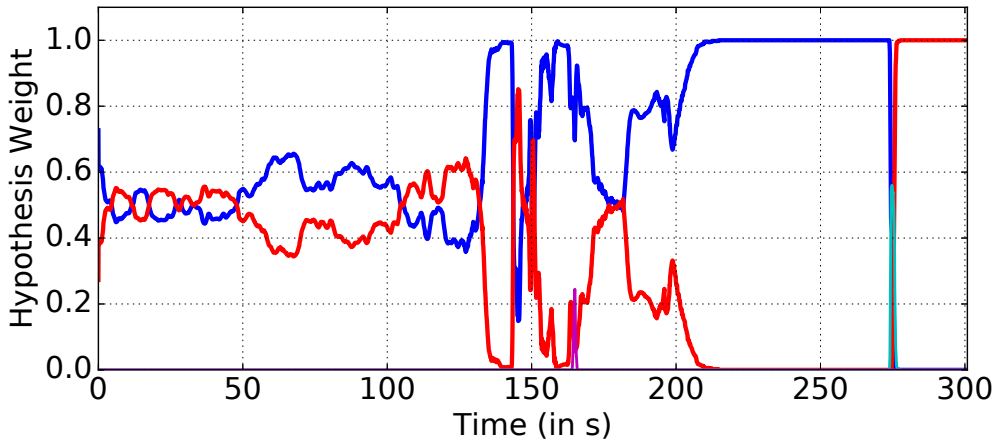


Figure 3.22 – Evolution of the hypothesis weights on the open sky dataset. Two majors hypotheses are prominent.The strong variations happening at $t = [140, 175]$ s are caused by a roundabout. The correct hypothesis is the blue one at $t = [0, 275]$ s and the red one from $t = 275$ s till the end of the travel.

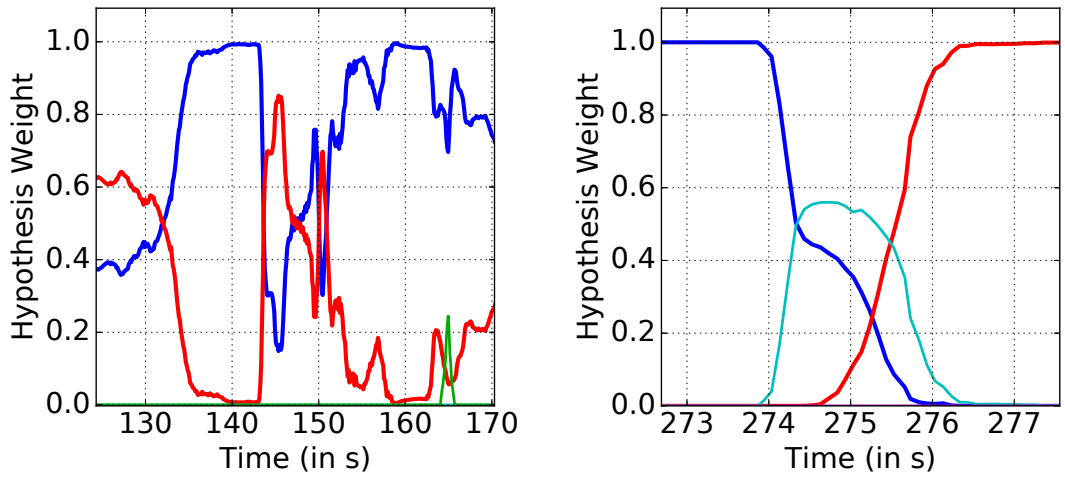
takes up the majority of the weight after the roundabout. Note the very short rise and fall of a third hypothesis at $t = 165$ s: it corresponds to the hypothesis "*the car stays in the roundabout*". The particles following this hypothesis are very quickly eliminated as the theoretical yaw rate does not correspond at all with the odometry data.

The second event is a lane change, during which the car goes from the right to the left lane at the end of the recording ($t = 275$ s) as shown in Fig. 3.23b. Before the event, the filter had isolated a single hypothesis. The lane change is expressed by a progressive transfer of particles to the other lane, which is translated by the steady decrease of weight for the original lane and the opposite for the new one. Moreover in this case, the lane change is happening at the same time as a lane bifurcation: this creates a new hypothesis (in cyan) as particles are cloned from the original lane (in blue). With the same principle as previously, this hypothesis is eliminated, although more slowly (the lane directions are not orthogonal, as it is the case for a roundabout exit). This only slows down the transition of all the weights from the right lane to the left one.

3.4.3.2 Urban Conditions

For the second dataset (see Fig. 3.15b), the HPL has been set to a value of 15 m. This is due to the presence of high-rise buildings inducing GNSS errors. 15 m is an upper-bound of the error encountered during this recording. Note that with a real-time estimation of the integrity bound, the HPL would have a dynamic value, adapting to the actual conditions.

Fig. 3.24 shows the evolution of the different hypothesis weights. The correct matching hypothesis is represented by the blue curve. The initialization takes place at the entrance of a roundabout. This situation is more challenging than the previous one: the weight variations are stronger but the filter finally converges to the left/right lane hypotheses. The car crosses multiple roundabouts during the travel (at $t = \{122, 206, 340, 520\}\ s$). It is visible that these areas create great variation of the



(a) At a roundabout.

(b) During lane change.

Figure 3.23 – Details of the test travel.

weights, not only at initialization. This is mainly due to the greater yaw rate variation. Moreover, lane detection is unavailable due to the generally poor camera performance in high curvature areas. The filter is then less constrained and particles tend to change link more easily. The short-lived hypothesis at the roundabout exit (as developed previously) is also observed.

Thanks to the fact that their evolution is independent from the GNSS fix, the particles are not sensitive to multipath issues: Fig. 3.25 show the weight evolution during such an event. The car follows the blue GNSS trajectory on the right (going northbound), the system is clearly affected by a multipath signal, deviating the position to the right of the road. This corresponds to $t = [516, 520]\,s$, Fig. 3.25b shows no variation during this time frame (the variations visible from $t = 524\,s$ are mostly due to the car entering a roundabout). This behavior is linked to the design of the algorithm, that relies more on odometry than on GNSS, used very sparsely.

3.4.3.3 Behavior in Lane Forking

A lane forking happens at $t = 450\,s$: the rightmost lane separates in two, one continuing on the bypass and another one on the right exiting the bypass (see Fig. 3.26a). During this travel, the car takes the exit, following a trajectory driving on the rightmost lane. The blue curve in Fig. 3.26b represents this trajectory, the cyan one corresponds to the right lane continuing on the bypass and the red one, the left lane.

Arriving at the lane forking, the right hypothesis bears the majority of the weight. At the forking (materialized by the right top node, denoted by a red dot on Fig. 3.26a), the particles belonging the right hypothesis are being cloned, leading to a weight transfer between the exit and the continuing right lane (respectively blue and cyan curves). The majority of the weight gets transferred to the cyan hypothesis. This is due to the fact that the car follows a straight trajectory that corresponds more closely to the continuing lane. The human driver tends to follow a smoother trajectory for comfort that may not match the map's reality (i.e. a stronger turn for an exit).

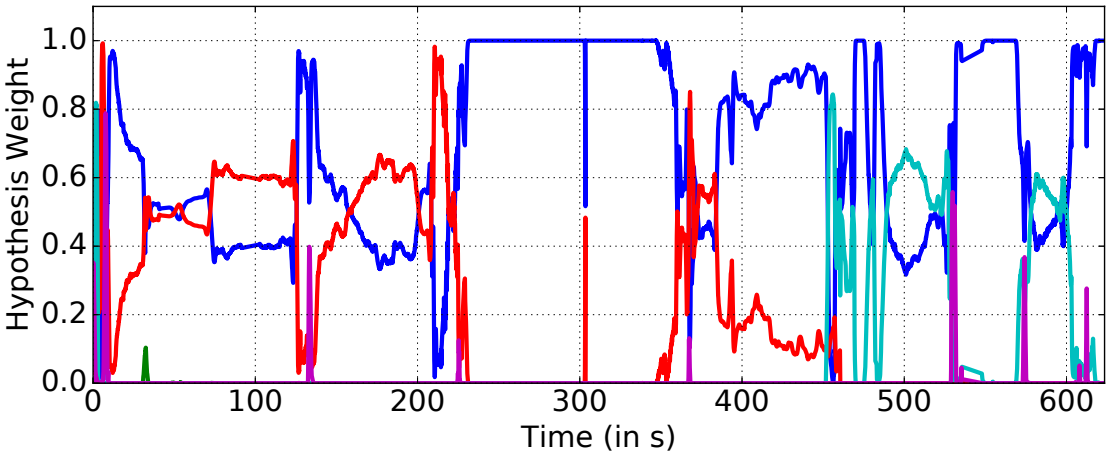
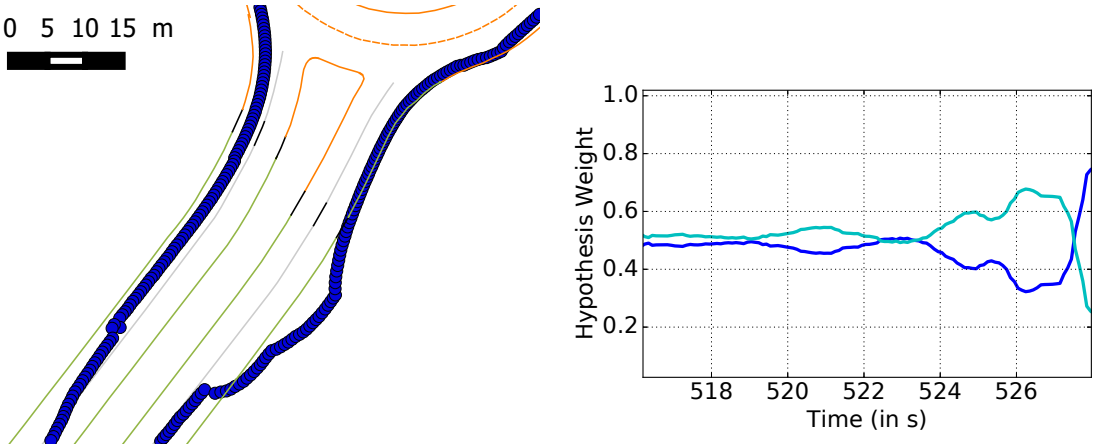


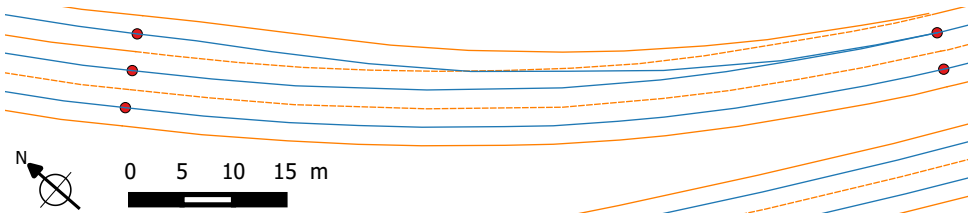
Figure 3.24 – Evolution of the hypothesis weights on the second dataset. Multiple hypotheses are present, mainly due to the intersection encountered (including roundabout exits), but two majors hypotheses are also prominent in this recording.



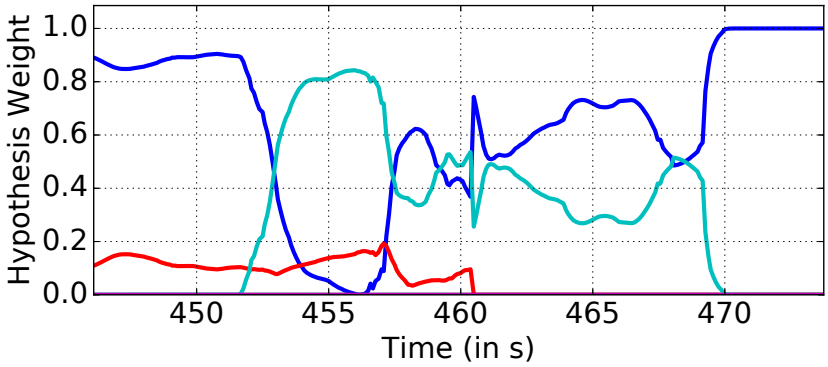
(a) GNSS track (in blue) in an area affected by multipath.

(b) The hypothesis weights stay relatively stable.

Figure 3.25 – The system travels through areas prone to GNSS multipath, but the filter is unaffected as the hypothesis weights stay stable.



(a) Map of the lane forking. The car travels westbound on the rightmost lane and takes the exit.



(b) Evolution of the weights associated with the different links in play.

Figure 3.26 – Study of a lane forking event.

This illustrates the importance of not setting a too narrow likelihood function for heading criteria, letting a margin for human induced trajectories. At $t = 457\,s$, the heading difference is noticeable and the driver matches the car's trajectory to the exit direction: the correct hypothesis becomes the most probable one. Meanwhile, the left lane hypothesis becomes too unlikely and is eliminated.

Later, at $t = 468\,s$, the exit takes a sharper turn, while the continuing lane goes straight to a roundabout entrance. The exit hypothesis becomes the only likely hypothesis with respect to the odometry data: the filter isolates it with high confidence.

As shown in Fig. 3.26b, the weights evolution is well adapted for decision making, as staying very smooth during the filter execution. Note that when a hypothesis is eliminated (at $t = 461\,s$), there is a transient phase that quickly stabilizes afterwards.

3.4.3.4 Quantitative Results

Table 3.2 shows performance metrics of the filter on one run: during the tests, the map-matching always included the correct matching hypothesis in the solution set. Moreover, for 67.8% of the time, this hypothesis was bearing the highest weight. Note that in cases where an incorrect hypothesis has the highest weight, the correct hypothesis generally has a comparable weight (see Fig. 3.24 at $t = [70, 125]\,\mathrm{s}$) or the situation is temporary (spike at $t = 210\,\mathrm{s}$). The filter converged to one or two hypotheses 95.1% of the time. This percentage takes into account the initial convergence and lane forking events that produces a larger number of hypotheses.

Table 3.2 – Matching metrics on the set of matching hypotheses.

Metric	%
Set containing the correct hypothesis	100
Set of 2 or fewer hypotheses	95.1
Correct best hypothesis	67.8

3.4.4 Localization Results

The SMMAL-PF method solves simultaneously the MM and localization problems. The localization performance is assessed in this section, comparing the results of the developed method and of a classical DR algorithm (simple integration of the odometry using the car's dynamic model and the CAN bus wheel speed and yaw rate). To do this, both algorithms have been executed using the same input data, corresponding to the urban conditions dataset.

As the DR algorithm is mono-hypothesis, the SMMAL-PF results must be adapted to do the comparison. The filter is executed normally, as described in Section 3.3 (i.e. potentially yielding multiple hypotheses), using typical parameter values (see Section 3.4.1.3). To obtain a mono-hypothesis result, a selection of one hypothesis has been made a posteriori, not impacting the filter's execution in real-time. When possible, the selected hypothesis corresponds to the matching ground truth. The resulting trajectory is then representative of the localization performance of the SMMAL-PF.

Fig. 3.27 compares the results of the two localization methods: the SMMAL-PF with the DR. As expected, a pure DR solution diverges rapidly while the developed solution is well localized around the actual roads thanks to constraints coming from the map. This shows the positive influence of the map constraints when combined with a PF to limit the odometry drift due to sensor biases and noises.

3.4.5 Map-Matching Consistency

3.4.5.1 Estimator Consistency

Bar-Shalom et al. [2001] qualify an estimator $\hat{x}$ (of the state x), with a covariance matrix P_x, as consistent at a given time if it satisfies:

$$\begin{aligned} E\left[x-\hat{x}\right] &= 0 \\ E\left[\left(x-\hat{x}\right)\left(x-\hat{x}\right)\right] &= P_x \end{aligned} \tag{3.69}$$

The first condition corresponds to the unbiasedness of the estimate, while the second expresses that its mean squared error should match the estimated covariance provided by the filter. In other words, a consistent estimator must represent the state with fidelity and, more importantly, must not be overconfident by underestimating its covariance which is often the case in practice.

This problem has been extensively studied for SLAM algorithms. For instance, Castellanos et al. [2004] characterized the EKF-SLAM performance and concluded that the method has a tendency to lose consistency quickly during its execution. Factors, such as an initial bias can affect negatively the filter's consistency as the

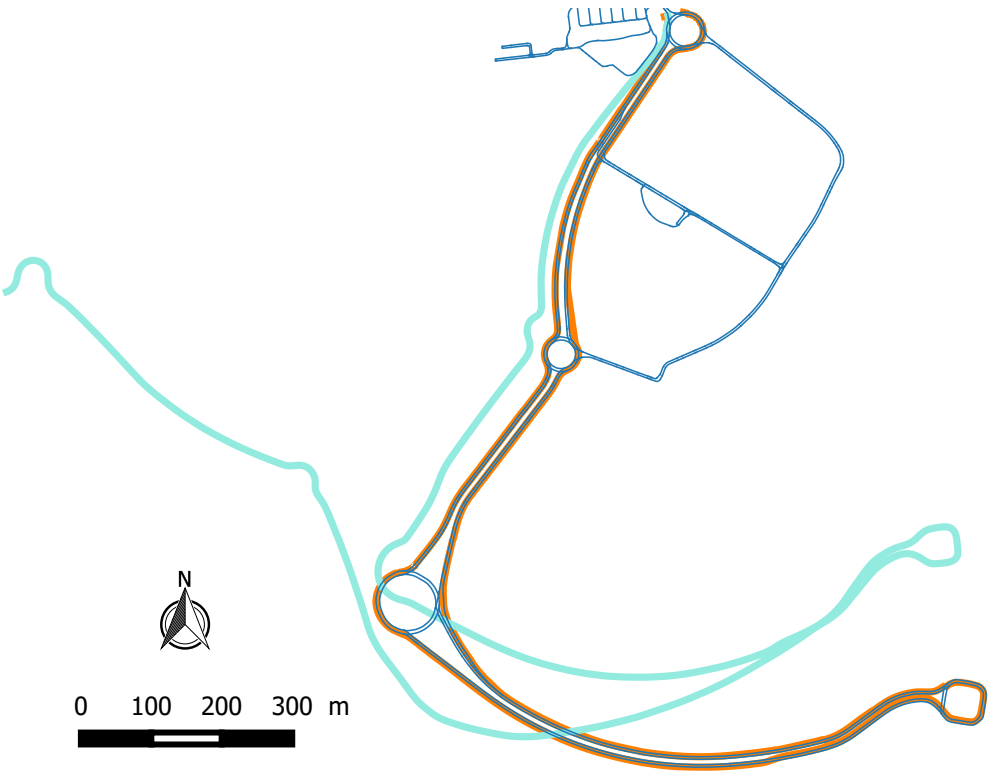


Figure 3.27 – Results of a classical DR-only solution (in blue) compared to the map-aided odometry (in orange).

covariance over-converges, not giving enough time to correct the bias: the filter does not verify (3.69) and thus is considered inconsistent.

3.4.5.2 Application to Multi-Hypothesis Map-matching

As developed all along this section, map-matching is a crucial element for positioning. In the scope of this research, its objective is to determine the ID of the current polyline. Studying the consistency of this ID estimate is thus a good indicator of performance. In the SLAM case, the state consists of spatial and continuous properties: the position of the system and of the different mapped features. It is the main difference with a map-matching algorithm whose main state component is the matched link.

This discrete property renders unsuitable the previous consistency characterization, as the notion of covariance, tied to a Cartesian position, is not applicable to it. The notion of consistency must be adapted to map-matching. In the case of multi-hypothesis map-matching, the element conveying the notion of uncertainty (as would the covariance do) is the possible presence of multiple hypotheses. Therefore, by analogy with the consistency criteria on the covariance of (3.69) (*the MSE must be contained within the covariance*), the map-matching consistency can be expressed as:

A map-matching algorithm is consistent if it returns as a solution a set of hypotheses containing at least the correct matching hypothesis.

Table 3.3 – Matching metrics on the set of matching hypotheses after 400 executions.

Metric	%
Set containing the correct hypothesis	97.7
Set of 2 or fewer hypotheses	95.8
Correct best hypothesis	64.3

3.4.5.3 Evaluation of the Map-Matching Consistency

Using the previous definition, the map-matching consistency of the algorithm is assessed in this section, using the map-matching ground truth described in Section 3.4.1.4. A repeatability study has been performed to estimate a statistic on the consistency criteria. The filter has been executed in batch 400 times in order to generate representative statistics (using the urban conditions dataset, containing more challenging situations). Each execution contains $5,986$ time steps, corresponding to the number of GNSS fixes returned by the receiver, at which the SMMAL-PF evaluates the map-matching. The algorithm has therefore been tested on a total of 2,394,400 occurrences which is significant for such a study.

Table 3.3 shows the resulting data. One of the most relevant statistic is the percentage of sets containing the correct hypothesis. It corresponds by definition to the percentage of time during which the filter is consistent. This statistical study showed a loss of consistency 2.3% of the time. This is due to the random nature of PF.

Fig. 3.28 provides details about per-execution statistics. It represents the histogram of the percentage of consistency loss per execution (expressed between 0 and 1): instead of considering all the time steps together, each program execution is taken individually to compute the percentage of time steps a loss of consistency happened during a single travel. A majority of the executions (57.8%, 231 out of 400) stay completely consistent, visible as a spike at the 0 value.

When studying the rest of the executions (i.e. the ones that have output some inconsistent results), the consistency losses only represent a small proportion of the travels: 95% of the executions bear less than 10% of inconsistent time steps, as shown by Fig. 3.29, the cumulative histogram of the percentage of consistency loss per execution. This indicates that, when the SMMAL-PF becomes inconsistent, it is for a short time. In other words, the filter is able to detect such events and act to converge again quickly which shows that the inconsistencies are of small scale.

3.5 Conclusion

Most MM algorithms aim to return a unique matching solution, taking the risk to be erroneous in ambiguous situations. The filter developed in this chapter follows another rationale: the goal is to return the smallest set of hypotheses that contains the correct matching with high probability. It has been designed to limit the risk of eliminating a likely hypothesis by returning a set of solutions.

Among the estimation approaches that have been presented, the PF has been chosen as fulfilling the previous requirements. PF is a common solution for MM, allowing multi-hypotheses to be tracked. The developed map-matching method, called SMMAL-PF, is able to handle multiple hypotheses that are bound to appear

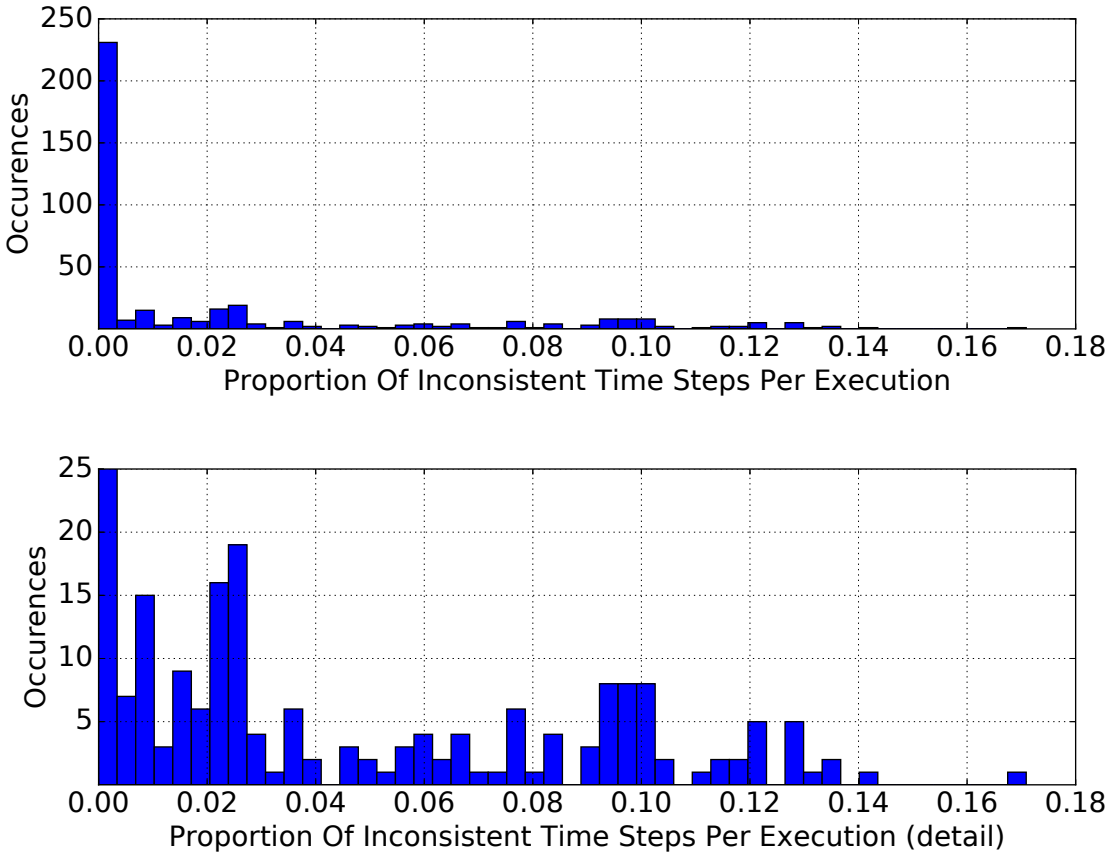


Figure 3.28 – Histogram of the proportions of inconsistent filter for each execution. The majority stays completely consistent through their execution. The histogram below represents the same data but zoomed for better readability.

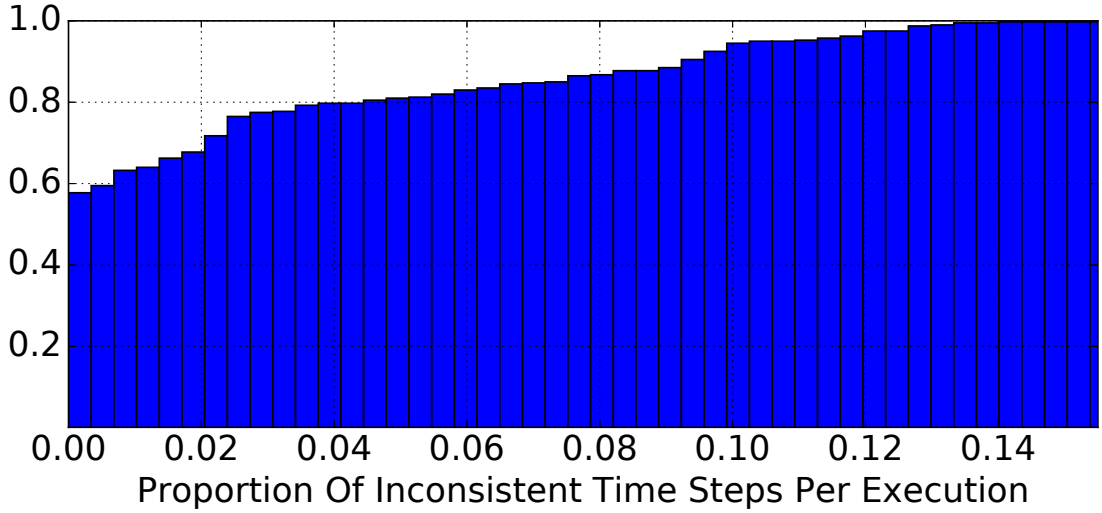


Figure 3.29 – Cumulative histogram of the proportions of inconsistent filter for each execution. The ratio of consistency loss stay small, representing 95% of the time less than 10% of a single travel.

when using HD-maps, by using the flexibility of PF. Map information is integrated into the filter during the likelihood calculation step. Other sources of information (e.g., exteroceptive information) can be integrated in the same manner. GNSS information is used minimally: only to initialize the filter and, as a gating process using the HPL value, to eliminate the particles that went astray. This choice has been made to keep the SMMAL-PF as independent from the GNSS as possible.

Contrarily of what is usually done, the multi-hypothesis ability is extended to the output of the map-matching algorithm, focusing on the correctness of the solution rather than its uniqueness: if some ambiguities are still present, the SMMAL-PF prefers to output a set of hypotheses, conserving the matching consistency rather than risking to select an incorrect one. This vision is justified by the targeted applications: classic matching algorithms are used in applications such as guidance, path planning or control that require a unique solution; whereas the SMMAL-PF algorithm has been designed to be used in map-related learning applications. These must rely on a robust and accurate localization and it is preferable to discard learning information than misplacing it.

The performance of the SMMAL-PF has been tested using real data, showing a high consistency. Some errors are however present, due to the stochastic nature of PF and the limited computing power available (limiting the number of particles). The next chapter will present a coherence checking method, comparing the SMMAL-PF hypotheses with the GNSS fix in order to cope with these errors. It will also be used to try reducing the hypothesis set down to a single one, by eliminating the incoherent ones, in order to making the SMMAL-PF result usable by all applications.

Chapter 4

Integrity Monitoring and Decision Making With a Multi-Hypothesis Map-Matching System

Contents

4.1 Introduction 104
4.2 Integrity Monitoring Methods and Architectures 104
4.2.1 Receiver Self Integrity Monitoring 104
4.2.2 Integrity Monitoring Using GNSS and Additional Sensors 107
4.3 SMMAL-PF Coherence Checking Using Redundancy . 108
4.3.1 Working Hypotheses 108
4.3.2 Proposed Architecture 109
4.3.3 "Use/Don't Use" Decision 109
4.3.4 Five Different "Use/Don't Use" Situations 111
4.4 Comparison of Different Tests for Hypothesis Selection 113
4.4.1 Deterministic Approach 113
4.4.2 Covariance Estimation for Consistency Testing 116
4.4.3 Consistency Test Based on a Chi-Squared Test 118
4.4.4 Consistency Test Based on the Kullback-Liebler Divergence 127
4.5 Graphical Illustration of the Results 132
4.6 Conclusion 134

4.1 Introduction

As presented in Chapter 3, positioning for an intelligent vehicle implies a map-matching procedure using GNSS positions merged with other sensors. The discrepancy that can occur sometimes between high-definition maps and low-accuracy GNSS fixes has to be taken into account. To this end, a map-matching method with a multi-hypothesis output has been presented in the previous chapter.

In practice, most applications that use the output of the navigation system can only manage a single positioning hypothesis. The matching results of a multi-hypothesis navigation system must therefore be refined in order to be usable by such systems.

This chapter deals with the usability of the map-based positioning system, i.e. the characterization of the hypotheses set. Two approaches are developed and studied: the first one is a deterministic approach, calculating the distance between the GNSS estimate and the matching hypotheses, that can be assimilated to an uncertainty level. The second is probabilistic: the hypotheses set is first refined, computing a metric considering the GNSS estimate and hypotheses as distributions and then analyzed to determine its usability.

4.2 Integrity Monitoring Methods and Architectures

This section presents different methods that can be applied to monitor a GNSS receiver's integrity. This monitoring is needed because, contrarily to the space segment, the receivers can evolve in extremely varying environments (ranging from open-sky conditions to urban canyon) that can only be monitored at the receiver side. Depending on the methodology used, GNSS integrity monitoring can be separated into two main categories.

4.2.1 Receiver Self Integrity Monitoring

The internal integrity monitoring regroups methods basing their monitoring on internal information only. It handles GNSS faults (for instance the failure of a clock inside a satellite). Measurement redundancy is mandatory in order to apply a 3-step methodology called *Detection, Identification, and Adaptation (DIA)* [Teunissen, 1990]. One of the most representative examples is the RAIM algorithm [Brown, 1996], that applies a DIA method called Fault Detection and Exclusion (FDE). FDE simplifies DIA into two main steps. In this section, we recall the basis of such an approach.

Note that some methods do not perform in the measurement domain but directly in the position domain [Blanch et al., 2007; Brenner, 1996; Joerger et al., 2014]. These methods, called solution separation, will not be developed here.

4.2.1.1 Fault Detection

The first step remains the fault detection. To do so, statistical tests are applied to the measurements. A criteria detecting the presence of a fault must be defined. To do so, the influence of a bias is studied.

It is assumed that the observation model is linear:

$$y = Hx + b + e_y \tag{4.1}$$

where y is the observation vector, b is the bias vector and $e_y \sim \mathcal{N}(0, Q_y)$. Therefore:

$$y \sim \mathcal{N}(\mu_y, Q_y) \mid \mu_y = Hx + b \tag{4.2}$$

Aitken [1936] gives the minimum variance estimation as:

$$\begin{aligned} \hat{x} &= (H^T Q_y^{-1} H)^{-1} H^T Q_y^{-1} y \\ \hat{x} &= H^+ y \end{aligned} \tag{4.3}$$

The residual vector r is then equal to:

$$\begin{aligned} r &= y - H\hat{x} \\ &= y - HH^+ y \\ &= (I - HH^+) y \\ r &= S \cdot y \end{aligned} \tag{4.4}$$

It can be deduced from (4.4) that $r \sim \mathcal{N}(\mu_r, Q_r)$ with $\mu_r = S \cdot \mu_y$ and $Q_r = S \cdot Q_y \cdot S^T$. The following studies the influence of the bias vector on μ_r.

Considering a null vector $b = \overrightarrow{0}$:

$$\begin{aligned} \mu_r &= S\mu_y = S(Hx + b) \\ &= SHx \\ &= (I - HH^+)Hx \\ &= Hx - Hx \\ \mu_r &= \overrightarrow{0} \end{aligned} \tag{4.5}$$

The distribution is thus centered. If $b \neq \overrightarrow{0}$:

$$\begin{aligned} \mu_r &= S(Hx + b) \\ &= S \cdot b \\ \mu_r &\neq \overrightarrow{0} \end{aligned} \tag{4.6}$$

In this case, r is not centered. This difference can therefore be used as criterion to determine the presence of a fault in the measurements. Often chosen as decision variable is the squared sum of the residuals normalized with their respective covariance, called Normalized Sum of Squared Error (NSSE):

$$\text{NSSE} = \sum_{i=1}^{m} \left(\frac{r_i}{\sigma_i} \right)^2 \tag{4.7}$$

with m, the number of measurements (i.e. the dimension of r). As the error components are considered Gaussian, the NSSE follows a χ^2 distribution with $m - n$

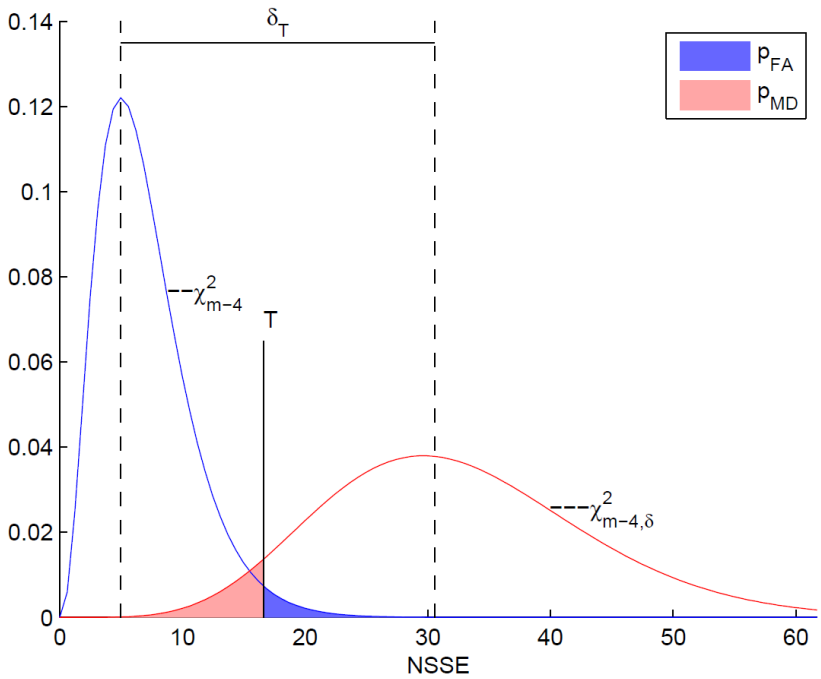


Figure 4.1 – Illustration of the statistical test on the NSSE, using χ^2 distributions. It is characterized by a probability of false alarm p_{FA} and of miss-detection p_{MD}, set by determining the test threshold T and the non-centrality parameter δ_T (image from [Drevelle, 2011]).

degrees of freedom, n being the state dimension. In the case of GNSS positioning $n = 4$, therefore in the nominal case NSSE $\sim \chi^2_{m-4}$. As a consequence, the fault detection requires at least 5 measurements to be applicable.

The test has two hypotheses: H_0: "no fault is present" and H_1: "a fault is detected". Using respectively (4.5) and (4.6), they can be rewritten:

$$\begin{aligned} H_0 : \text{“no fault is present”} &\Leftrightarrow NSSE \sim \chi^2_{m-4} \\ H_1 : \text{“a fault is detected”} &\Leftrightarrow NSSE \sim \chi^2_{m-4,\delta} \end{aligned}$$

where $\chi^2_{m-4,\delta}$ is the non-central χ^2 distribution with a non-centrality parameter δ.

This setting is thus perfectly fit to apply a χ^2 test [Pearson, 1900], taking the NSSE as decision variable. If the NSSE stays under a given decision threshold T it is assumed to follow the centered χ^2distribution, hence validating H_0. Otherwise, the alternative hypothesis H_1 is.

Fig. 4.1 shows the two distributions in play along with the threshold T. Two important properties are visible graphically: the probability of false alarm p_{FA} and of miss-detection p_{MD} (respectively the blue and red surfaces). They are defined by:

$$\begin{cases} p_{\text{FA}} &= P(\text{NSSE} > T | H_0) \\ p_{\text{MD}} &= P(\text{NSSE} < T | H_1) \end{cases} \tag{4.8}$$

In general, these values are set by the application needs. Two parameters enable the adjustment of these properties to the right values. First, the threshold T is chosen to correspond to a target p_{FA}, it is the only leverage as the centered χ^2

distribution is fixed. The second parameter is the non-centrality parameter δ_T. It is fixed after T to correspond to a target p_{MD}.

4.2.1.2 Fault Exclusion

The exclusion step combines DIA's identification and adaptation steps: in place of adaptation, FDE simply excludes the faulty measurement out of the calculation. An important assumption is the presence of at most a single faulty measurement. Note that this assumption may not be respected in automotive application, but holds true in the FDE development context (i.e. aeronautics).

In the case a fault has been detected by the previous test, the continuation of the algorithm is the identification of the faulty measurement. Most of the existing strategies rely on finding the measurement maximizing a criterion linked to the induced level of error. For example the w-test [Baarda, 1968] searches the measurement maximizing the absolute value of w_i so that:

$$\begin{aligned} w_i &= \frac{r_i}{\sigma_i \sqrt{S_{i,i}}} \\ i_{\mathrm{fault}} &= \underset{i=1..m}{\operatorname{argmax}} |w_i| \end{aligned} \tag{4.9}$$

Once identified, the faulty measurement i_{fault} is discarded and the position can be computed without.

4.2.2 Integrity Monitoring Using GNSS and Additional Sensors

Having multiple sources for the same information is an important principle often applied to increase functional safety. Redundancy in measurements can be used to detect when a fault affects one of them, as presented in the previous section with the RAIM methods. This relies on multiple measurements of the same type (e.g., pseudoranges).

Another category of integrity monitoring processes information from other systems, diversifying the sources of information. For instance, Nebot and Durrant-Whyte [1999] apply a stronger redundancy requirement by using physically different sensors (see Fig. 4.2). This yields failure modes that are different and independent, so that failures from a given sensor would be detectable by another.

To do this the authors chose sensors relying on different physical principles to create two separate estimation loops: the first based GNSS and the second, millimeter wave radars. This effectively reduces the risk of having simultaneous failures to a minimum. These two estimation loops perform independently with their own sensor data and return a result each. An arbitration mechanism is then needed to combine them into a joined final result. It consists of a statistical test estimating if a significant difference between the loops exists. If it is the case, a way to determine which estimate is correct is to compare them to a third source of information.

This architecture then acts closely as a 2-out-of-3 voting system, common in functional safety applications. A N-out-of-M voting system is composed of M sources and N of them have to agree to validate the gathered information. Nebot and

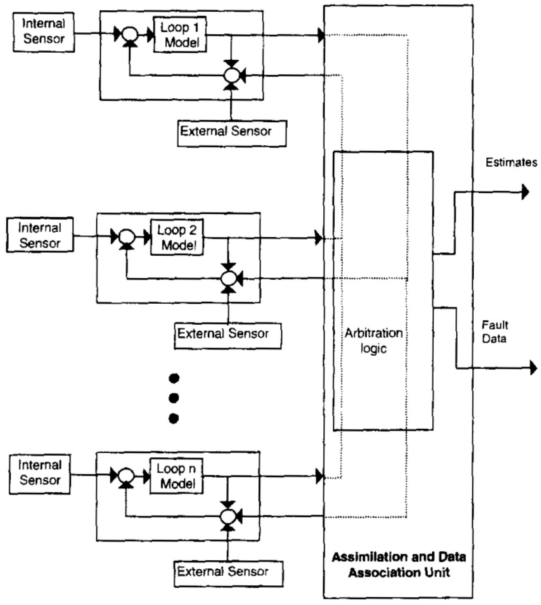


Figure 4.2 – Using redundancy is a key element for functional safety. Using different estimation models limits the common failure modes (image from [Nebot and Durrant-Whyte, 1999]).

Durrant-Whyte [1999] deviate from this by considering the third source of information as a referring value, corresponding to a residual monitoring sensor (monitoring the two other sources). If the two estimating loops agree, their result is considered correct. If they do not, they are compared to the referring source. If one matches it, this result is considered correct, otherwise no result is considered as no data coherence has been found. This procedure determines the usability of the positioning system at the given time.

4.3 SMMAL-PF Coherence Checking Using Redundancy

This section describes the coherence checking principles applied to a multi-hypothesis localization system, such as the SMMAL-PF. The test procedure is presented as well as illustrative examples of the different possible decisions.

4.3.1 Working Hypotheses

As a reminder, this study is based on the following hypotheses:

- The HPL value respects the positioning integrity.
- The map is accurate and correct.
- A dual-frequency GNSS receiver is used.
- An accurate odometry model is available.

The first two are directly related to the filter's likelihood management. If the HPL is smaller than the error at a given time, the filter risks to eliminate all the particles, causing a reset, thus diminishing its availability. In the worst case scenario, the filter could even diverge, returning an erroneous matching until integrity is restored with a correct HPL value. Symmetrically, the map must be accurate as a geometrical offset can cause a hypothesis to be wrongly eliminated by exceeding the HPL. It must also represent the road network without errors as the odometry information must follow the road (on-road driving assumption).

The latter two express the need for a good knowledge about the system input. Using a dual-frequency GNSS receiver removes some sources of bias (the ionospheric perturbations) and thus provides a much more controlled positioning. Having a correct model is also important as the whole odometry relies on it, the odometer sensors must also be characterized accurately to know the operating noise. These hypotheses can be considered as respected since consumer vehicles are modeled during their development and their set of sensors is consistent, so a statistical study can easily be performed.

Note also that this system does not aim to provide positions at a very high frequency (as could a control system expect, for example): the algorithm focuses on the correct use of the map.

4.3.2 Proposed Architecture

This section presents the principle of coherence checking applied to the map-matching algorithm presented in Chapter 3. The developed method combines the two usages of redundancy presented in Section 4.2. Nebot and Durrant-Whyte [1999]'s use of multiple estimation systems is applied by separating the positioning system into two sources: on one hand, the GNSS fix and on the other hand, the result from the map-aided odometry. This holds true if the two sources are independent, hence the map-matching algorithm design with minimal use of GNSS data (see Chapter 3).

One noticeable difference is that the SMMAL-PF is designed to output multiple hypotheses when needed, adding another level of redundant information that must be managed. Instead of comparing only two sources, the algorithm must consider every hypothesis individually during testing. All hypotheses are distinct by design, comparing them between each other is therefore unnecessary. The test is thus performed only between each hypothesis and the GNSS fix to keep exclusively the coherent hypotheses, as presented on Fig. 4.3.

A final decision step is then performed in order to determine the global usability of the positioning system.

4.3.3 "Use/Don't Use" Decision

The choice of a coherence test will be presented in Section 4.4, studying different tests between the hypotheses and the GNSS fix. This section considers the test as chosen and presents first the decision procedure.

At this step of the algorithm, all remaining hypotheses are coherent with the GNSS fix. By definition, they also correspond to positions matching the car's DR with the map data. In other words, these hypotheses verify every positioning-related sensors (in the present case: GNSS, map data, odometry and camera information if

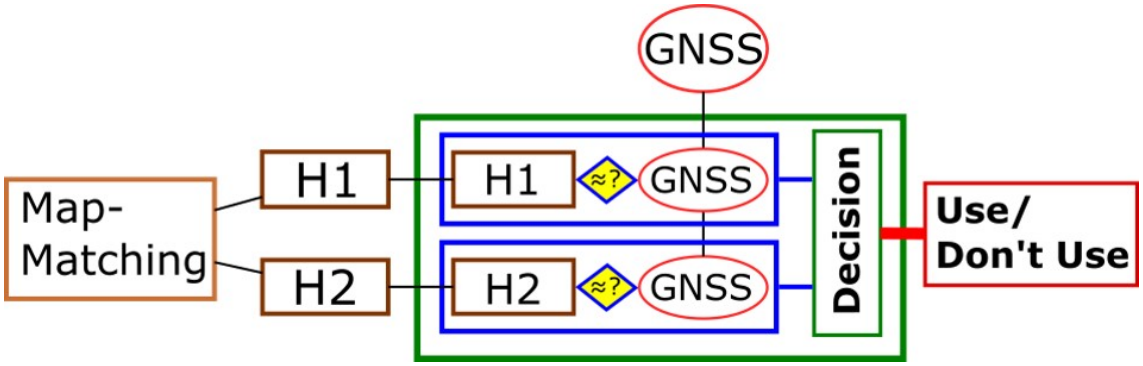


Figure 4.3 – Organization of the coherence checking procedure. It takes as input the multiple hypotheses from the map-matching and the GNSS fix. Each hypothesis is then compared to the GNSS to take a decision on the usability of the positioning system.

Algorithme 4.1 Testing the usability of the positioning system by checking the coherence of the hypotheses.

1: **function** USABILITY TEST(CoherentHypos)
2: **for each** hypothesis j **in** CoherentHypos **do**
3: **if** $w_j <$ weight threshold **then** ▷ see Section 4.3.3.1
4: **Remove** j **from** CoherentHypos
5: **end if**
6: **end for**

7: **if card**(CoherentHypos)$= 1$ **then**
8: **Use** ▷ A unique coherent hypothesis, **Use**
9: **else**
10: **Don't Use** ▷ Multiple coherent hypotheses or none, **Don't Use**
11: **end if**
12: **end function**

available) and are thus the best candidates for the matching result.

4.3.3.1 Eliminating Low-Weight Hypotheses

An additional selection is applied to the hypotheses to eliminate the least likely elements. This procedure avoids considering low-likelihood hypotheses (in the sense given by the map-matching), that nevertheless satisfied the coherence test. Such a situation can arise for example when a bias suddenly affects the GNSS (e.g., due to a multipath) and closely matches a low-weight hypothesis. Even though the previous test considers them coherent, the map-matching algorithm gave this hypothesis a low level of trust.

A basic implementation is a weight threshold (e.g., 0.1) under which these potentially misleading hypotheses are eliminated (see Algorithm 4.1, line 2 to 6). After this selection step, the remaining hypotheses are thus both coherent with all the positioning data and trusted by the filter (i.e. with high weight).

4.3.3.2 Testing the Final Hypothesis Set

The usability of the positioning system (i.e. the GNSS receiver together with the different available sensors) can be then assessed from this final set by simply verifying if the previous selections managed to isolate a single hypothesis (see Algorithm 4.1, line 7 to 11).

The test has two possible outcomes, tagging the positioning system as:

- *Use*, if the remaining set contains only one single hypothesis. This situation is considered to have no matching ambiguity remaining: the GNSS positioning is in accordance with the map-matching results and the filter gives a high likelihood to the hypothesized position. The positioning system can be trusted and a single solution is returned.

- *Don't Use*, if the remaining set contains multiple hypotheses or none. In the first case, the filter was unable to resolve every ambiguity and cannot decide between multiple hypotheses. In the second case, all hypotheses have been eliminated during the previous selection. It can either be due to a fault detection and/or a too low likelihood. In either case, the positioning system can not be trusted in the current situation where no definitive decision can be made on the matching result.

This is the final result of the coherence test. The algorithm has determined the usability of the position computed, i.e. if the matching result can be trusted or if some ambiguities remain. Moreover, in the case where the situation is tagged as *"Use"*, this procedure ensures a compatibility with most systems by outputting a single position, which is what most applications expect from a positioning system.

Note that the GNSS is a key element of the procedure. In the absence of GNSS (e.g., inside a tunnel), the redundancy of the positioning system is not assured. As a consequence, in the same manner as RAIM, the usability system is unavailable: no decision can be taken. A safe handling of the situation would be to label the positioning system as *"Don't Use"*.

4.3.4 Five Different "Use/Don't Use" Situations

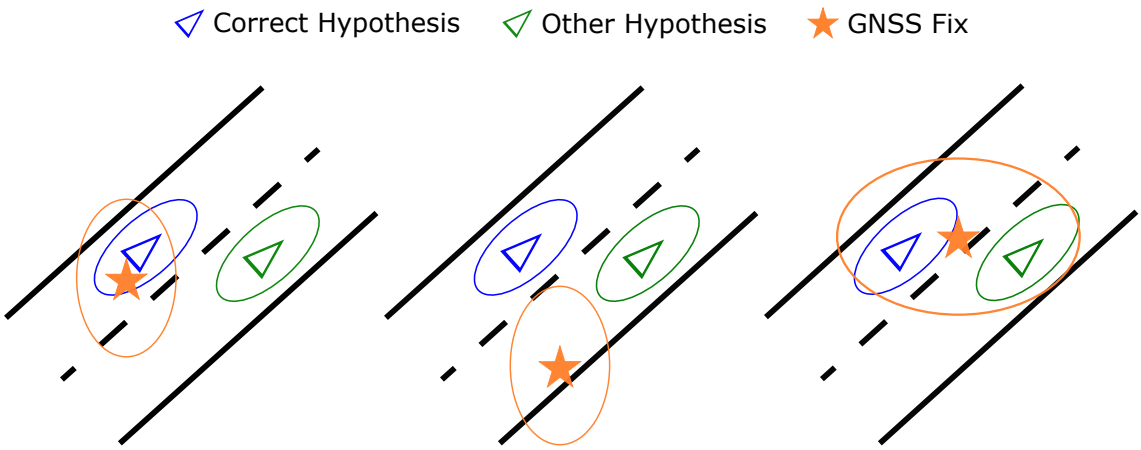
This section illustrates the different situations the positioning system can encounter. These examples consider a solution set composed of two high-weight hypotheses, representing the left and right lanes of a road. They are represented in Fig. 4.4 by the blue and green arrows, with uncertainty ellipses. The blue hypothesis is considered as the correct one. The GNSS is represented by an orange star and uncertainty ellipse. Note that these values are purely illustrative and do not represent an actual situation.

For simplicity, a hypothesis is considered as coherent if it is inside the GNSS uncertainty ellipse (actual test is described in Section 4.4).

4.3.4.1 Nominal Results

The three different situations yield a correct classification:

- **Correct Use** (Fig. 4.4a): only the correct hypothesis is coherent with the GNSS fix, the green one is eliminated. This leaves a unique hypothesis which



(a) Correct *Use*: a single coherent hypothesis.

(b) Correct *Don't Use*: eliminating all hypotheses.

(c) Correct *Don't Use*: subsisting ambiguities.

Figure 4.4 – Illustration of the different cases that can occur with a correct classification.

is tagged as *"Use"*. This is the positive outcome of the test, indicating that the positioning system is functional.

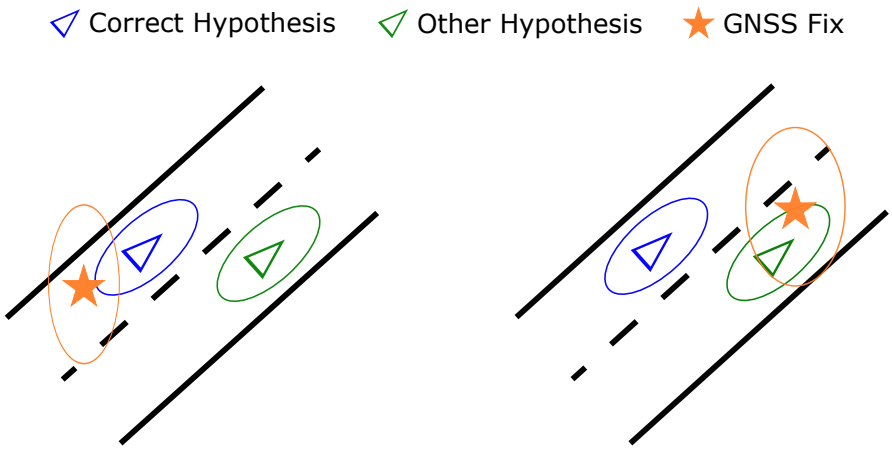
- **Correct Don't Use**: this classification happens in two different situations, in case that none of the hypotheses is coherent with the GNSS fix (Fig. 4.4b) or if both are (Fig. 4.4c), giving a single solution with integrity is impossible. The situation is tagged as *"Don't Use"*: the localization systems are not coherent or some ambiguities were not eliminated. In either case, the localization system must not be used safely.

4.3.4.2 Degraded Results

In some occasion, the filter can provide a wrong classification. These events can be separated into two classes:

- **False Alarm** (Fig. 4.5a): the GNSS fix and the correct hypothesis are declared incoherent by the test even though the GNSS position is correct. The correct hypothesis is eliminated and the situation is tagged as *"Don't Use"*. This can be considered as a false alarm, since a favorable situation is discarded. But by doing so, the system integrity has not been violated, therefore the algorithm performed as intended. The notion of false alarm may not be meaningful in this case. It is more correct to qualify this as a loss of availability of the positioning system caused by a too pessimistic test.

- **Missed Detection** (Fig. 4.5b): in the worst case (responsible for a loss of integrity), the GNSS is faulty and matches uniquely with a wrong hypothesis. This situation will be tagged erroneously as *"Use"* with the incorrect solution and is misleading. However, this kind of event is unlikely since the GNSS is supposed to not be affected by constant bias in extended periods. The same hypothesis is made by Nebot and Durrant-Whyte [1999], as both estimation must fail in the exact same way. The probability of occurrence of such an event is usually very small if the system has been well designed.



(a) Incorrect *Don't Use*: false alarm.

(b) Incorrect *Use*: missed detection.

Figure 4.5 – Illustration of the different cases that can occur with an incorrect classification.

4.4 Comparison of Different Tests for Hypothesis Selection

This section presents the hypothesis selection procedure required before making the usability test described in Section 4.3.3. The different methods presented rely on the comparison between the two position estimates available: the map-aided odometry and the GNSS fix, focusing on handling the multi-hypothesis characteristic of the odometry.

4.4.1 Deterministic Approach

A first approach is inspired by the RAIM. The aeronautic HUL (see Section 2.3.1.3) estimates the uncertainty level of a given situation by studying the geometry, level of measurement noise and positioning residuals. It considers the worst case scenario. If the HUL exceeds a threshold, the coherence is not validated.

Following the same principles, but adapting them to the current setup (no access to raw measurements in the scope of this research), a metric similar to the HUL is developed, using the multiple hypotheses returned from the map-matching odometry [Li et al., 2017b].

4.4.1.1 Principle

It is observed that in practice, GNSS receivers often tend to underestimate their uncertainty, potentially leading to incorrect covariance matrices. Similarly, a PF can also return incorrect covariance estimates, depending on its tuning. Based on these observations and for simplicity, this section presents a method removing the covariance from the calculation, in order to be unaffected by its possible error.

To adapt to the available data, the calculations takes place in the position domain. The GNSS fix and the different matching hypotheses are then considered as plain 2D geographic positions (an hypothesis is represented by the weighted mean

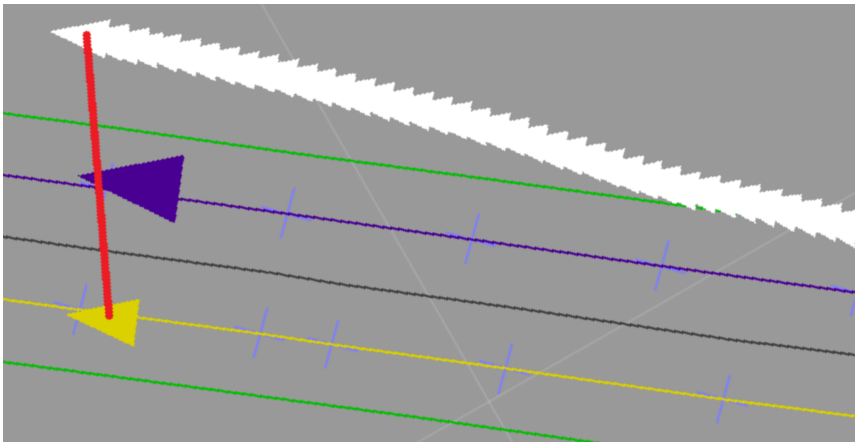


Figure 4.6 – Biased GNSS position (the GNSS track over time is displayed by white triangles, a deviation is clearly visible) along with the associated matching hypotheses at the current time index (colored triangles). The MA-HUL is denoted by the red line, corresponding to the distance between the GNSS position and the farthest matching hypothesis.

position of all its composing particles). Relying on map-matching hypotheses, the metric is named Map-Aided-HUL (MA-HUL) [Li et al., 2017b].

The principle of the MA-HUL is to estimate the coherence by comparing the GNSS position with the matching hypotheses. The multiple hypotheses mirror the ambiguity. Note that the GNSS fix can be affected independently by a bias or by multi-path.

4.4.1.2 Euclidean Distance

The Euclidean distances between the GNSS fix and the position hypotheses (see Section 3.3.8) are indicators of the coherence between the GNSS fix and the results from the map-matching. They illustrate the discrepancy between the two sources of positioning. They can be viewed also as estimates of the GNSS position uncertainty with respect to the map-matching dataset [Li et al., 2017b]. In other words, they approximate the uncertainty of the whole positioning system by considering the matching hypotheses (individually), the current GNSS position and all related biases.

The MA-HUL is set to be the largest of these computed distances (see Fig. 4.6):

$$\text{MA-HUL} \ = \max_j ||X_{\text{GNSS}} X_{\text{hyp}_j}||^2 \tag{4.10}$$

where X_{GNSS} is the GNSS fix and X_{hyp_j} is the jth hypothesis estimated position, represented by its barycenter (weighted mean of the coordinates of all particles).

Choosing the largest distance is motivated by integrity considerations, as done in RAIM. The MA-HUL can also be viewed as the minimum radius of a circle centered on the GNSS fix enclosing all hypotheses, being comparable to an HUL. The MA-HUL thus characterizes the positioning system with a conservative level of consistency linked to the map-matching process.

The MA-HUL, homogeneous to a distance, can be used similarly to a classic HUL to assess the integrity of the positioning system. By comparing it with a threshold

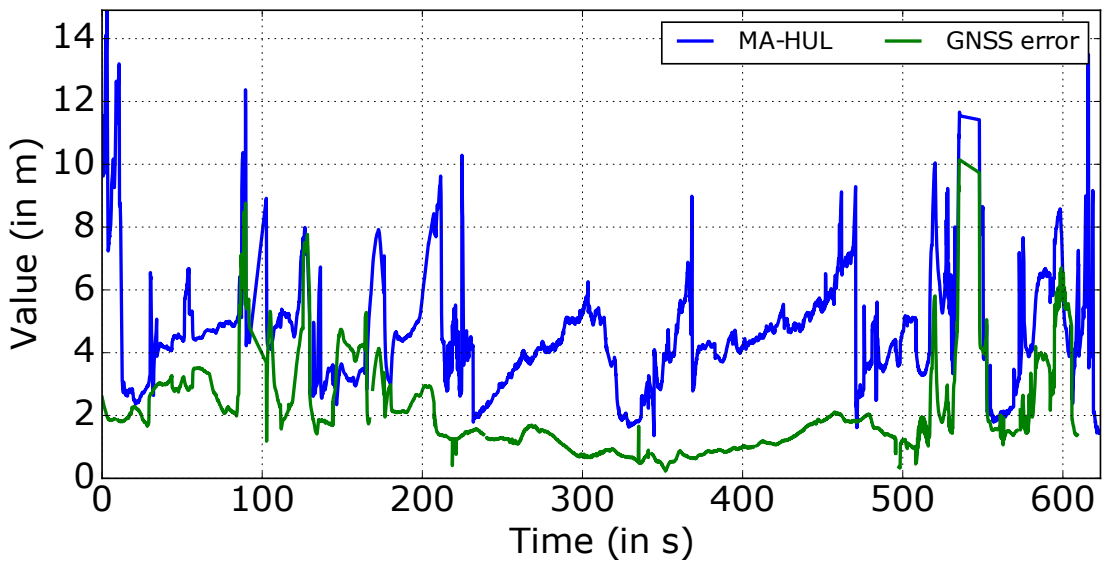


Figure 4.7 – The MA-HUL (in blue) appears as an upper bound of the GNSS error (in green, computed using data gathered by an IMU).

representing the maximum acceptable error, the system can determine when the error level gets out of safe operation. The threshold has to be set to an adequate value, corresponding to the operational needs of the system.

4.4.1.3 Results

The MA-HUL is analyzed on the mixed test travel. Its raw value, compared with the GNSS error, is shown in Fig. 4.7. The MA-HUL acts 94% of the time as an upper-bound of the GNSS error, respecting its original design inspired by the RAIM's HUL. Additionally, the MA-HUL does not exceed the 15 m HPL value, as it relies on the map-matching solution that does not allow particles to cross that HPL limit.

This metric is an overestimation of the GNSS error which is sometimes quite pessimistic. This is clearly visible between $t = [230, 330]\,s$ where the MA-HUL can be up to 5 times greater than the actual error. This demonstrate the pessimistic nature of this metric: this can cause a great loss of availability to the system. For example, on the previous time interval, a system requiring an maximum error of $3\,m$ should have been available but the MA-HUL would indicate a potential greater error, thus rendering the system unavailable. One of the reason of such behavior, is the particle spread occurring during the PF map-matching: in straight lines (e.g., $t = [230, 330]\,s$) the particle set tends to spread along the road direction and to retract during bends. The marking detection also influence the set by centering it around the correct lateral position. The separation of the MA-HUL into two metrics, along-track and cross-track, is developed in Appendix C. This causes lower uncertainty level in the separated directions, potentially enabling applications requiring accurate positioning in a single direction.

Seemingly contradictory with the previous statement, the MA-HUL can represent closely the GNSS error (e.g., at $t = 130\,s$). This important variation of performance is a limiting characteristic for this metric, appearing being poorly discriminant in certain cases.

Algorithme 4.2 Probabilistic coherence checking procedure.

```
function COHERENCE CHECKING
    N_GNSS ← GNSS Distribution Estimation                    ▷ see Section 4.4.2.1
    for each hypothesis j do
        N_Hyp[j] ← Hypothesis Distribution Estimation        ▷ see Section 4.4.2.2
    end for

    CoherentHypos ← COHERENCE TEST(N_GNSS, N_Hyp)            ▷ see Algorithm 4.3
    USABILITY TEST(CoherentHypos)                            ▷ see Algorithm 4.1
end function
```

4.4.1.4 Discussion

The MA-HUL appears to be a good metric to estimate the GNSS error upper bound. It is however not adapted for coherence checking because of its pessimistic nature. In the best case (i.e. when there is only one map-matched hypothesis), the MA-HUL is a direct indicator of the positioning error and can thus be used to check the consistency, outputting a *Use* flag if it remains below a threshold (as is done with the aeronautic counterpart).

Unfortunately, as the map-matching process often returns several hypotheses, MA-HUL values are often large (e.g., by considering a low-likelihood far hypothesis), decreasing the positioning system availability. Moreover, the MA-HUL being deterministic it is complicated to handle a risk associated with the decision rule.

For these reasons, the following section studies probabilistic methods to address the coherence checking problem.

4.4.2 Covariance Estimation for Consistency Testing

Even though the different covariances may not be well estimated, using them can be worthwhile to compensate the weakness of a purely deterministic approach. A probabilistic approach is thus developed and presented in Algorithm 4.2.

Taking into account the covariances of the different estimates enables a better consideration of the situation's uncertainty. The first step is therefore the covariances estimation.

4.4.2.1 GNSS Covariance

The covariance estimation of a GNSS fix is generally returned by the receiver itself, reflecting the uncertainty affecting the positioning computation. For example, the NMEA standard GST sentence provides a 1-sigma uncertainty ellipsoid, specifying the semi-major a and semi-minor b axes and the orientation ϕ of the ellipsoid.

The first step is to compute the corresponding standard deviations. Not considering the rotation, i.e. working in the ellipsoidal frame projected in the east-north Cartesian reference frame, the covariance matrix is diagonal:

$$\Sigma_{\text{ell}} = \begin{bmatrix} \sigma^2_{\text{ell}_x} & 0 \\ 0 & \sigma^2_{\text{ell}_y} \end{bmatrix} \tag{4.11}$$

a and b, returned by the GNSS, verify the ellipse equation:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \tag{4.12}$$

For a Gaussian distribution, a 1-sigma ellipsoid contour expressed in the ellipsoidal frame is given by:

$$\left(\frac{x}{\sigma_{\text{ell}_x}}\right)^2 + \left(\frac{y}{\sigma_{\text{ell}_y}}\right)^2 = 1 \tag{4.13}$$

Therefore:

$$\begin{cases} \sigma_{\text{ell}_x} & = a \\ \sigma_{\text{ell}_y} & = b \end{cases} \tag{4.14}$$

To get a covariance matrix corresponding to the GNSS data in the working frame, the orientation of the ellipse has to be taken into account by applying a rotation of an angle ϕ to the diagonal covariance matrix:

$$\hat{\Sigma}_{\text{GNSS}} = R.\Sigma_{\text{ell}}.R^T \tag{4.15}$$

with R the 2D rotation matrix of angle ϕ. Generally, this rotation creates cross-covariance terms so that:

$$\hat{\Sigma}_{\text{GNSS}} = \begin{bmatrix} \sigma^2_{\text{GNSS}_x} & \sigma_{\text{GNSS}_x\text{GNSS}_y} \\ \sigma_{\text{GNSS}_x\text{GNSS}_y} & \sigma^2_{\text{GNSS}_y} \end{bmatrix} \tag{4.16}$$

In practice, the GNSS uncertainty data are generally overconfident, since effects such as atmospheric perturbations or ephemeris errors are not taken into account. Covariance estimates can then be underestimated by few meters. To compensate these sources of error, we propose to add a constant variance offset (e.g., 1 m) to the GNSS covariance before doing the rotation.

4.4.2.2 Hypotheses Covariance

Each hypothesis Hyp_j is a set of n_j weighted particles (with P_j, the set containing the indexes of the particles belonging to the jth hypothesis). In the following, the weights are considered normalized by hypothesis ($\sum_{i \in P_j} w^i = 1$). Each hypothesis Hyp_j is then assumed to follow a multivariate normal distribution, so that $\text{Hyp}_j \sim \mathcal{N}(\bar{X}_j, \Sigma_{\text{Hyp}_j})$ with $\bar{X}_j = [\bar{x}_j, \bar{y}_j]^T$. Note that a distribution is computed for each hypothesis. An estimate of the covariance matrix is:

$$\Sigma_{\text{Hyp}_j} = \begin{bmatrix} \sigma_x^{j2} & \sigma_{xy}^j \\ \sigma_{xy}^j & \sigma_y^{j2} \end{bmatrix} \tag{4.17}$$

with:

$$\begin{cases} \sigma_x^{j2} = & a_j.\sum_{i \in P_j} w^i (x^i - \bar{x}_j)^2 \\ \sigma_y^{j2} = & a_j.\sum_{i \in P_j} w^i (y^i - \bar{y}_j)^2 \\ \sigma_{xy}^j = & a_j.\sum_{i \in P_j} w^i (x^i - \bar{x}_j)(y_i - \bar{y}_j) \end{cases} \tag{4.18}$$

where:

$$a_j = \frac{1}{1 - \sum_{i \in P_j} (w^i)^2}$$

is a normalizing factor and:

$$\bar{x}_j = \sum_{i \in P_j} w^i x^i;\ \bar{y}_j = \sum_{i \in P_j} w^i y^i$$

are respectively the weighted arithmetic means in the x and y directions.

4.4.3 Consistency Test Based on a χ^2 Test

We estimate the distributions of the map-matched hypotheses and GNSS positions using Gaussian approximations. A test procedure must be set to determine whether the GNSS fix is consistent with the map-aided dead-reckoned hypotheses. If it is not, it is very likely that either one of them (or even both) are faulty. The objective is to measure the distance between the hypotheses and the GNSS fix to assess if they are coherent with each other. The Mahalanobis distance is an interesting metric for this purpose [Li et al., 2018a].

Some considerations must be taken in order to apply it to the current problem. Since the GNSS fix and the means of the hypotheses can be considered independent, the GNSS uncertainty is added to the hypothesis distributions. This is meaningful as the goal is to characterize the distance between the two distributions. The situation is now composed of a GNSS fix (a single position) and hypothesis distributions with modified covariances expressed as:

$$\Sigma_j = \hat{\Sigma}_{gnss} + \Sigma_{hyp_j} \tag{4.19}$$

The Mahalanobis distance between the GNSS and each hypothesis estimate can then be calculated by:

$$D_{M_j}(\bar{X}_j, X_{\text{GNSS}}) = \sqrt{(\bar{X}_j - X_{\text{GNSS}})^T \Sigma_j^{-1} (\bar{X}_j - X_{\text{GNSS}})} \tag{4.20}$$

with X_{GNSS} being the GNSS fix expressed in the local Cartesian frame and $\bar{X}_j$ the mean value of a given hypothesis set.

The Mahalanobis distance is an adequate metric for evaluating the consistency between two uncertain positions by taking into account the covariances of the variables [Bar-Shalom and Li, 1993]. In the following, the notation $D_{M_j}(\bar{X}_j, X_{\text{GNSS}})$ will be simplified to $D_{M_j}(\bar{X}_j)$ for a better readability as the same GNSS fix is used for every hypothesis for a given time step. $D_M(X)$ will be used when referring in general to any hypothesis.

4.4.3.1 Detecting an Inconsistency

An interesting property of the Mahalanobis distance is that if X follows a centered normal distribution (i.e. $X \sim \mathcal{N}(0, \Sigma)$), then $D_M^2(X) \sim \chi_p^2$ (with $p = 2$ in this case, as the current localization problem is in 2 dimensions). This does not hold if the estimate X is biased or inconsistent. This principle can be applied to develop a method aiming to detect faults in the positioning system. The Mahalanobis distance defined by (4.20) can be used as a decision variable in a χ^2 test in order to determine the presence of an incoherence, as presented in Section 4.2.1.1. If it is the case, a bias for instance can be present in the distribution and thus be detected by the statistical test.

Algorithme 4.3 Metric calculation based on Mahalanobis Distance, determining the hypothesis coherence.

```
1: function COHERENCE TEST(N_GNSS, N_Hyp)
2:     for each hypothesis j do
3:         Compute D²_{M_j}(N_{Hyp_j}, N_GNSS)          ▷ see Section 4.4.3
4:         if D²_{M_j} < χ² critical value then          ▷ χ² test, see Section 4.2.1.1
5:             CoherentHypos ← j
6:         end if
7:     end for

8:     return CoherentHypos
9: end function
```

The principle is as follows: let H_0 be the null hypothesis that *"given the GNSS fix X_{GNSS}, the position estimate X provided by the SMMAL-PF is not affected by any fault"*. If H_0 is verified, then $D_M^2(X)$ follows a centered χ_2^2 distribution.

To assess H_0, a χ^2 test is performed using the squared Mahalanobis distance as the decision variable. The critical value T is set in order to correspond to a defined significance level, equivalent in this case to a probability of false alarm p_{fa} (the probability of detecting a fault when none is present). For instance, for probability $p_{fa} = 0.01$, the corresponding χ^2 critical value is $T = 9.2103$ (value given for 2 degrees of freedom).

$D_M^2(X)$ is compared to the critical value T: H_0 is considered to be verified if $D_M^2(X) < T$, and rejected otherwise, validating the hypothesis H_1 that "*a fault is detected*". An overview of the method presented in this section is described in Algorithm 4.3. It computes a decision variable using the distributions computed in Section 4.4.2.

4.4.3.2 Testing the Hypotheses

The consistency of the map-matching results is tested at each iteration of the filter. The previous χ^2 test is applied separately to each hypothesis Hyp_j using the same GNSS fix, in order to flag the hypotheses not satisfying H_0 as presenting a fault. This defines the alternative hypothesis H_1 as: *"either the hypothesis is wrong or the GNSS fix is false"*. The set of coherent hypotheses is then determined for the current time step by only keeping the hypotheses verifying H_0.

It is important to note that the coherence of the situation takes into account the entire positioning system. Even if the Mahalanobis distance is computed taking the GNSS fix as reference, the test does not assume that the latter should be considered as correct. This means that it is possible to classify a correct matching hypothesis as inconsistent, due to GNSS errors (see Fig. 4.8). The matching situation will then be qualified as inconsistent but the source of the problem will not be determined.

In other words, the χ^2 test indicates the coherence of the GNSS fix in its globality with the map-matching algorithm and not the correctness of the matching itself (nor of the GNSS): eliminating a hypothesis does not necessarily means it is incorrect, but that the situation does not allow the positioning system to be trusted, hence the inconsistency of the situation.

After this test, the Use/Don't Use test (see Section 4.3.3, Algorithm 4.1) can be

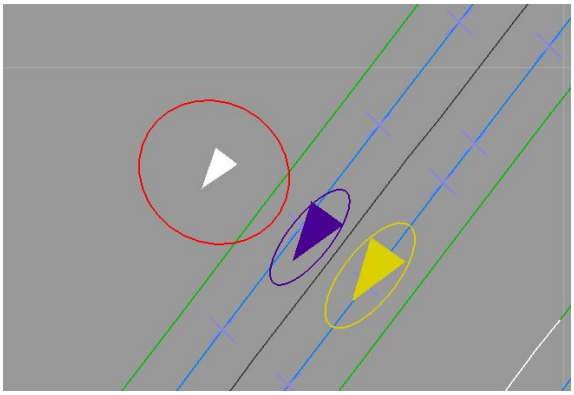


Figure 4.8 – Flagging a correct matching hypothesis as inconsistent is possible if the GNSS position present a large bias. In the situation depicted here, both hypotheses (in purple and yellow) are eliminated as the GNSS (in white) present a strong error. This is an instance of the case depicted in Fig. 4.4b.

applied on the remaining hypothesis set which only contains coherent hypotheses.

4.4.3.3 Results

The same datasets as the previous chapter have been used (see Section 3.4.1 and Appendix B). To study the χ^2 test performance, repetitive executions have been performed using the open sky dataset. This allows to characterize the behavior induced exclusively by the hypotheses, as the GNSS error is limited during this travel. The filter is configured in the same manner with 1000 particles and an HPL value of 5 m.

Probability of False Alarm Given the chosen p-value, the test is expected to have a probability of false alarm (p_{fa}) of 0.01. To verify this property, the results of the χ^2 test applied to the hypothesis corresponding to the ground truth is studied. The situation is flagged as false alarm when the ground truth is eliminated by the test. This part only studies the χ^2 test. After 400 repetitions (with $3,008$ tests each, one for each time step for a total of $1,203,200$ tests), the test did not raise any false alarm: the p_{fa} is respected (the filter performs even better). This can be explained by the HPL restricting greatly the particle set, the hypotheses are thus always in accordance with the GNSS. This again illustrates the importance to choose a correct HPL value for an optimal performance.

This is confirmed by choosing a larger HPL: when set to 15 m, the filter leaves more freedom to the particles to evolve, leading to hypothesis distributions farther away from the GNSS. After 400 runs, the algorithm yields $p_{fa} = 0.02$.

On the complete travel (mixed peri-urban and open sky environments), using the same definition as previously, the p_{fa} after 400 repetitions is 0.07. As exposed previously, receivers are most of the time over-confident, returning an underestimated covariance matrix. Part of these false alarms are caused by an erroneous GNSS with a too small covariance (i.e. a correct hypothesis eliminated due to a large GNSS bias).

In order to characterize exclusively the test performance, independently from sensor performance, a false alarm rate must be calculated, removing the occurrences

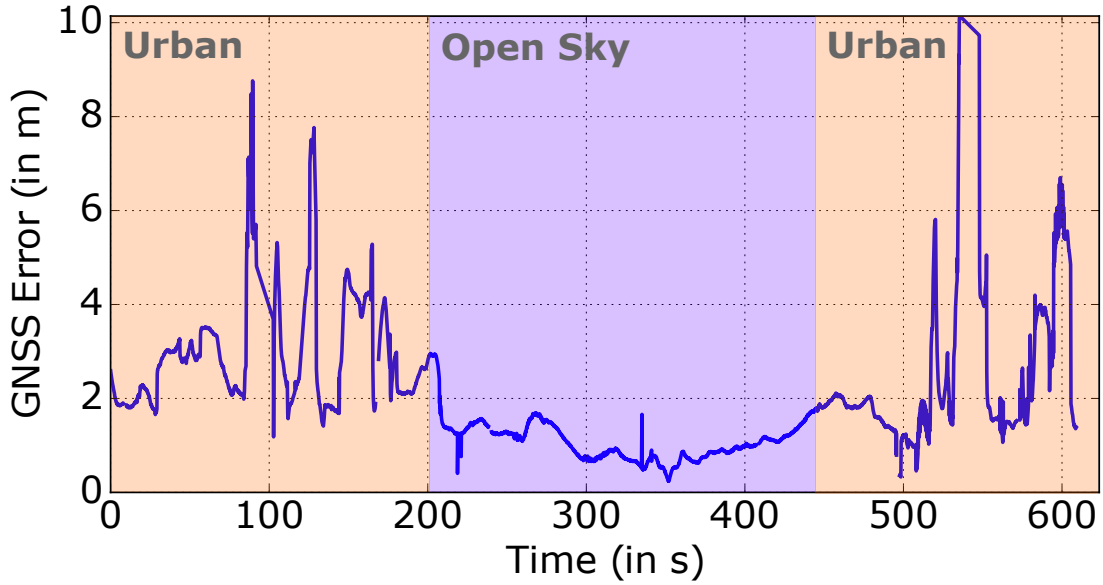


Figure 4.9 – GNSS error evolution along the mixed travel, measured using an embedded IMU as reference. The two types of environment are easily distinguishable: peri-urban presents higher and more unstable error, while open sky conditions yields stable smaller error.

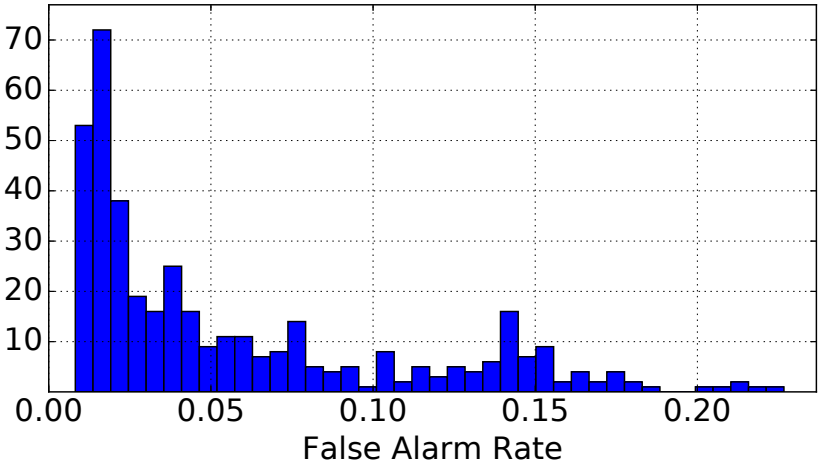
caused by GNSS errors. Fig. 4.9 shows the GNSS error along the travel. It has been computed using ground truth data, gathered by an IMU. It is reasonable to consider it as responsible for a part of the rejections, at least during the peri-urban parts (e.g., at $t = [90, 150]$) in view of the large errors that can occur in this type of environment. In this case, rejecting the correct matching hypothesis cannot be considered as a false alarm, as it is due to a fault on the GNSS receiver side (for instance, a situation such as depicted in Fig. 4.8).

With this consideration, a corrected p_{fa} can be computed: Fig. 4.10 shows different statistics, such as the distribution of the false alarm rate over all iterations (see Fig. 4.10a), removing the rejections with a GNSS error greater than 3.5 m (threshold corresponding to a lane width). The values are visibly concentrated around $p_{fa} = 0.02$, which is the median value of the set (see Fig. 4.10c). In other words, the false alarm rate was smaller than 0.02, 50% of the time during the study (see cumulative histogram, Fig. 4.10b).

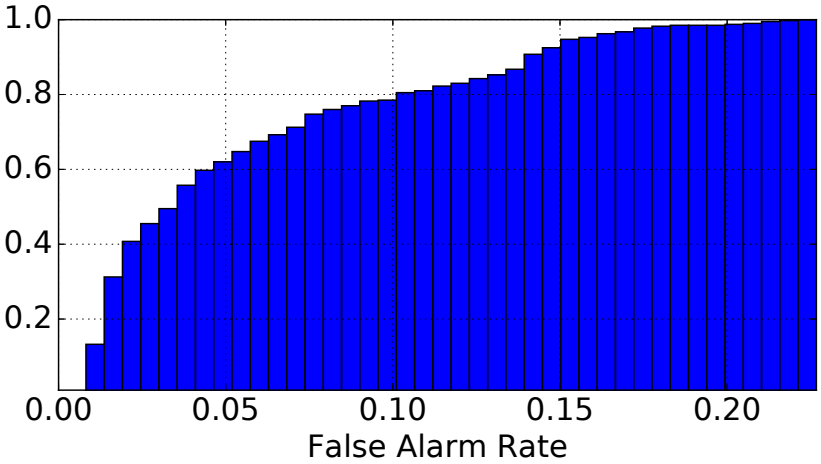
Despite experimental conditions during which the GNSS covariance matrix is often under-estimated, the algorithm presents a p_{fa} of the same order of magnitude as expected from the chosen χ^2 test p-value.

General System Availability The following results are based on the entire data returned by the map-matching algorithm, i.e. considering all hypotheses. They focus on the general performance of the method in terms of fault rejection, with a study of the system availability. The mixed travel is used, as presenting various events of interest. The following data consider a single execution of the filter.

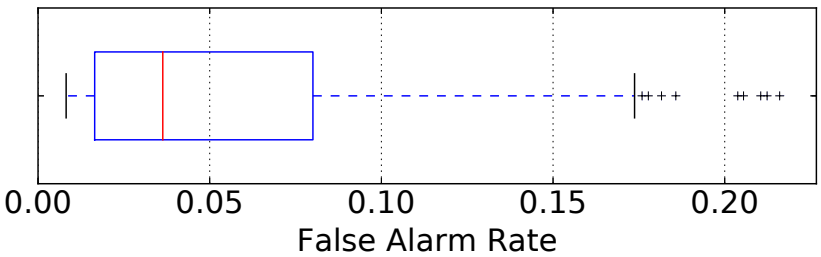
The weight of the different hypotheses are plotted in Fig. 4.11. The 0.1 weight threshold is also denoted as a yellow line. This will be used to perform a first hypothesis rejection, by removing the hypotheses with a weight lower than the threshold.



(a) Histogram of the false alarm rate.



(b) Cumulative histogram of the false alarm rate.



(c) Box plot of the false alarm rate.

Figure 4.10 – Statistics about the false alarm rate distribution for the mixed travel, eliminating epoch with high GNSS error.

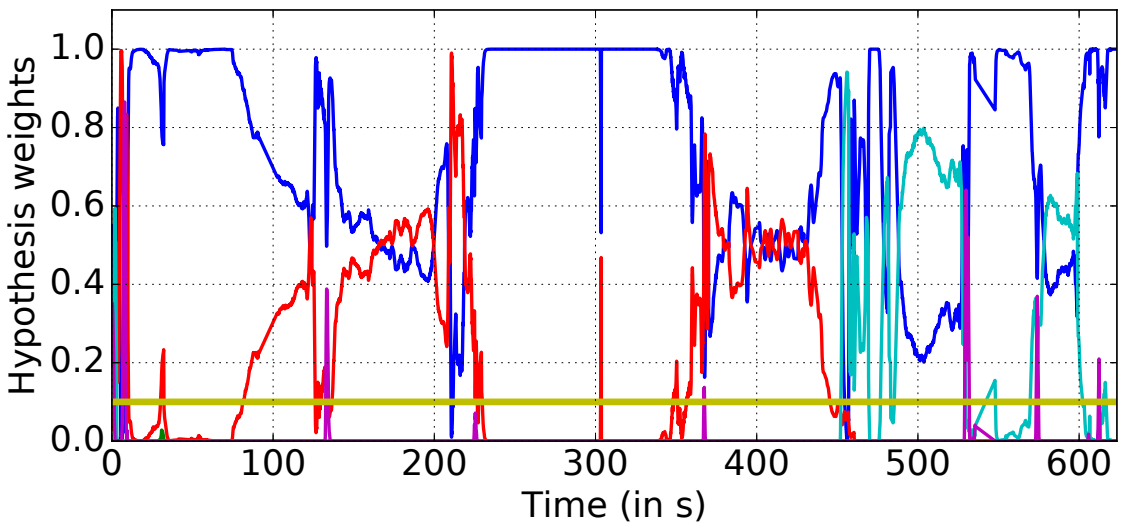


Figure 4.11 – Evolution of the hypothesis weights. Two hypotheses are prominent most of the time, reflecting the nature of the traveled roads (2-lane roads). The ground truth is drawn in dark blue, while the other colors denote different adjacent hypotheses.

The correct hypothesis is drawn in dark blue. The car is driving on the right lane except for 2 lane changes to the left one. The other relevant hypotheses are the one drawn in red and cyan, which represents the adjacent left lane during the travel. This hypothesis is the most ambiguous when driving on a two-lane road. Combining their weights represent 99.9% of the total weight, 95% of the time (see Fig. 4.12). For more clarity, in the following we will combine all incorrect hypotheses into a single one.

Fig. 4.13 shows the evolution of the Mahalanobis distances of all hypotheses. The χ^2 test rejection threshold is drawn in bold yellow, corresponding to $p_{fa} = 0.01$. These data will be used to perform the second hypothesis rejection by removing the hypotheses not verifying the test (D_M^2 higher than the threshold). For reference, the corresponding GNSS error throughout the test has been presented previously in Fig. 4.9.

The statistics for the combined χ^2 test and the weight rejections are shown in Table 4.1. The rejections of the correct matching hypothesis and of the incorrect ones are presented separately (the two lines). The first two columns present the rejections classified by reason, while the last one shows the combined rejection percentage. Note that the latter is not the sum of the two former as an hypothesis can be rejected by both tests simultaneously.

The table shows that the correct matching has been eliminated 2.61% of the time due to a failed χ^2 test and 1.07% of the time due to a low weight, for a total of 3.37% of rejection (i.e. the hypothesis has been eliminated 0.31% of the time for both two reasons simultaneously).

These percentages are relative to the total number of time steps during the test. On the "Incorrect hypothesis" line, the percentage in parentheses represents the statistic calculated relatively to the number of time steps during which an hypothesis other than the correct one is present. In other words, it only takes into account the time steps when an "other" hypothesis exists and thus, can be eliminated. This

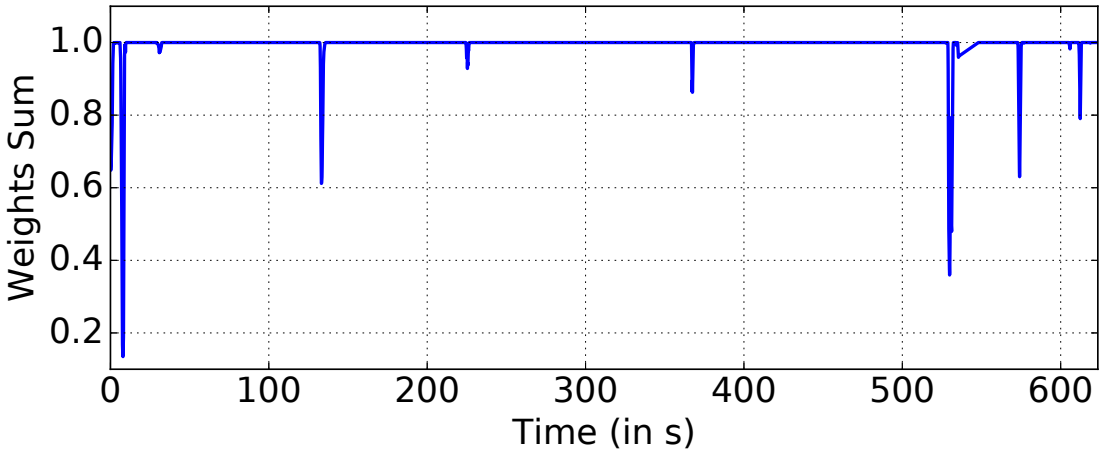


Figure 4.12 – Summing the two highest weights shows that the filter considers two main hypotheses most of the time.

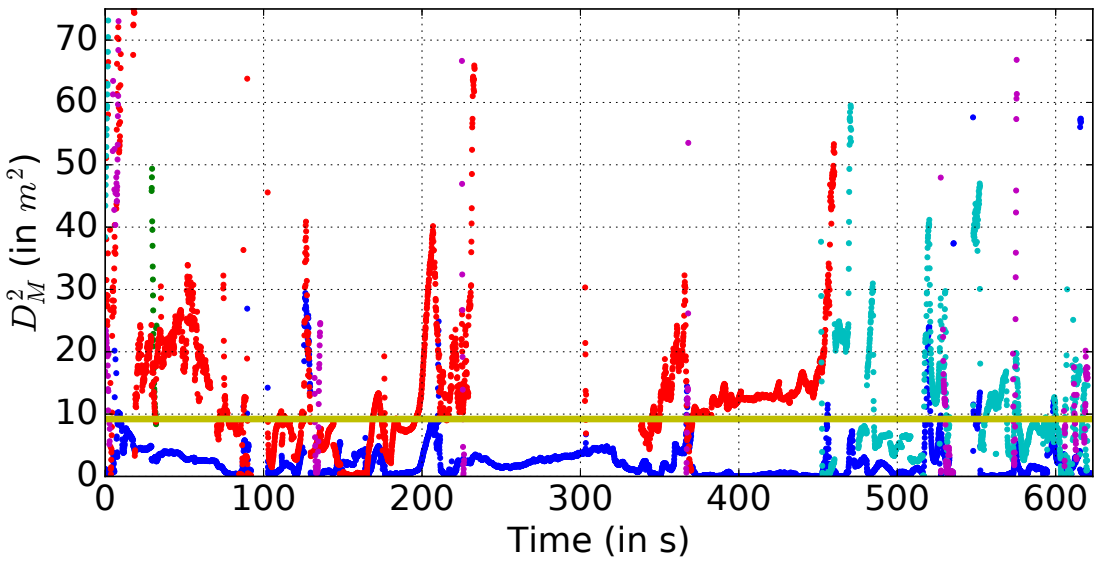


Figure 4.13 – Squared Mahalanobis distances for the matching hypotheses (the correct hypothesis shown in dark blue). The yellow line denotes the critical value for $p_{fa} = 0.01$.

Table 4.1 – Rejections based on χ^2 test and weight threshold. Percentages are calculated relatively to the total number of time steps.

% rejection	χ^2 test	Weight based	Combined
Correct matching	2.61	1.07	3.37
Incorrect hypotheses	46.66 (58.01)	43.32 (53.85)	71.72 (89.16)

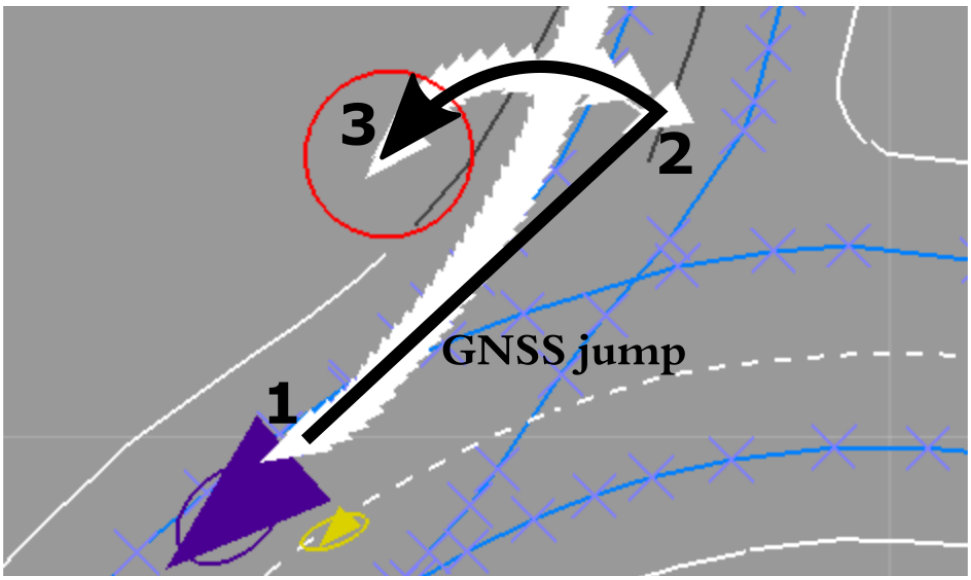


Figure 4.14 – Screenshot of the situation at $t = 127\,s$. A biased GNSS position causes the two matching hypotheses, including the correct one, to be rejected.

statistic is therefore more representative of the test efficiency to eliminate the incorrect hypothesis.

Both tests are effective in removing the wrong hypotheses, with however some unavoidable false alarms. Combining the two tests improves the rejection rate of the incorrect hypothesis, reaching 89.16% while limiting the false alarm, rejecting only 3.37% of the correct hypothesis. Note again that rejecting the correct matching only means that the positioning system as a whole should not be trusted, and that can be caused by a GNSS error. This situation arises during the experiment at $t = 127\,s$. The situation is described in Fig. 4.14. This screenshot of the actual program running shows that the rejection of the correct matching (violet, larger triangle) at this time step is due to a GNSS error. The GNSS fix jumps back about 10 m (depicted by the arrow between points 1 and 2). The current fix (point 3) is affected by a GNSS error and is not coherent with any of the matching hypotheses that are not affected, relying mostly on odometry. As a consequence, no hypothesis remains in the solution set at that time step after the rejection tests: a peak is visible in Fig. 4.13 where both Mahalanobis distances increase drastically (exceeding 4 times the test threshold).

Fig. 4.15 summarizes the different causes of rejection. The tests are executed at each filter iteration. Rejections of the correct hypothesis and of the other are presented in two separate graphs. Each line represents a different cause (cf. legend): the top two lines of each graph represent respectively the χ^2 test and the weight rejections. The bottom line aggregates the two causes, showing the overall rejections of the given hypothesis. The third graph shows the two aggregations side-by-side:

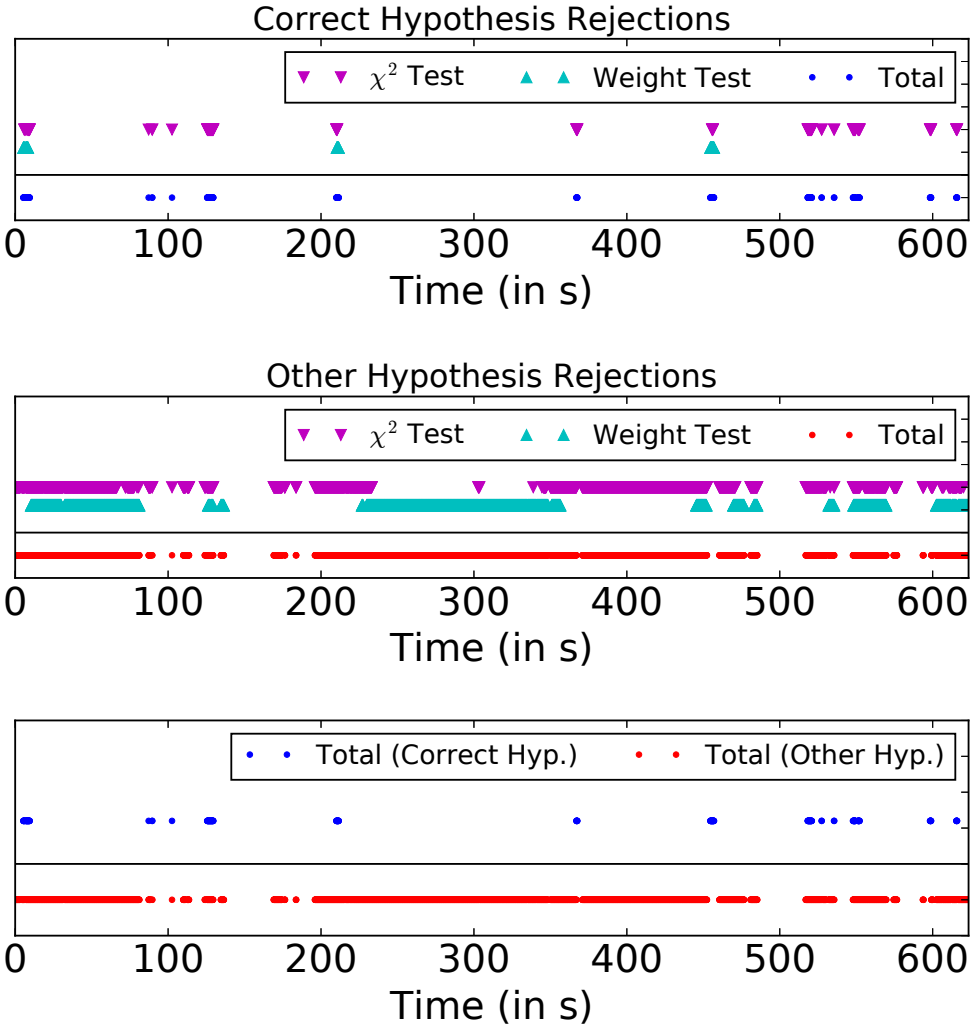


Figure 4.15 – Rejection classification by cause, for the correct matching and the others. A point indicates that the corresponding hypothesis was rejected at this time step for the given reason.

it reaffirms that the tests eliminate more frequently the incorrect hypotheses (true positive), as seen in Table 4.1.

Fig. 4.16 shows the result of the "Use/Don't Use" classification determined from the previous data. Table 4.2 shows that the algorithm has an availability of 69.01%, defined as the amount of time during which the filter classifies the situation as usable, isolating the correct matching hypothesis. For 30.34% of the time the algorithm flags the positioning system as "Don't Use": a discrepancy between the map-aided dead-reckoning and the GNSS fix is detected, preventing the system to be trusted. Finally, the misdetection rate is only 0.65% (the filter classifies an incorrect position as usable). Note that false alarms (the rejection of the correct hypothesis) are not meaningful to consider: as the algorithm is designed to prioritize integrity, it prefers to be pessimistic, classifying a good situation as "Don't Use" rather than the opposite. The false alarm event is thus included in the "Don't Use" category by design, only affecting the availability of the system.

To provide more context, Fig. 4.17 shows the number of hypotheses remaining in

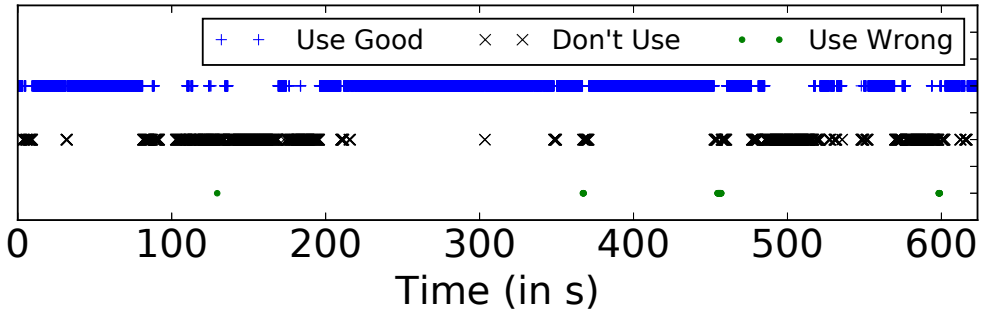


Figure 4.16 – "Use/Don't Use" decisions by the consistency test. It can be seen that misdetections ("Use Wrong") are rare. Positioning system availability corresponds to "Use Good".

Table 4.2 – "Use/Don't Use" classification rates, expressed in percentage of the total number of filter execution steps.

	Don't Use	Correct Use	Incorrect Use
%	30.34	69.01	0.65

the solution set after the tests. The "Use" classifications corresponds naturally to when a single hypothesis remains. Studied together with Fig. 4.9, the occurrences where none remains mostly coincide with peaks in GNSS errors. This correlation is confirmed by the very sporadic nature of the both phenomena. The other "Don't Use" not caused by GNSS errors are due to ambiguities, i.e. multiple hypotheses.

4.4.4 Consistency Test Based on the Kullback-Liebler Divergence

The Mahalanobis distance presented previously is a good metric candidate as the situation meets requirements allowing it be used in a χ^2 test. Other measures are available for estimating the difference between distributions. One that also has been studied during this research is the Kullback-Liebler Divergence (KLD) [Kullback and Leibler, 1951].

4.4.4.1 Definition

The KLD is notably used in information theory [Akaike, 1998] as a measure of the dissimilarity between two distributions P and Q. It characterizes the "surprise" of using Q to represent P, i.e. the difference of behavior between the two distributions. P is typically considered as the available data, for example from samples (i.e. posterior probability distribution) and Q is the model (i.e. prior probability distribution)

The KLD is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx \tag{4.21}$$

with $p(x)$ and $q(x)$, the probability density functions of P and Q respectively. It is important to note that it is a divergence, as the KLD does not respect some

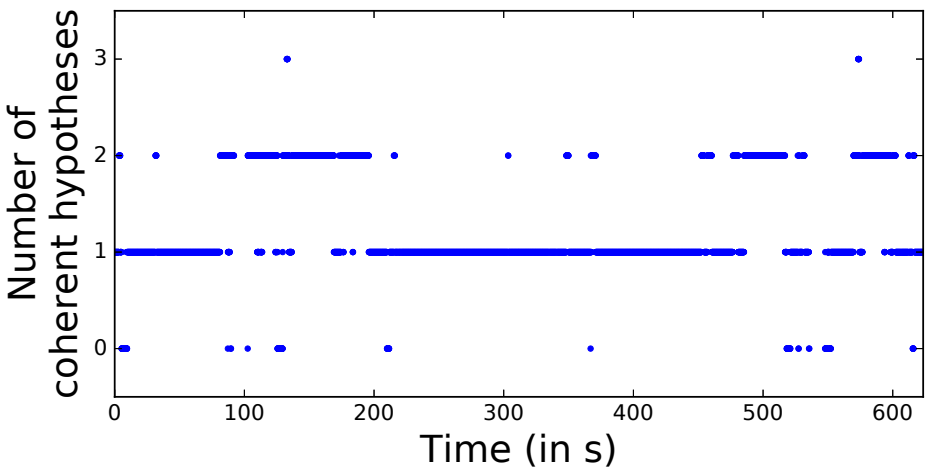


Figure 4.17 – Evolution of the number of coherent hypotheses contained in the final set after the coherence tests.

properties to be considered as a metric: it is asymmetric and does not verify the triangle inequality.

KLD gained popularity in robotics for its use in conjunction with information filters [Al Hage et al., 2017]. It has also been used in particle filtering to improve the sampling of the particles (KLD-sampling [Fox, 2003]) by continuously adjusting the number of particles to the underlying problem.

With Gaussian distributions ($P \sim \mathcal{N}_0(\mu_0, \Sigma_0)$ and $Q \sim \mathcal{N}_1(\mu_1, \Sigma_1)$), the KLD can be rewritten as [Duchi, 2014]:

$$D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left[(\mu_1 - \mu_0)^{\mathrm{T}} \Sigma_1^{-1} (\mu_1 - \mu_0)\right] + \frac{1}{2}\left[\mathrm{trace}\left(\Sigma_1^{-1}\Sigma_0\right) + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) - k\right] \tag{4.22}$$

with k, the state dimension. This expression isolates two components of the divergence, regrouped in different brackets in (4.22): the first can be identified as the squared Mahalanobis distance between Q and μ_0 (see (4.20)).

Both the Mahalanobis distance and the KLD are examples of Bregman divergences [Banerjee et al., 2004], the former is generated by the convex function:

$$F_{\mathrm{MD}}(X) = \frac{1}{2} X^T \cdot \Sigma \cdot X. \tag{4.23}$$

In this case, the Mahalanobis distance considers the mean difference $(\mu_1 - \mu_0)$ and a single covariance matrix $\Sigma = \Sigma_1^{-1}$. It is therefore a measure of the divergence between the two distributions mean, weighted by the model's covariance.

The KLD can thus be seen as an extension of the Mahalanobis distance. The second part of (4.22) is also the expression of a Bregman divergence, for the covariance matrices, using the convex function:

$$F_{\mathrm{KLD}}(X) = -\log(\det(X)). \tag{4.24}$$

This extending component expresses the divergence between the covariance matrices of the two distributions. Using the KLD seems natural in order to add more

importance to the covariances, compared to the method developed with a Mahalanobis distance alone (see Section 4.4.3).

The calculation takes into account the distributions separately, considering one as data and the other as model. As developed in Section 4.4.3.2, there is no preference between the map-matching solutions and the GNSS fix to act as a model. To reflect this, the half-sum can be used instead, additionally making the metric now symmetric.

$$\tilde{D}_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2} D_{\mathrm{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) + \frac{1}{2} D_{\mathrm{KL}}(\mathcal{N}_1 \| \mathcal{N}_0) \tag{4.25}$$

4.4.4.2 Results

KLD has been tested on the travel previously used to test the Mahalanobis distance, in order to compare both results based on the same data. Let $P_{\mathrm{Hyp}}, P_{\mathrm{GNSS}}$ be the distributions (supposed to be Gaussian here) representing respectively the hypotheses and GNSS. Fig. 4.18 shows 3 KLD variations:

- the first one is $D_{\mathrm{KL}}(P_{\mathrm{GNSS}} \| P_{\mathrm{Hyp}})$, taking the hypothesis distributions as reference (i.e. prior distribution);
- the second is $D_{\mathrm{KL}}(P_{\mathrm{Hyp}} \| P_{\mathrm{GNSS}})$, taking the GNSS distribution as reference;
- the third is the half-sum $\tilde{D}_{\mathrm{KL}}(P_{\mathrm{Hyp}} \| P_{\mathrm{GNSS}})$.

Each different hypothesis is represented in a different color, the blue line denoting the correct matching hypothesis. The results show that compared to the Mahalanobis distance, the KLD returns a very large range of values (note the different y-axis scales). The top figure shows that at $t = [400, 450]\,s$, $D_{\mathrm{KL}}(P_{\mathrm{GNSS}} \| P_{\mathrm{Hyp}})$ takes a value of 1000 for the red hypothesis, which is 100 times the value obtained with the Mahalanobis distance.

These two can be directly compared as the KLD asymptotically follows a χ^2 distribution with $\frac{1}{2}k(k+1)$ degrees of freedom [McDonald, 1974; Rencher, 2002], the two measures are then expected in the same order of magnitude (the critical values are respectively 9.210 and 11.345 for 2 and 3 degrees of freedom). The half-sum has naturally the same characteristics. Note that both graphs are truncated as the KLD values reach levels that would render the graphs unreadable.

The half-sum range is however greatly reduced, thanks to the influence of second divergence $D_{\mathrm{KL}}(P_{\mathrm{Hyp}} \| P_{\mathrm{GNSS}})$. When considering the GNSS as prior knowledge, the KLD is clearly more contained, staying mostly in the expected order of magnitude. This is due to the nature of the GNSS covariance being steadier than the hypotheses that are affected by the map data. The remaining of the study will consider only the latter KLD $D_{\mathrm{KL}}(P_{\mathrm{Hyp}} \| P_{\mathrm{GNSS}})$, as it is the only one compatible with the scope of this research.

The KLD behaves closely to the Mahalanobis distance, as shown in Fig. 4.19. The most noticeable difference happens at $t = [230, 320]\,s$: this interval corresponds to the travel on the bypass in open sky. While the Mahalanobis distance for the correct matching (in blue) is small, the KLD exceeds the threshold, eliminating it from the solution set. This is due to the KLD being more sensible to a covariance difference between the two distributions. In this section of the travel, the road is rectilinear causing the particles to spread, forming a very elongated distribution along the road.

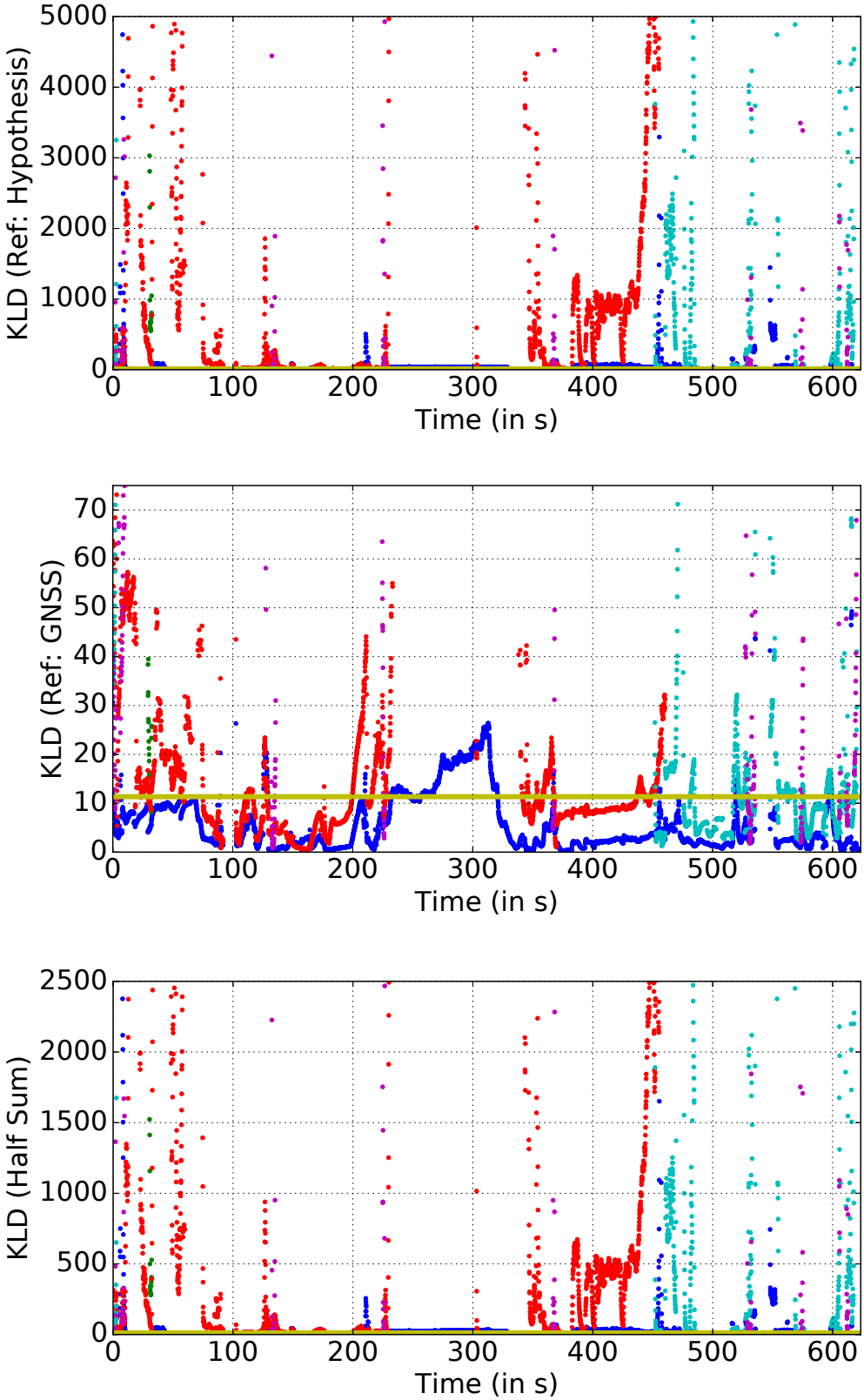


Figure 4.18 – Three variations of the KLD on the same data. From top to bottom, the divergence is computed using, first the hypothesis as reference, then the GNSS. The third KLD is the half sum of the previous two.

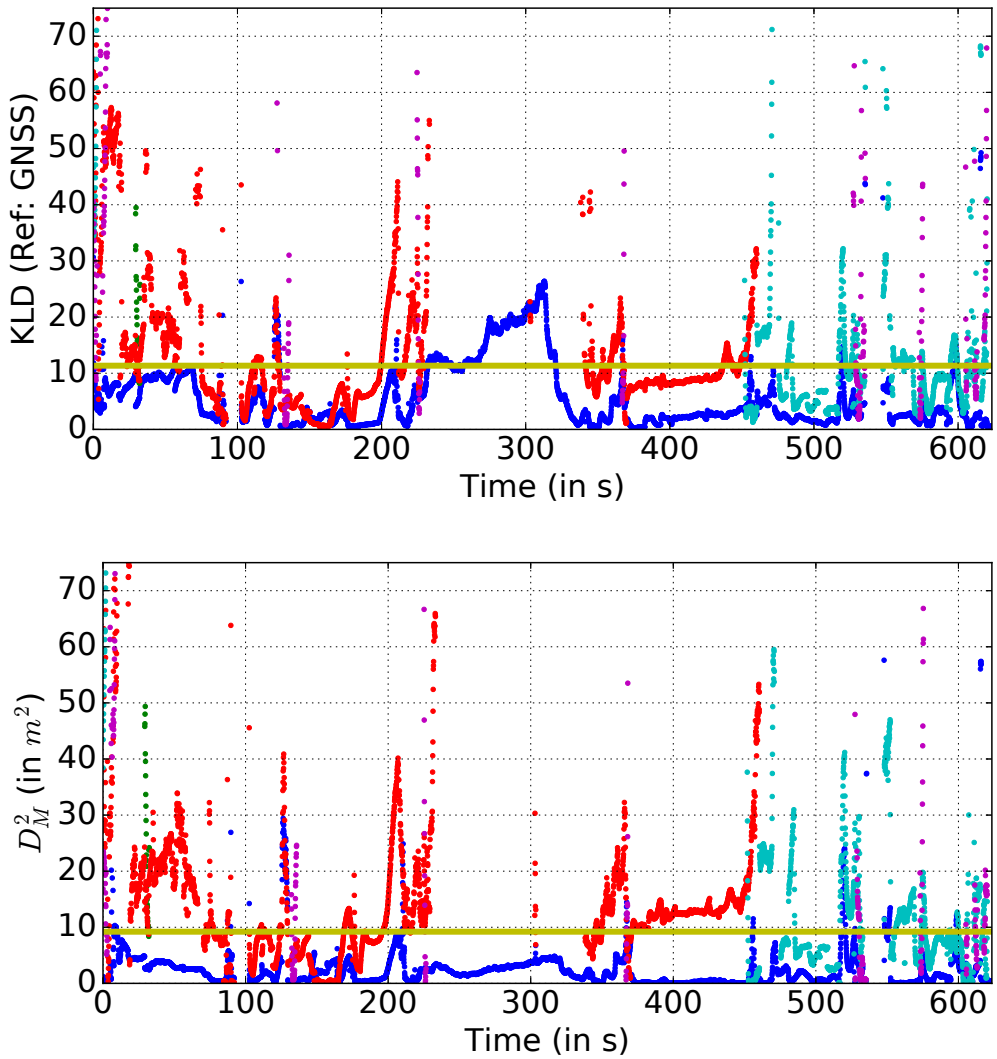


Figure 4.19 – KLD computed with the GNSS as reference, along with the Mahalanobis Distance graph for reference.

Meanwhile, the GNSS reception conditions are very good, causing the receiver to estimate a small covariance. These two very different distribution shapes cause the increase of the divergence. It is confirmed by the quick decrease observed at $t = 310\,s$ corresponding to the beginning of a bend, reducing the longitudinal spread of the particles.

Another difference is present at $t = [380, 450]\,s$, corresponding to a roundabout exit. Two hypotheses are present (left and right lanes) with relatively small covariance, thanks to the high curvature during the roundabout. Both measures for the correct hypothesis are largely under the threshold. However, for the other hypothesis, the Mahalanobis Distance exceeds the threshold, while the KLD stays just below. This is explained by the fact that the Mahalanobis Distance uses the sum of the covariances, estimating a larger distance, while the KLD only uses the GNSS one for its Mahalanobis component. The additional Bregman divergence on the covariance matrix does not compensate this difference, notably because the hypothesis and GNSS covariances are very similar.

This case shows that in certain situations the KLD may be more conservative:

Table 4.3 – Rejections based on χ^2 test using the KLD measure. Percentages are calculated relatively to the total number of time steps.

% rejection	χ^2 test	Weight based	Combined
Correct matching	14.98	1.07	15.89
Other hypothesis	28.32 (35.2)	43.32 (53.85)	56.52 (70.26)

Table 4.4 – "Use/Don't Use" classification rates, expressed in percentage of the total number of filter execution steps.

	Don't Use	Correct Use	Incorrect Use
%	58.37	41.13	0.50

more hypotheses are conserved, especially when the GNSS covariance is small (i.e. in good reception conditions, when the GNSS receiver could be trusted). The positioning availability is thus impacted negatively when using the KLD, as shown in Table 4.3 and Table 4.4.

Using Table 4.1 to compare with the Mahalanobis distance, the correct matching hypothesis is eliminated 6 times more often, while the incorrect one is kept 1.6 times more. It impacts drastically the Use/Don't Use classification by nearly doubling the unavailability of the system and reducing the "Use" classification by a third. Note that by acting relatively more conservatively, a decrease of the misdetections is observed but it is not significant.

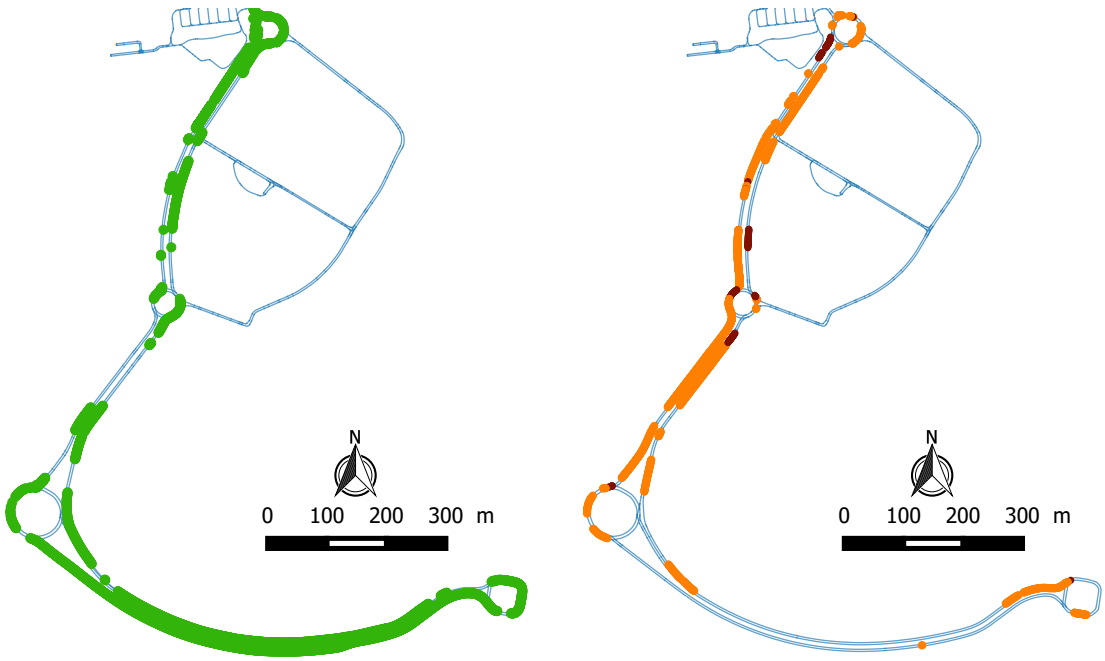
These results show that the KLD is less adapted for the consistency test considered in this research than the Mahalanobis test. It is caused by its focus on covariances difference. While it can be considered as a strength in most cases, it is a weakness in this study as the covariances can be of very different shapes: the hypothesis covariances being influenced by the map while the GNSS one depends on the reception quality of the environment.

4.5 Graphical Illustration of the Results

This section presents the results of the Use/Don't Use classification operated using the χ^2 coherence test, being the most adapted to the requirements of the hypothesis checking as shown by the previous study. Fig. 4.20 shows maps of the test travel, annotated with the Use and Don't Use information (respectively Fig.4.20a and 4.20b). These results are taken after a single travel.

The green areas on Fig. 4.20a denote roads on which the positioning system has been considered usable, i.e. where a unique coherent hypothesis has been isolated. Naturally, the system performs the best on the bypass (southernmost area) with open sky conditions and good lane marking detection.

This last point is crucial for a stable "Use": during the beginning of the travel, in urban conditions, some "Use" classifications are observable but are sporadically interrupted by "Don't Use" tagging (see.Fig. 4.20b). In these area, no lane markings are detected. The particle set is thus not constrained laterally and both lanes of the roads are then considered likely by the map-matching. This creates ambiguous situations where no positioning decision can be taken, they are denoted in orange in Fig. 4.20b. This phenomenon is also visible at the southeast roundabout's exit: the



(a) Areas marked as "Use".

(b) Areas marked as "Don't Use". In red, none of the hypotheses are consistent with the position computed by the GPS receiver. Ambiguous situations are denoted in orange.

Figure 4.20 – Maps representing the results of the χ^2 usability test, on a single test travel (mixed conditions).

road forms a sharp curve so that the camera is unable to detect the lane markings (one of the main weakness of camera-based lane detection). The situation becomes thus ambiguous for the same reason as previously.

Another case of punctual loss of usability happens in the middle of the bypass, during the lane change. This time, lane detection is available but the nature of the event creates a momentary ambiguity. When the vehicle is transitioning from one lane to the other, both hypotheses are equally valid. This ambiguity could however easily be filtered out as a lane change can be detected using the camera information (see Section 3.3.6.2).

Finally, Fig. 4.20b also shows in red the "Don't Use" cases caused by the elimination of all hypotheses. They corresponds to epochs when the GNSS error spikes, affecting negatively the Mahalanobis distances of all the hypotheses and making them all fail the coherence test. This is the expected behavior as the GNSS fix is completely offset in such situation, indicating an issue in the localization system.

Note that some areas are not labeled, this is due to GNSS shortages that cause unavailability of the test, as explained in Section 4.3.3.2. To avoid such unlabeled roads, a solution can be to set all the road to "Don't Use" a priori, and only potentially change to "Use" after the vehicle travels on the given road. This would indicate the system that it cannot trust the positioning system where it has not been tested yet.

4.6 Conclusion

In this chapter, different methods for map-matching coherence assessment have been presented. A deterministic approach has first been developed, providing a map-aided uncertainty level. The MA-HUL can be used to estimate the uncertainty affecting the positioning system. This metric effectively acts as a GNSS error upper bound, but gap between the actual GNSS error and the MA-HUL is too variable to be used reliably. Probabilistic approaches have to be used as they present a higher robustness that overcomes the ease of handling of the deterministic approach.

By considering the different positions as probabilistic distributions, a metric can be computed in order to determine the coherence of the different hypotheses with the GNSS position. In conjunction with the likelihood scores provided by the map-matching, a global usability characterization of the positioning system can be performed. It provides a "Use/Don't Use" indicator for the system at a given moment.

Two probabilistic metrics have been studied: the Mahalanobis distance and KLD, as both of them can be used as decision variable in a χ^2 test. As covariances take an important part in the divergence calculation, the KLD gives too much importance to the shape difference between the two compared elements. In this research, the studied distributions are most of the time quite different from one another. The GNSS most of the time return a circular/ellipsoidal covariance, while the map-matching hypotheses are impacted by the map data, shaping their covariance by following the drivable space. Experimental results showed that the Mahalanobis distance yields better results in terms of system availability and is therefore preferred over the KLD. The chosen solution is therefore the probabilistic approach using the Mahalanobis Distance as coherence metric.

Experimental tests demonstrate the ability of such tests to characterize correctly the usability of the positioning system by isolating the correct matching hypothesis. This information can now be used in order to process the map safely, for reading information but also, even more critically, to write data into the map. A robust matching is essential in order to write in the correct location, preserving the map integrity.

Chapter 5

The Map as a Backbone to Assess Autonomous-Driving Ability Through Successive Passages

Contents

5.1 Introduction . . . 136
5.2 Learning Strategy . . . 136
5.2.1 Online Learning . . . 136
5.2.2 Unsupervised Learning . . . 136
5.3 Adding Information in an Existing Map . . . 137
5.3.1 Need for Reliable Positioning . . . 137
5.3.2 Recording Structure . . . 137
5.3.3 Study of Possible Recorded Information . . . 138
5.3.4 Determination of Usability . . . 140
5.4 Repetitive Travels . . . 140
5.4.1 Accumulation of Data . . . 140
5.4.2 Processing Repetitive Information . . . 141
5.5 Efficient Writing In SQLite Database . . . 141
5.5.1 Direct Insertion Approach . . . 141
5.5.2 Direct Insertion With Two Tables . . . 142
5.5.3 Bulk Insertion . . . 145
5.5.4 Using Transactions . . . 145
5.5.5 SpatiaLite Performance Impact . . . 147
5.6 Results . . . 148
5.6.1 Positioning Usability . . . 148
5.6.2 Marking Detection . . . 149
5.6.3 Vehicle Speed . . . 149
5.6.4 Example of Functionality Usability Determination . . . 150
5.7 Conclusion . . . 151

5.1 Introduction

Making an autonomous vehicle evolve in a completely unknown environment is a very difficult task. Common solutions include SLAM in order to create a knowledge of the environment from scratch. With the high availability of road maps, prior map information can be used to improve this process [Lee et al., 2004].

This learning phase can be done autonomously for small scale systems and environments. But in the case of autonomous vehicles on open roads, an exploratory phase using manual human driving is more realistic. The typical use case is a commuting car, driving every day on the same home-to-work road. This type of repetitive travels represents the majority of a personal car's uses [Krumm, 2012] and is therefore a representative study.

The learning phase allows also the different systems needed for autonomous driving to gather performance information on the targeted travel. The repetitiveness of the travels offers an opportunity to improve the robustness of the learning by gathering redundant information. But to do so, a storing process must be developed. Map uses are often limited to reading operations, providing information to other systems. In the scope of this research, it can be used as an information storing medium, acting as a learning canvas. In order to do this, positioning within the map must be done with high confidence: the previous chapters presented methods answering this need.

This chapter first presents the learning strategy. Then, it introduces the information writing procedure, making use of the map-matching method. The recording structure is presented, along with some example of information gathered from a single travel. Then, the management of multiple travels recording is studied. Finally, the writing performance of the SQLite format is reviewed.

5.2 Learning Strategy

5.2.1 Online Learning

There are several approaches to record information in a map.

A first solution is to keep in memory all the data acquired during a travel and then to store them in the map database afterwards, at the end of the travel. When using this strategy, map-matching can be done in post-processing and can be solved using forward and backward propagation, like a smoother,which helps reducing the ambiguity and improves the accuracy. Nevertheless, storing all the raw data may require a huge amount of memory, and, in case of a software failure, all data are lost. A post-processing strategy has therefore to be handled carefully in order to be sure that no information has been lost.

Another approach consists in doing the learning in real-time while the vehicle is manually driven. The goal is to record information as previously defined, to assess Autonomous Driving (AD) capabilities. This is the approach studied in this chapter.

5.2.2 Unsupervised Learning

Learning can be divided into two paradigms [Hastie et al., 2009].

The first is called "supervised learning" and consists on asking the driver to follow exactly a predetermined trajectory defined at the lane level. Map-matching is in this case simplified a lot since it consists to map-match the estimated pose on a unique polyline. The learning process can then write information at all time. The main drawback of this approach is that it relies on the driver respecting the driving instructions to remove potential ambiguities. This is a bit restricting and may not be accepted by usual drivers. Moreover, each deviation of the driver with respect to the given instructions will induce a recording error.

The second paradigm regroups the "unsupervised learning" methods, where no instruction is given to the driver to follow a predefined path. In this case, the learning algorithm does not have knowledge of the path polyline and therefore has to determine the correct matching with certainty while learning. The map-matching method developed in the previous chapter is then perfectly adapted to this process since it manages in real-time the confidence of the matching process. This is the approach that is studied in the following: an online unsupervised learning.

5.3 Adding Information in an Existing Map

5.3.1 Need for Reliable Positioning

In the scope of this research, maps are considered as correct and accurate (i.e. corresponding to the ground truth). With this consideration, it is important to preserve the integrity of the information it contains. The process of adding information in the map must therefore be taken with caution to not interfere with the original data. Moreover, any added information must be correctly localized in order to correspond to the map accuracy.

An accurate enough localization is therefore mandatory to operate a learning map, which is the geographic information system that is not only used in a *read-only* fashion but in *read-write*. An incorrect matching could be detrimental to the learning phase: misplaced information can cause a bad diagnosis of the expected system performance on a given area. The previous map-matching method has been designed to avoid such events, only indicating a usable position when it has been determined without ambiguity.

The learning map process places itself as client of the Use/Don't Use classification presented in Chapter 4. Writing in the map is only allowed when the positioning system indicates a "Use" situation: every recording must be attached to an element of the map in order to be well integrated into the GIS.

5.3.2 Recording Structure

A learning recording process is composed of the following elements:

- **Geographic coordinates**: As presented previously, a recording is added as an element of the map. Global geographic coordinates are the basic elements to integrate information in the GIS. To stay consistent, the coordinate system used for localization has to be the same as the map's.

- **Timestamp**: Beside the spatial information, the temporal dimension is also an important factor, especially when repetitive acquisitions are made. Each

recording must therefore include a timestamp including the date and hour. This information can be retrieved by GNSS (see Section 2.2.5), the timestamp can thus have the same resolution (generally 1 s or 0.1 s) . The timestamp can be used for example to study the evolution of system performance over time (i.e. different travels), or in different times of the day (e.g., daylight, night time).

- **Map-matched element**: Each recording must be attached to an existing map structure. The map-matching result can conveniently be used for this purpose (it is unique as the positioning system is in a "Use" state). The nature of the map element may vary, depending on the type of information stored. The *link* is a good candidate element as it is the element returned by the matching algorithm and is a good level of discretization. It is also possible to match at a smaller level as the map-matching actually matches at the segment level (as presented in Chapter 3). If the application requires it, this level can be used for a finer recording.

- **Learning information**: Various information can be recorded. The goal being to assess the performance of the different elements composing the vehicle, it depends on the system configuration and the information needed for autonomous navigation. Some examples are developed in the next section.

The storage of these recordings must respect the original map's integrity. In this research, the map is stored as a SQLite database, the learning data are simply written aside of the existing tables. Original map elements can be referred to by using foreign keys, without modifying them. This takes full advantage of the relational database management offered by SQLite. SQL requests can then be designed to regroup directly all recordings related to a given map element (e.g., the matched link) to process them and determine the system performance in a given area.

For a better structure, each recording type is stored in a different table. For example, a table can be created for each individual sensor. This allows a specialization of each recording based on the sensor it represents (see Fig. 5.1): all tables share the same basis (the coordinates, timestamp and matched element) with specialized data fields (the learning information).

5.3.3 Study of Possible Recorded Information

5.3.3.1 Use/Don't Use Classification

The goal of the learning phase is to characterize the different systems required for given autonomous functionalities. In this sense, the "Use/Don't Use" classification is already a result and can thus be integrated in the map. Each "Use" event is recorded (since a "Use" is a reliable matched point) in a dedicated table from which a map can be drawn, showing the areas where the positioning system can be trusted (see Section 5.6).

As this table focuses on the positioning system, additional information about the positioning system can also be recorded. It can be metrics characterizing either the GNSS reception quality (e.g., SNR, number of visible satellites, HDOP) or the map-matching results (e.g., covariance matrix of the hypothesis, number of hypotheses before coherence testing).

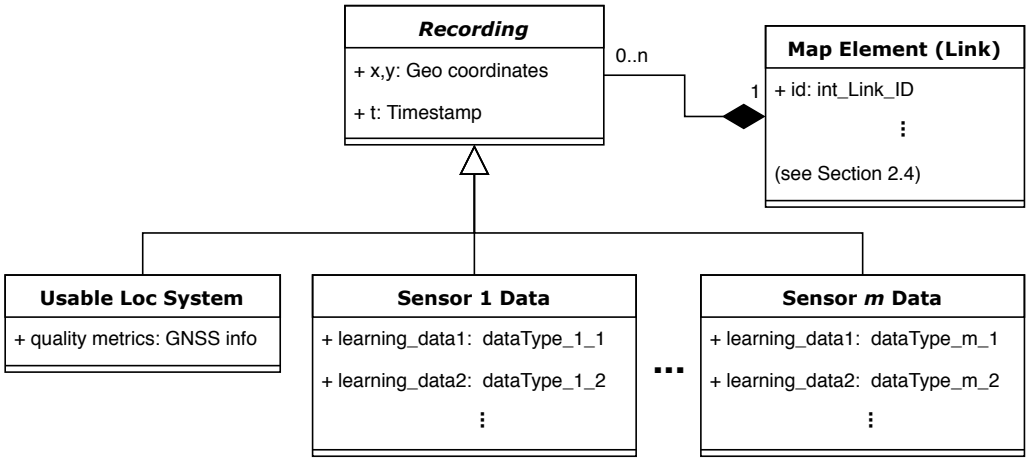


Figure 5.1 – UML exposing the *Recording* classes. A composition with a map element is mandatory. Each learning recording type (implementing their own specific data fields) inherits from a base case *Recording*.

Note that in the case of unsupervised learning, which is applied in this research, only the "Use" events are recordable as the "Don't Use" outcomes are ambiguous. In the case of supervised learning, please note that a map of the "Don't Use" areas could be made, as this information would not be ambiguous in this case.

5.3.3.2 Lane Marking Detections

Lane markings are very important elements for lateral positioning (see Section 3.3.6.2). The original map stores the markings present on the road network, but it does not guarantee the actual detection of the markings by the exteroceptive sensors. External factors can affect negatively their detection such as bad lighting conditions (for camera-based systems) or wet roads. Moreover, lane markings fade with time as painting deteriorates. Recording detection quality (metric returned directly by the camera sensor, describing its confidence about the detection, scaling in ascending quality from 0 to 3) yields therefore important learning data, characterizing the availability of the features in given areas.

5.3.3.3 Others Possible Information

A large number of information types can be interesting to record. As developed previously, it depends on the composition of the system: the learning map serves as an empty canvas for other system to use as a learning data storage; the exact data recorded and its processing will be specific to each individual sensor.

The following lists some commonly available information that may be of interest for autonomous driving:

- Weather/Lighting conditions: using the state of automatic wipers and headlights systems. These data may be relevant for multiple systems and thus included in all recordings. It can for example explain some events such as a degraded lane marking detection (wet road).

- Vehicle's speed: when compared to the speed limitation stored in the map, traffic congestion can be detected. This can hint the path planning to choose

another road if available. This procedure is already applied in modern online guidance systems.

- Landmarks detection: depending on the available sensors, landmarks can be stored. They can be compared to the map data in order to improve the positioning in the same manner as lane markings.

- Pedestrian detection: areas with dense pedestrian presence could also be marked in the map in order to indicate the system a hazardous zone (pedestrians being vulnerable road users, extra care must be taken).

5.3.4 Determination of Usability

Each sub-system (sensor, or group of sensors) must have its own specific requirements for usability (e.g., respecting a quality indicator). For keeping the study generic, the only one studied in details in this research is the positioning system, whose composition can be considered as common (GNSS and map).

A second level of usability describes the autonomous functionality usability, using the results from the previously described sensor usability. This level determines what automation can be performed in which areas. To do this, a list of requirements must once again be created, describing the different functionalities. For instance, lateral-only control (e.g., LKA) may require only the lane marking detection while fully-autonomous driving will require the whole sensors to be available and usable. This list of requirements remains to be determined as it is highly implementation-dependent: no general rule can be presented. The remaining of this section only focuses on the use of the map as learning canvas and the management of the recorded data.

This usability study is performed on a given portion of the road network. As the goal of an autonomous functionality is to perform continuously a certain amount of time or distance, it makes little sense to study the usability punctually. The link level is more adapted, representing a stretch of road with the same properties. In some rare cases, a link could be considered too long to be studied as a whole. This problem can be easily fixed by limiting the maximum link length, creating more successive sub-links. This does not impact the map-matching performance, as presented in Section 3.3.5. A more advanced approach could be dynamically splitting the links to adapt them to the learning step.

5.4 Repetitive Travels

5.4.1 Accumulation of Data

The commuter car driving repetitively on the same roads, each travel allows new recordings to be saved. The data structure defined in Fig. 5.1 allows transparently this type of data accumulation: a recording is uniquely referenced by a position and a timestamp. This means that two recordings can be taken at the exact same geographic position, as long as their timestamps are different: the recording process does not change compared to a single travel. The newest data do not affect the already existing recordings.

5.4.2 Processing Repetitive Information

Multiple strategies can be applied in order to manage the successive acquisitions. Two main visions exist:

- Considering all recording independently: the information from the different travels is all aggregated together to form a single database. The benefit of doing so is the greater number of data considered on a given area. This allows the use of learning method relying on great number of samples (e.g., statistic-based, Bayesian estimation). For example, prior probabilities of certain properties, such as the lane marking detection, can be determined statistically.

- Regrouping the recordings by trajectory (i.e. by travel): this is possible by regrouping them using the timestamps (they are also inserted chronologically in the database). A temporal analysis of the data is then possible: the evolution of the studied property can provide insight if some discrepancy is observable. For example, if the lane markings were detected on a given road on the first travel and not in the second one, the algorithm may check the weather conditions that could be in cause (e.g., rainy conditions).

Depending on the nature of handled information, one or the other strategy is more adapted. For example, elements such as lane marking detection or GNSS reception quality are mostly location dependent, considering each recording independently is then a reasonable strategy. Time of acquisition is also an important factor (e.g., varying satellite constellation in view) and can be taken into account for the learning phase but it does not require to consider the recordings as part of a trajectory. On the other hand, information such as vehicle speed takes its full meaning within a trajectory, expressed as the speed profile of a given travel.

5.5 Efficient Writing In SQLite Database

The SQLite writing performance has been studied using the library version 3.7.6. A test program has been developed to test the repetitive table insertions. Different test cases are studied, representative of the behavior of the learning map execution. The computer used for this study is a mid-range laptop (AMD A8-4500M CPU@1.90 GHz with 8 GB RAM).

5.5.1 Direct Insertion Approach

First the insertion in a single table is tested, calling a SQL request for each insert. This simulates the recording of 50,000 individual learning values, on the fly (i.e. right after acquisition). A dummy table is used with a structure corresponding to a single learning field; a row then contains the minimal quantity of information: a timestamp, coordinates, a link ID and a single data field. The pseudo code (listing the actual SQLite function calls with simplified interface) for this test is given in Algorithm 5.1.

Fig. 5.2 presents the test results. Fig. 5.2a shows the evolution of the SQLite INSERT requests duration. Aside from a noisy variation caused by OS operations during the writing on disk, the execution time stays constant even when the table

Algorithme 5.1 The loop used to test the insertions on the fly of data in a single table.

```
1: function TEST SINGLE TABLE ON THE FLY
2:     for i ← 0, nb_inserts do
3:         stmt ← INSERT INTO test_table VALUES (i_th row)
4:         sqlite3_prepare_v2(stmt)    ▷ Preparing dummy data to write in the map
5:         sqlite3_reset(stmt)
6:         sqlite3_step(stmt)
7:     end for
8: end function
```

Algorithme 5.2 The loop used to test the insertions on the fly of data in two different tables.

```
1: function TEST TWO TABLES ON THE FLY
2:     for i ← 0, nb_inserts/2 do
3:         stmt ← INSERT INTO test_table1 VALUES (i_th row)
4:         sqlite3_prepare_v2(stmt)    ▷ Preparing dummy data to write in the map
5:         sqlite3_reset(stmt)
6:         sqlite3_step(stmt)

7:         stmt ← INSERT INTO test_table2 VALUES (i_th row)
8:         sqlite3_prepare_v2(stmt)    ▷ Preparing dummy data to write in the map
9:         sqlite3_reset(stmt)
10:        sqlite3_step(stmt)
11:    end for
12: end function
```

reaches 50,000 rows. To put this number in perspective, this represents about 14 hours of constant recording at 10 Hz. This constant insertion time may not be verified when the database size reaches several GB. This corresponds to a very high number of recordings, the 50,000 rows only representing about 1.20 MB. In the context of learning map, the writing time can be considered as constant.

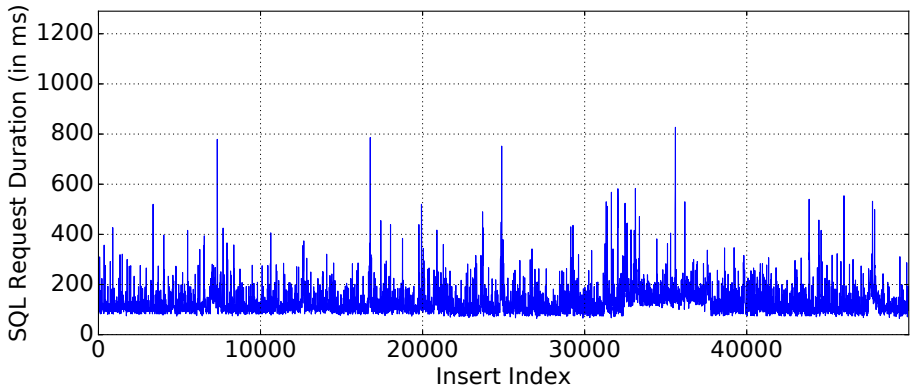
The mean insertion duration is 132 ms, corresponding to a performance of 7.57 inserts per second (insert/s).

5.5.2 Direct Insertion With Two Tables

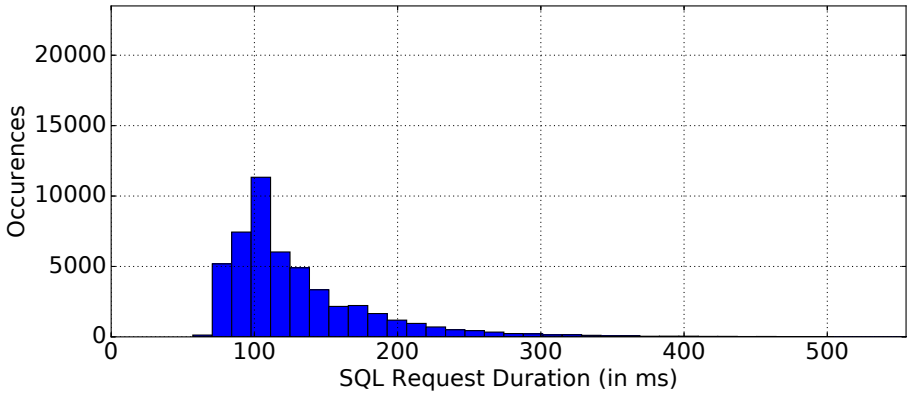
Secondly, multiple tables are used, simulating the recording of different learning data. The case of two tables is studied to assess the impact of swapping between tables. Algorithm 5.2 describes the procedure: instead of writing constantly on a single table, two consecutive inserts write on two different tables. The test then writes 25,000 values in each table.

Fig. 5.3 shows the results. As expected, the request duration also stays constant throughout the test with the same noise (see Fig. 5.3a). A small increase is observable with a mean execution time of 151 ms (6.6 insert/s). However the distribution (see Fig. 5.3b) is more widespread, with 2 visible peaks at around 112 ms and 145 ms.

Writing in two tables adds in average 15% of writing time, but the distribution shows that difference may not be always present effectively. These two insertions

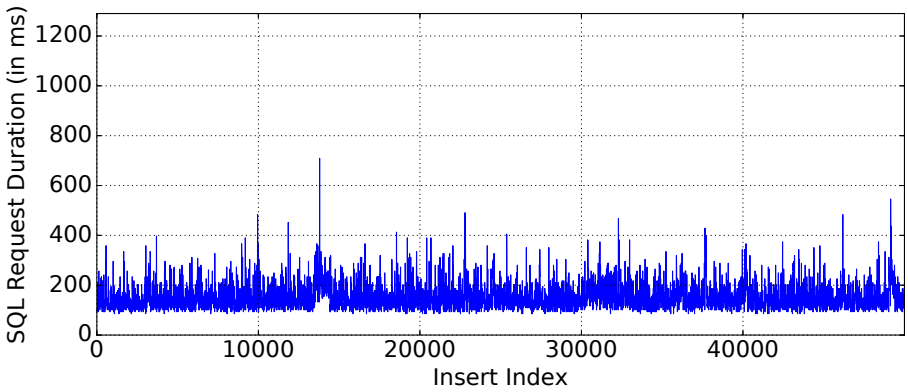


(a) The INSERT execution time remains steady while the table size grows (up to 50,000 rows). The some variation may be caused by disk access and its underlying OS level operations.

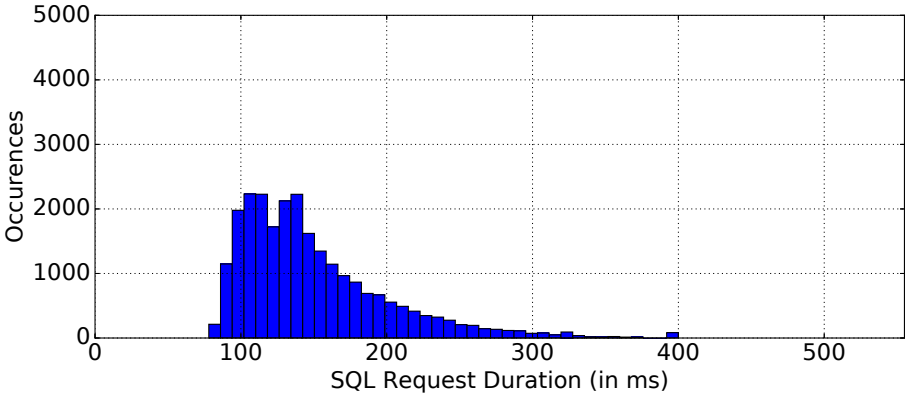


(b) Distribution of the request duration showing values concentrated around the mean value of 132 ms.

Figure 5.2 – SQLite INSERT performance, writing the rows one by one in a single table.



(a) The INSERT execution time remains steady while the table size grows (up to 2 times 25,000 rows). The some variation may be caused by disk access and its underlying OS level operations.



(b) Distribution of the request duration showing values concentrated around the mean value of 151 ms, higher than for a single table.

Figure 5.3 – SQLite INSERT performance, writing the rows one by one in two distinct tables.

Algorithme 5.3 The loop used to test the insertions, 500 at a time, in a single table.

```
function TEST SINGLE TABLE 500 AT A TIME
    stmt ← INSERT INTO test_table VALUES                    ▷ Base of Statement
    for i ← 0, nb_inserts do
        stmt ← append (i_th row)                            ▷ Appending data to the base
    end for
    sqlite3_prepare_v2(stmt)        ▷ Preparing dummy data to write in the map
    sqlite3_reset(stmt)
    sqlite3_step(stmt)
end function
```

Algorithme 5.4 Use of a SQL transaction for efficient insertions.

```
function TEST SINGLE TABLE USING A TRANSACTION
    sqlite3_prepare_v2(BEGIN TRANSACTION)                   ▷ Starting a transaction
    sqlite3_reset(stmt)
    sqlite3_step(stmt)

    for i ← 0, nb_inserts do        ▷ Buffering all statement in the transaction
        stmt ← INSERT INTO test_table VALUES (i_th row)
        sqlite3_prepare_v2(stmt)
        sqlite3_reset(stmt)
        sqlite3_step(stmt)
    end for

    sqlite3_prepare_v2(COMMIT)                              ▷ Commiting the transaction
    sqlite3_reset(stmt)
    sqlite3_step(stmt)
end function
```

methods may thus be considered equivalent. It could then provide a solution to the loss of performance of very large tables: creating a new empty one for real-time data recording and aggregate all of them offline.

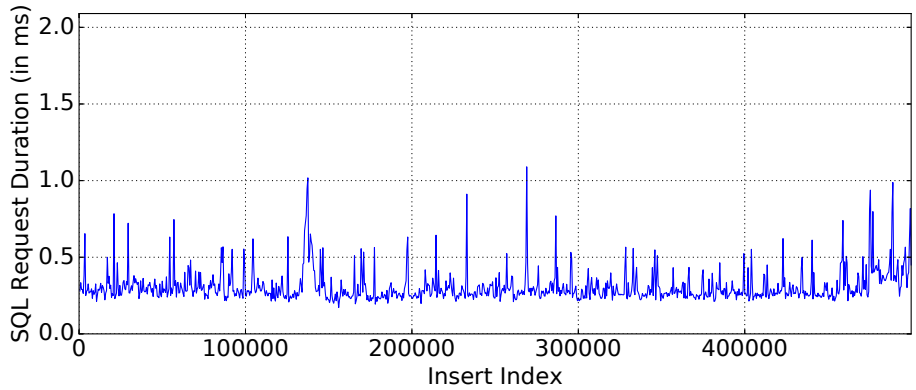
5.5.3 Bulk Insertion

This test regroup a set of INSERT commands into a single SQL statement, as described in Algorithm 5.3. This is simply done by appending successively the rows in a base INSERT statement. This solution is however limited by SQLite maximum number of insert terms that is 500 [SQLite, 2018]. To get enough data, the 500-inserts statement is repeated in order to write a total of 500,000 rows.

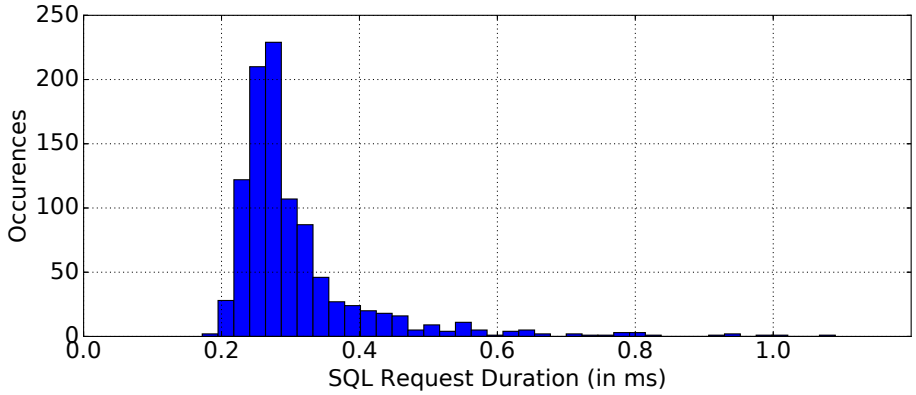
The results presented in Fig. 5.4 are without any doubt better than the previous ones: the mean request time is 0.31 ms, corresponding to 3225.80 insert/s.

5.5.4 Using Transactions

Another solution uses a SQL functionality combining the previous methods: an SQL transaction. It is presented in Algorithm 5.4.



(a) Mean duration of a single insertion. Execution time also remains steady while the table size grows (up to 500,000 rows). The some variation may be caused by disk access and its underlying OS level operations.



(b) Distribution of the request duration showing values concentrated around the mean value of 0.31 ms

Figure 5.4 – SQLite INSERT performance, writing the rows in bulk, 500 at a time, in a single table.

Method	Comments	Duration by insert (in ms)	Performance (in insert/s)
On the fly	Ease of use, direct request	132	7.57
500 bulk request	Need to buffer data before request	0.31	3225.80
Transaction	Ease of use: direct request but delayed execution. Require COMMIT	0.03	34,099.43

Table 5.1 – Mean insert time and corresponding performance for the different insert strategies: transactions outperform the two others.

A transaction is started with a BEGIN TRANSACTION SQL statement (see line 2). This allows multiple SQL statements (see line 5) to be regrouped into the transaction, without being applied to the database yet. All the statements will be applied only on a commit statement (see line 11).

The whole transaction (i.e. from the BEGIN TRANSACTION statement to the end of the commit) for 500,000 rows has taken 14.663 s, that is to say 0.03 ms per row. The performance therefore reaches $34,099.43$ insert/s.

All the previous results are consolidated in Table 5.1 (the multi-table results are omitted for the reason presented in Section 5.5.2).

The same performance issues as in reading operations (see Section 3.3.4) are detected. This is not specific to SQLite but inherent to SQL generally. The best solution is therefore to buffer all writing data and do the writing operation once. It then appears clearly that using a transaction is the most efficient solution and is therefore the preferred solution for writing in the map. It combines the ease of writing of the "on the fly" method with the efficiency of the writing in bulk: there is no need to buffer the data on the application side as the SQL transaction takes care of it.

The transaction also provides atomicity of the operation: either all data are written or none, assuring the integrity of the database in case of power failure for example. To insure a good writing of the learning data, the best strategy would be to commit the inserts regularly, creating multiple consecutive transactions. Either a fixed time interval or a given number of inserts are both valid criteria for a commit.

5.5.5 SpatiaLite Performance Impact

The previous tests considered plain SQLite databases, using two separated columns for 2D-coordinates. This section studies the impact of the Spatialite library on the writing performance. The two columns X and Y have been replaced by a single SpatiaLite 2D-geometry column. The transactional method remains the chosen method.

Tests show that in the same conditions, a $500,000$ INSERTs in a single table with a geometric row takes 113.43 s to be executed. This corresponds to 0.23 ms per insert, yielding a performance of 440.78 insert/s. Compared to the non-SpatiaLite version (see Section 5.5.4), the performance is degraded, decreasing by a factor of 7.56, staying however at a high enough level for most situations. An application

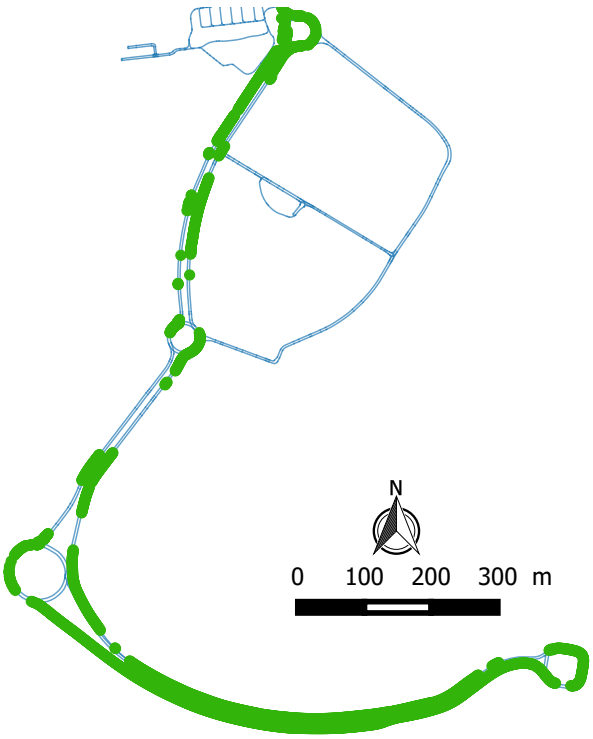


Figure 5.5 – Map representing the data from the "Use" SQLite table, showing the area where the positioning system has been determined usable.

may fall back to the separate X and Y fields in case it requires this very high insert rate, a post-processing could then be done to convert them back to a SpatiaLite geometry.

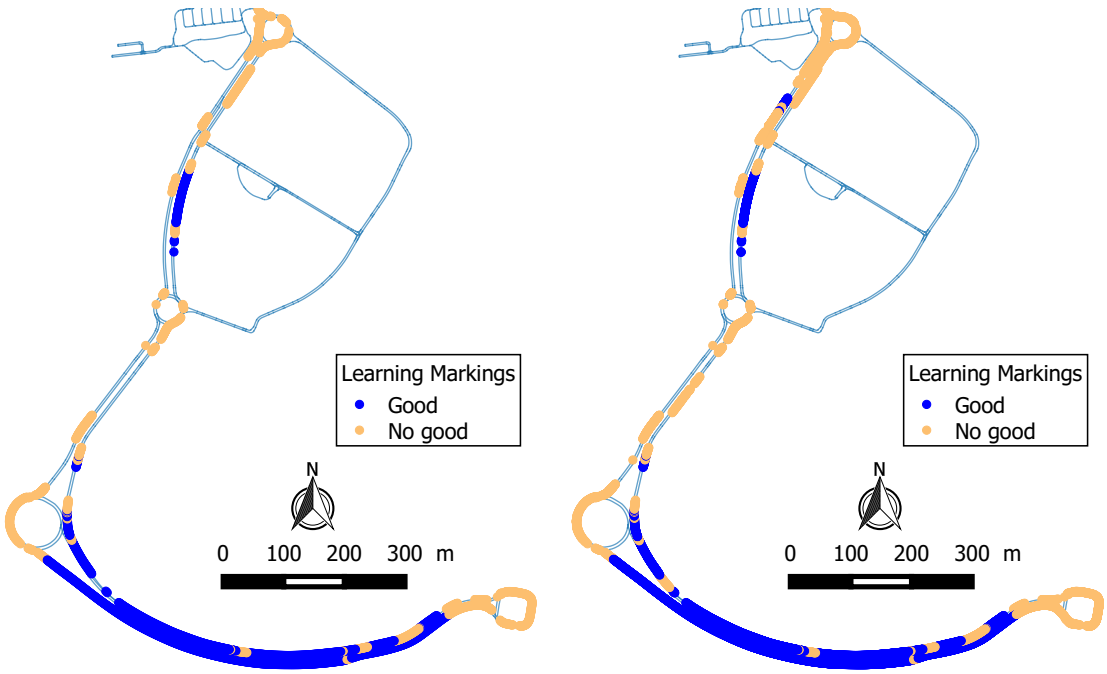
Memory wise, SpatiaLite adds an overhead that increases the size of a recording. At 86 bytes per row (containing a single learning data), it stays reasonable. For instance, for a recording at 10 Hz in a vehicle traveling at 50 km/h, this represents 60 kB/km for a single recording. Note that this does not scale linearly with more stored information: the main overhead of the recordings are the fields contained in the *Recording* structure of Fig. 5.1. Common for all recordings at a given time step, they can be grouped and stored only once: each learning table simply referencing it by a foreign key. This allows an efficient storage of the learning data in a contained size.

5.6 Results

5.6.1 Positioning Usability

Fig. 5.5 shows the resulting usability map of the positioning system after a single travel. This information is important for two reasons: firstly, it characterizes the quality of the positioning on the considered travel and secondly, it provides the areas where learning data can be gathered and saved in the map. During this test, the positioning systems is considered usable 57.5% of the time (3,439 out of 5,986 time steps).

As developed in Section 5.3.1, the data writing can only happen when the position on the map is determined without ambiguities, i.e. when the positioning system



(a) After the first pass (b) After a second pass: new areas are mapped

Figure 5.6 – Map representing the data from the lane marking detection table showing the areas with good quality markings detection and areas without.

is tagged as usable. It is therefore natural that the areas marked in the following results (Fig. 5.6 and 5.7) form subsets of the ones marked in Fig. 5.5.

5.6.2 Marking Detection

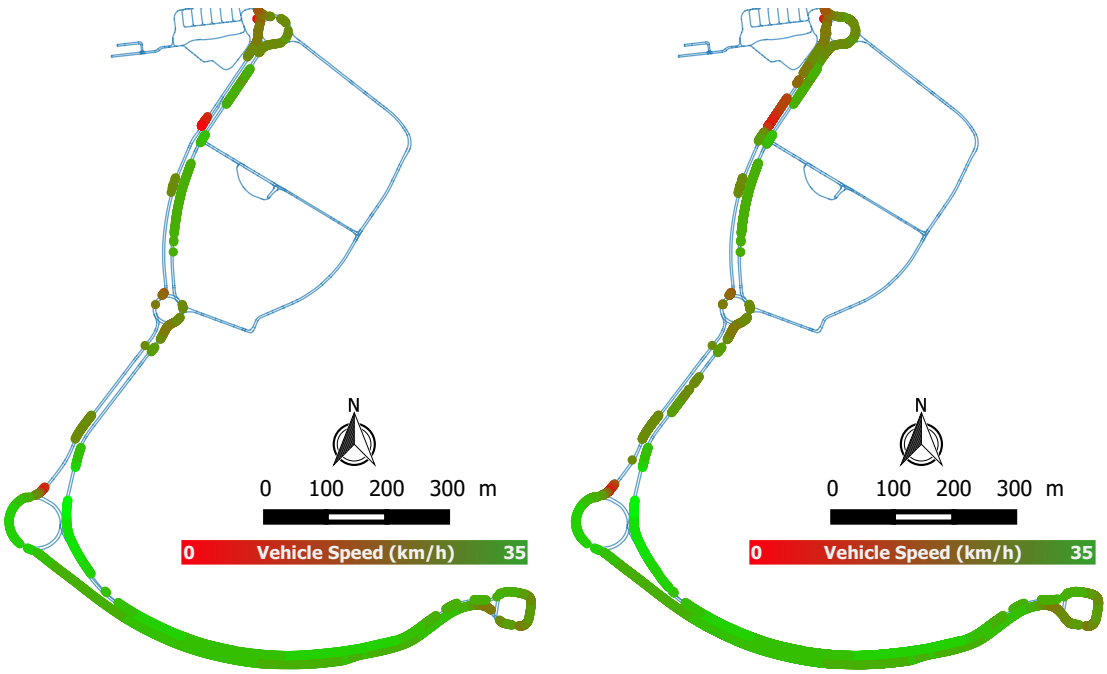
The camera markings detection performance is used as an example of sensor related learning data, Fig. 5.6 shows the classification on the test travel. Two classes are used: "Good" and "No good" detections. These maps show that in some situations, lane markings are not mandatory for a usable positioning. Note that the majority of roads taken have no the lane markings, being only found consistently on the bypass.

Comparing the two maps allows to see the improvements brought by multiple passages: Fig. 5.6a represents the data after a single travel and Fig. 5.6b the same database enhanced with new information from a second travel. Newly mapped areas are visible mostly on the north-south road, in both ways.

5.6.3 Vehicle Speed

In the same manner, a contextual information is studied: the vehicle speed (see Fig. 5.7). As previously, the map is completed with successive travels. The speed values do not change as the travels are done consecutively in the same traffic conditions by the same driver (driving at a very conservative speed).

Some events are observable such as the deceleration when entering a roundabout, the bottom left one being more consequent, the roundabout being larger. Another recorded deceleration is noted in the city part of the travel near an intersection that corresponds to a red traffic light, on which the car comes to a complete stop.



(a) After the first pass (b) After a second pass: new areas are mapped

Figure 5.7 – Maps representing the recorded vehicle speed on the learning travel and its evolution after two passes.

5.6.4 Example of Functionality Usability Determination

This section illustrates the determination of usability of an autonomous functionality, characterized by a criteria named AD-compliance. It considers as an example the requirements for AD-compliance to have an accurate positioning information with good lane markings detection. The corresponding logical statement is thus:

$$\text{AD-compliant} = (\text{Positioning usable}) \wedge (\text{Markings detected}) \tag{5.1}$$

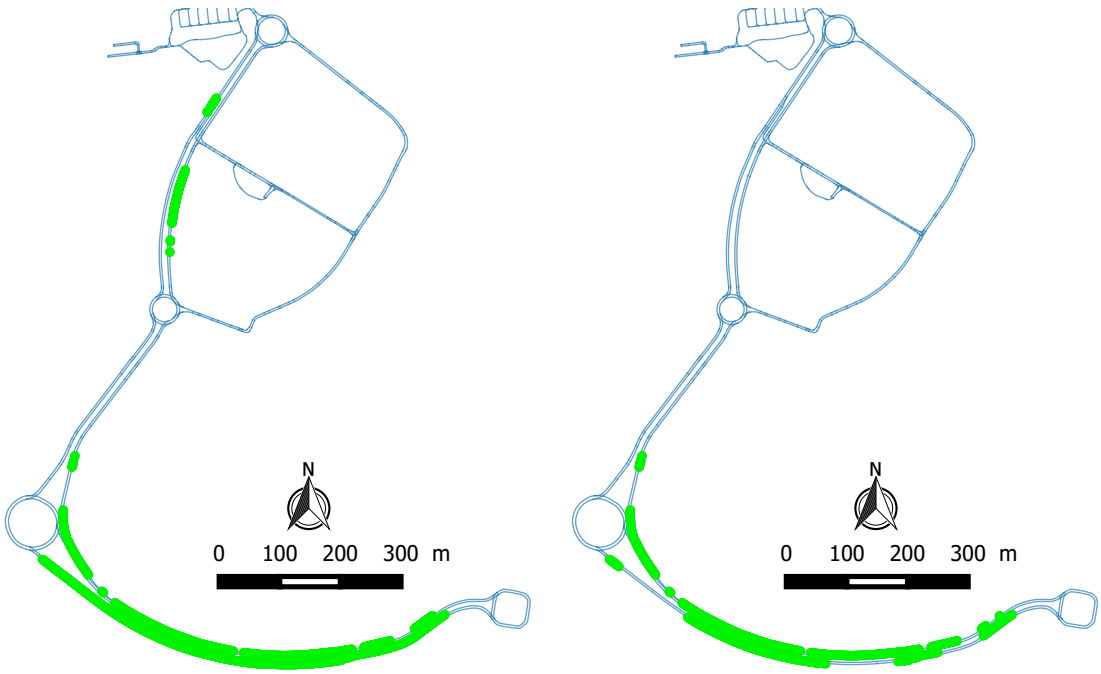
To obtain the AD-compliant map, this logical AND is simply applied using the learning data from the position usability and the marking detection (note that other logical operations can be used, for instance "the usability of sensor A OR sensor B" in case of interchangeable sources of information). The operation is done at each time step with the corresponding data. Fig. 5.8a presents the resulting map, showing the AD-compliant areas (corresponding to the example requirements).

Modifying the requirement is straight forward: elements from the logical statement just need to be added, or removed. For example, Fig. 5.8b shows the result of AD-compliance corresponding to (5.1) with an added minimum speed requirement:

$$\text{AD-compliant} = (\text{Positioning usable}) \wedge (\text{Markings detected}) \wedge (\text{Speed} > 25\,\text{km/h}) \tag{5.2}$$

The number of compliant areas is reduced in consequence, eliminating the ones in slow driving roads (located in urban areas). Note, again that these requirements are arbitrary and are only used to expose the usability determination method.

Note that several strategies can be applied in order to take into account multiple passages. The less safe one would be to consider AD-compliant an area that has



(a) Requirement on the positioning system and lane marking detection

(b) With additional requirement on the speed (must be greater than 25 km/h)

Figure 5.8 – Map representing the AD-compliant areas, determined using different requirements, in a logical AND operation.

been classified as such at least once, even if subsequent passages classified otherwise. This strategy should of course not be used for a safety critical classification. On the contrary, to stay conservative, one can require all passages to yield a positive AD-compliance. This can be very restrictive: for example, if a sensor is malfunctioning during a single passage but is performing well otherwise, the system will be assessed as non compliant. This behavior is however intended if an external cause is responsible for the malfunction (e.g., sun blinding the camera), these causes may need further investigation to determine the actual criticality of the situation.

In order to be less restrictive while respecting the previous caution, a voting system can be implemented, requiring a majority of AD-compliant passages. Equivalently, a maximum number of negative assessments (or a percentage) can be defined. In this case, several passages are need to make a decision, for example a n-out-of-p voting system will require at least n recordings.

5.7 Conclusion

This chapter has presented the use of the digital road map, no only as a source of information for intelligent vehicles, but also as a learning canvas. It provides a learning framework that can be used by the autonomous system to store data, in order to determine the performance of given sub-systems (e.g., sensors). It can then be used to determine the usability of autonomous functionalities by comparing their requirements with the actual sensors performance.

An extensive study has been performed on the map writing efficiency, concluding on the use of SQL transaction, providing a largely sufficient writing performance for

real-time recording (3,333 insert/s). Large amount of data, from multiple sources can therefore be recorded simultaneously. An example of the learning map use has been presented, joining the three learning data sources: the positioning system usability, the lane marking detection and the vehicle speed. An example of autonomous driving requirements associates the three elements to provide a map of AD-compliant areas.

These areas provide prior information to the system to where it is able to perform autonomously. Path planning algorithms can take this information into account in order to provide routes optimized for a greater proportion of autonomous driving, limiting the driver's need to takeover. Even more importantly, the system will be able to anticipate the takeovers caused by the end of an AD-compliant area and thus inform the driver in advance for a takeover in the best conditions.

Chapter 6

Conclusion

Contents

6.1 Findings 154
6.2 Perspectives 156
6.3 Towards Autonomous Vehicles 158

In 2013, autonomous vehicles were still at an exploration phase with different proofs of concept underway. Since then, a whole industry is being created, with all major vehicle OEMs[1] having different programs in place. As of October 2018, one of the leading companies in this domain has driven 16 million kilometers in an autonomous manner [Waymo, 2018]. While multiple issues still remain to be addressed, their deployment beyond laboratory experiments into field operational trials is imminent. High-Definition maps are a fundamental component of the system architecture for autonomous vehicles and an integral part of actuating ADAS. Within this context, this thesis addresses a fundamental issue: determining with certainty the location of an autonomous vehicle with respect to a High-Definition map, in other words, determining the map element corresponding to the vehicle location (e.g., its driving lane). This accurate map-matching is fundamental for a vehicle to start navigating autonomously and for advanced application such as data insertion in the map database.

As work in this thesis progressed and experience within the domain of intelligent vehicles improved, the findings brought by this research are several. These are included in a critical manner in Section 6.1. These findings and experience on addressing localization and maps for intelligent vehicles opened new perspectives, developed in Section 6.2, on how this contribution to research can be furthered.

6.1 Findings

Need for a Robust Positioning

A robust map-matching algorithm has been developed to address the critical need of information association to map elements at the lane level. A multi-hypothesis method based on a particle filter has been designed to return all likely matching hypotheses, returning potentially several in ambiguous situations. These developments leaded into a patent application [Li et al., 2018c]. Moreover, it has been designed to be as independent as possible from the GNSS fix, only using it during the initialization phase and during the tracking mode with protection levels.

An efficient use of the map connectedness data allows a throughout exploration of the road network. For instance, when facing new lanes (either at an intersection or a lane forking), a particle cloning process occurs sending the same quantity of particles to explore each branch, creating new hypotheses. Low likelihood elements are naturally eliminated by the filter, keeping only the best fitting. Lateral connectedness is allowed by letting the particles evolve freely on the 2D plan, enabling the representation of lane changing maneuvers. These event detection processes can be reinforced by using exteroceptive data such as camera lane detection.

A hypothesis coherence checking is then performed to try to eliminate incoherence with the positioning system. It is based on a χ^2 statistical test on a Mahalanobis distance between the hypothesis distribution and the GNSS fix. This method has also been the subject of a patent application [Li et al., 2018b]. Next, if a single hypothesis remains, it can be trusted and the map can thus be used with confidence for storing (i.e. annotating) georeferenced information for learning purposes. This procedure ensures the correctness of the stored information, to stay on par with the

[1] Original Equipment Manufacturer

original map data, which is considered as ground truth (in the sense of correct prior information) and thus must remain untouched.

Map as a Learning Tool

The map-matching result is verified with high confidence, allowing the use of the HD-map. From a simple source of information, the map can be turned into a learning tool by changing it from a read-only to a read-write database. The map provides an interface, acting as a learning canvas, on which different systems can add information.

In this thesis, this feature has been used to explore the possibility to assess the performance of different sensors, with the idea to determine the usability of given autonomous functionalities. This is a good illustration of the map-driven learning process as it demonstrates the importance of the geographic dimension, as sensor performance can vary heavily depending on the navigation environment. It allows the study of parameters in function of the vehicle location. By doing so, a sensor performance map is created. This, in turn, is used to assess the usability of different autonomous functionalities by referring to their requirements in terms of sensor performance. A final map describing the areas compatible with these functionalities is obtained. This is a crucial information that can be used for high-level path planning aiming to maximize the availability of autonomous functionalities to create the best user experience possible in a safe way. It also addresses the problem of driver takeover requests: by knowing when a compatible area ends, the driver can be alerted in advance to facilitate the takeover in safe conditions.

Learning Data Recording

The chosen map database format, namely SpatiaLite, has been tested for read-write operations. The requirements were to be able to insert learning data, while keeping the prior map data unchanged. For this purpose, it was necessary to design the database architecture to manage each sensor in a separate table. Foreign keys are used to point to existing map elements (e.g., links), associating the acquired data with the prior map.

A writing performance benchmark has been carried out, demonstrating the feasibility of on-the-fly writing in the map, using SQL transactions. With an estimation of 60 kB/km per learning variable (considering a 10 Hz recording, driving at 50 km/h), the learning method is also memory-efficient. This is a very important result for repetitive trajectories like with commuting cars.

Experimental Results

Experiments were performed by equipping a passenger vehicle with a set of sensors (i.e. GNSS receiver, inertial units, an intelligent camera) and the access to the vehicle CAN bus. All the experiments were made on public roads, with data acquired considering real time constraints and the randomness encountered when driving on public roads.

The algorithms proposed as part of this thesis were implemented in C++ for efficiency. The performance of the map-matching algorithm has been assessed, along

with the learning map framework. A proof of concept has been realized by storing metrics gathered from the smart camera and from the positioning systems, processing it to characterize the sensor availability in given areas.

6.2 Perspectives

Additional Tests on Various Repetitive Travels

The evaluation of the whole procedure, from map-matching to the recording of learning data, has been performed using a dataset gathered by a test vehicle in a specific area, around the city of Compiègne. For further validation, acquisitions in different geographic areas are needed, including different topographic and road characteristics so as to challenge further the proposed algorithms. The constraint is the availability of accessible HD-maps. Currently, Heudiasyc and Renault are developing further the HD-maps. This should provide an opportunity for further tests, particularly in challenging conditions.

Most passenger cars are used in a repetitive manner (e.g., commuting) encountering varying traffic and weather conditions. As a result, on board sensors may have different behaviors and, at times, fail to fulfill their roles. Another interesting study would use repetitive travels with varying environmental conditions, for example different weather conditions. The goal would be to study challenging situations for the sensors (e.g., rainy conditions, low sunlight for a camera) and assess their performance as a group of sensors. This kind of variations are naturally difficult to predict and creating a complete collection of datasets is thus complicated.

Extending Work on Learning

The processing of learning data presented in this thesis is only at a proof of concept stage. More advanced learning methods are available and a more in-depth study is necessary to fully exploit the learning data potential. The objective in this thesis was to demonstrate the ability to annotate information onto existing maps.

Including more sensors to the learning procedure would also help in improving the learning results, as more information would provide a more substantial final result when aggregated. The requirements for the different sensors must also be defined, in order to adjust the "Use/Don't Use" classification of the different autonomous functionalities. This implies studying the client application, to determine critical sets of sensors (i.e. minimal requirements) needed for safe operation.

Adding Support for More Sensors

Adding more sensors could also benefit the whole algorithm chain. For example, obstacle detection could be used in conjunction with the map in the positioning phase: if a vehicle is detected on the side of the ego-vehicle, it can be assumed that a lane is present on this side, and some ambiguities could potentially be removed.

Another information is the type of the lane marking detected that is returned by the smart camera that could be compared to the information stored in the prior map. It has not been used during this thesis as no information about the reliability of the classification was available. A statistical characterization must first be performed

in order to process this information, as tests shown a non-negligible number of misclassification.

Collaborative Learning

Certainty on the association of the vehicle location and the prior map provides an opening for sharing this information across with other vehicles and the infrastructure through wireless communications links. The arrival of 5G connectivity and reborn interest on V2X technologies is leading to the concept of connected autonomous vehicles. The additional acquired information can be shared to extend the prior knowledge of the road network to other vehicles. By centralizing the learning data from a whole vehicle fleet to a cloud service, data can be aggregated, covering a wider area of the road network with information on sensor performance for example. This collaborative data gathering would benefit to all participants with richer, more complete and thus more powerful information, although cloud computing always raises problems of cyber-security. Anti-spoofing methods must be used on the server side, to verify the authenticity of the vehicle data and on the vehicle side when receiving data from the cloud. While being a great opportunity, cloud computing should not be at the center of a solution, but rather completing a offline solution.

Further Development Refinements

The C++ implementation has been designed for an easy portability, in terms of operating system (compatible with both Windows and Unix systems) but also of data-providing framework. To do this the developed Application Programming Interface (API) offers a framework-agnostic data input: a common data structure is available for the chosen framework to fill. The program can thus be fed using Pacpus Framework, ROS or even through comma-separated values (CSV) files.

One further improvement would be the development of a map loading API. To this day, this map data loading code is format specific. To make it completely generic is challenging, considering the vast number of available map formats. This development would be interesting for robustness assessment by testing other map formats.

A PF relies on the use of a great number of particles, as Monte-Carlo methods are based on a high cardinality of drawing. This is balanced by a relatively simpler calculation to make for each particle. Under these considerations, a PF is a good candidate for an implementation on highly parallel architecture, such as Graphics Processing Unit (GPU).

The computational load, originating from the high number of particles, could therefore be spread across the numerous GPU cores as the operations are independent for each particle. Some care must be taken, as GPGPU[2] porting is not that straight forward. One of the main caveat for PF is the random number generation: using C++ standard library random generator is not thread-safe, so concurrent access can be problematic, more so for highly parallel executions. The risk is to generate the same random number for multiple, if not all particles at the same time step, biasing the Monte-Carlo method. Pre-generating a list of random values before the parallel execution is a solution. NVIDIA CUDA parallel computing platform

[2]General-Purpose computing on Graphics Processing Units

provides the cuRAND library that can be a solution for random number generation (although available on NVIDIA platforms exclusively).

6.3 Towards Autonomous Vehicles

Autonomous vehicle is a very difficult goal to reach. One of the main tasks to fulfill (and one of the most challenging) is situation understanding. Sensors are the main tools available to the vehicle to do so but HD-maps are also a crucial source of contextual information. As demonstrated during this thesis, they also act perfectly as learning canvas to store georeferenced information.

This dual use of HD-maps reinforces the idea that digital road maps are essential elements for the development of advanced autonomous functionalities. However, their use is only possible if the vehicle current position can be matched with certainty on them. This thesis thus proposed a solution for a robust map-matching, the first step towards exploiting the maps at their full potential, to reach the fully autonomous vehicle.

Appendices

Appendix A

Global Navigation Satellite Systems

Contents

A.1 GLONASS . . . 161
A.1.1 History . . . 161
A.1.2 Space Segment . . . 162
A.1.3 Control Segment . . . 162
A.1.4 User Segment . . . 162
A.2 Galileo . . . 163
A.2.1 History . . . 163
A.2.2 Space Segment . . . 164
A.2.3 Control Segment . . . 164
A.2.4 User Segment . . . 164
A.3 BeiDou-2 . . . 164
A.3.1 Space Segment . . . 165
A.3.2 Control Segment . . . 165
A.3.3 User Segment . . . 165

This appendix presents different GNSS excluding GPS, already presented in Chapter 2. The position calculations will not be presented again, being based on the same principles (see Section 2.2.5).

A.1 GLONASS

A.1.1 History

The *Globalnaya Navigatsionnaya Sputnikovaya Sistema* (GLONASS) is the Russian GNSS alternative to the American GPS. Both have been developed during the same period: initiated by the USSR in 1976, the GLONASS has been declared fully operational in December 1995, only 6 months after the GPS. Subsequently, it fell into disrepair as an economic crisis hit the Russian Federation. In 2001, only 6 satellites out of 24 were still functional [Harvey, 2007]. The government then decided the restoration of the system. In 2011, the constellation reached full operational capability for the first time since 1996.

GLONASS is a highly strategic system, notably for Russia, providing a position-

ing system independent of the US Army. Moreover, it grants a better coverage in high latitude regions (e.g., in Siberia) that form a large portion of Russian territory. As for the GPS, two levels of service are offered: the Standard Positioning Service (SPS) and Precise Positioning Service (PPS).

A.1.2 Space Segment

As of March 2018, 24 GLONASS satellites are operational, following into 3 different orbits [Russian Space Systems JSC, 2016]. Their orbital inclination is larger than the GPS at 64°8', covering more efficiently the areas of high latitude. The satellites orbit at an altitude of 19.140 km, the corresponding period of revolution is 11h 15m.

Two frequency bands are used for transmission: L1 and L2 often noted G1 and G2 for clear distinction with GPS bands. A notable difference with GPS is the data transmission method. GPS uses Code-Division Multiple Access (CDMA) in which each satellite modulates a specific code over 2 common carrier frequencies, whereas every GLONASS satellite transmits on a different carrier frequency. The G1 and G2 frequencies for each satellite can be calculated by:

$$\begin{aligned} f_{kG_1} &= 1602 + k \cdot 0.5625\,\mathrm{MHz} \\ f_{kG_2} &= 1246 + k \cdot 0.4375\,\mathrm{MHz} \end{aligned} \tag{A.1}$$

This type of transmission is called Frequency-Division Multiple Access (FDMA). Note that this is the legacy behavior, as the modernization of GLONASS satellites will progressively introduce CDMA transmission.

A.1.3 Control Segment

Russia's large territory allows GLONASS to keep all control stations domestic (see Fig. A.1), except for a control station outside of Russia in Nurek, Tajikistan. The control segment is similar to the GPS one, composed off control stations in charge of tracking and monitoring the satellites, computing the ephemerids and performing corrections if necessary.

A.1.4 User Segment

GNSS receivers must be equipped with hardware able to process specificity of the GLONASS signals. It is however common nowadays, to find GNSS receivers compatible with both GPS and GLONASS. Using satellites from both constellations allows a better coverage and potentially a better geometric configuration, both improving the positioning accuracy.

Note that GLONASS uses the PZ-90.11 coordinate datum (aligned with the International Terrestrial Reference System in 2011), which differ from WGS84 used by GPS by few centimeters. This has to be taken into account when dealing with data from both systems (done by the receiver during calculation).

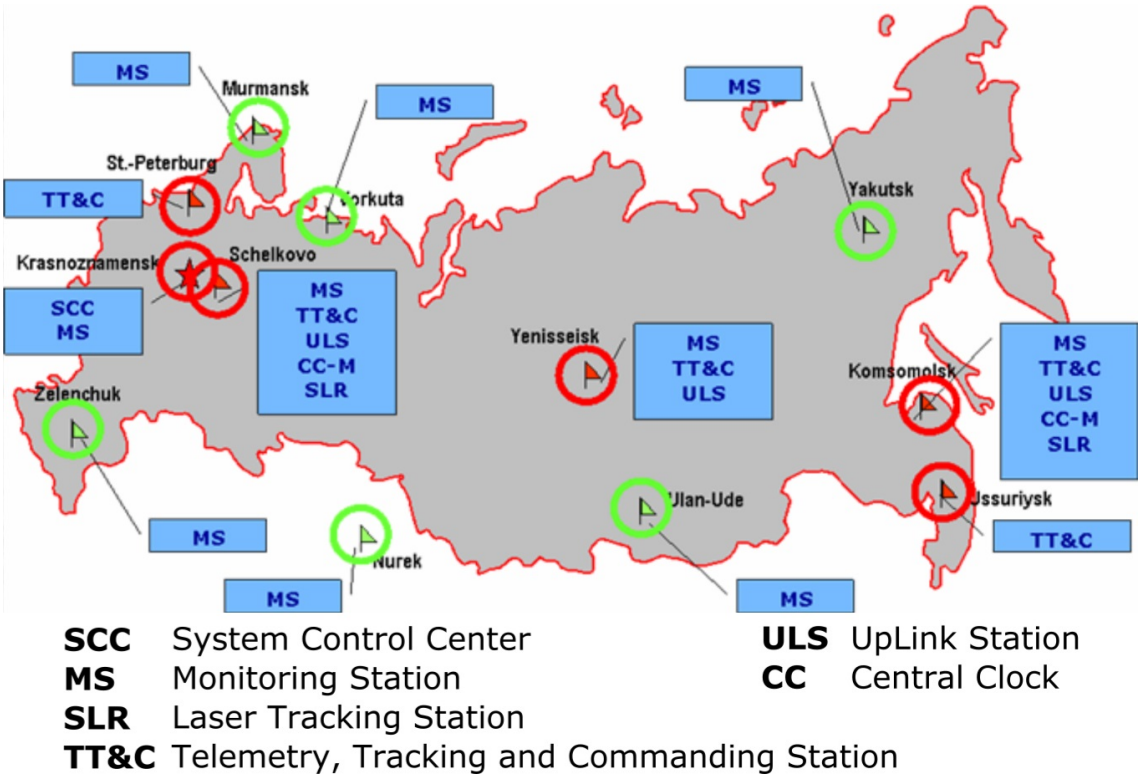


Figure A.1 – Map showing the different element forming the GLONASS control segment (image from CGSIC Meeting).

A.2 Galileo

A.2.1 History

The premise of Galileo, the European GNSS program, began in 1999. Supported by the EU and the European Space Agency (ESA), its goal is to provide a positioning system independent of the American and Russian solutions. The first launch of an operational satellite took place in 2011. As of March 2018, the system is still under completion. It reached early operational capability in December 2016 with, full operational capability is expected by 2020 with a 30-satellite constellation.

Galileo is the first GNSS being develop primarily for civilian use, being supported by the ESA and not a national army program. Once fully deployed, Galileo will offer five main services [Kaplan and Hegarty, 2006]:

- **Open access navigation**: this is the service open to any application, without charge. It is equivalent to the GPS SPS providing positioning down to 1 meter.
- **Commercial navigation**: this paid service will provide high accuracy positioning (centimetric). The performance will be increased by additional encrypted signals providing a guarantee of service and integrity.
- **Public regulated navigation**: government agencies will have access to this encrypted signal, available even when other services are disabled, for example in time of crisis.
- **Safety of life navigation**: the main concern of this service is high integrity. It is aimed at application such as aerial transportation.

- **Search and rescue**: provide a communication channel with emergency beacon to facilitate rescue mission.

A.2.2 Space Segment

30 space vehicles are programmed to be in orbit for full operation, 24 in service and 6 spares. The satellites orbit at an altitude of 23,222 km, on 3 different orbital places (56° inclination).

Three frequencies will be used to transmit the Galileo signals: E1 (1575.42 MHz), E5 (1191.795 MHz) and E6 (1278.75 MHz).

A.2.3 Control Segment

Being a European project, the control segment is spread across Europe. Satellite and Mission Control will take place in the Ground Control Centers, located in Oberpfaffenhofen, Germany and Fucino, Italy. Notably, telemetry, tracking and control stations are distributed around the world but still on European territory, taking advantage of the French overseas departments and regions (French Guiana, New Caledonia, Réunion). Two stations in "mainland" Europe are situated in Sweden and Belgium. Several other stations are distributed worldwide (not limited to European territory) serving as uplink and reference stations.

A.2.4 User Segment

As Galileo is still in deployment (as of the writing of this document), very few receivers are already compatible. Most of them are development kit used for early research, but some general public items have noticeably Galileo-support such as high-end smartphones (e.g., Samsung Galaxy S8). Once again, Galileo signals are usable with other GNSS signals to improve positioning.

Note that the American Federal Communications Commission has not yet approved Galileo signals use in the United States. As a consequence, some devices have been reported to shut down Galileo signal processing when located in the USA. A vote is set to take place in November 2018, where it should get official approval [Inside GNSS, 2018].

A.3 BeiDou-2

Beidou-1 was the first Chinese GNSS test, completed in 2003 with 3 geostationary satellites launched. It covered only a local area around China. The Chinese government was originally supposed to be part of the Galileo project but finally withdrew its participation, considering China's role in the project unsatisfying. The BeiDou Navigation Satellite System (often called BeiDou-2, abbreviated BDS) then began its development and deployment, aiming to reach a global coverage of Earth.

In December 2012, BeiDou-2 became operational in the Asia-Pacific area. The constellation is planned to be finished by 2020, reaching full worldwide operational capability.

A.3.1 Space Segment

BeiDou-2 constellation will be composed of 35 satellites. A specificity, compared to the previous GNSS is the inclusion of 5 geostationary satellites (used for backward compatibility with BeiDou-1). 27 other SVs will be placed in medium Earth orbit (MEO) at an altitude of 27,878 km, spread on 3 orbital plans, and 3 in inclined geosynchronous orbit.

BeiDou-2 will transmit on 4 frequencies, E1 (1575.42 MHz), E2 (1561.098 MHz), E5B (1207.14 MHz), and E6 (1278.75 MHz). These frequencies overlap with Galileo's: while being convenient for receiver-side implementation, questions of signal interference are raised.

A.3.2 Control Segment

BDS's control segment is composed as every GNSS, by a master control station, several uplink stations (currently 2 deployed) and monitoring stations (30 deployed).

A.3.3 User Segment

BeiDou-2 signals are compatible with the other systems so that receivers should be able to process signals from all GNSS simultaneously if equipped with the necessary hardware. A specificity of BDS is the Positioning Report Service, consisting on a link between the ground control segment and the users using Short Message Service (SMS).

Appendix B

Heudiasyc Datasets

Contents

B.1 Acquisition Area 167
B.2 System Setup 167
B.3 Travel 1: "CI_Rocade" 167
B.3.1 Overview 167
B.3.2 Events of Interest 168
B.4 Travel 2: "Rocade" 170
B.4.1 Overview 170
B.4.2 Events of Interest 170

This appendix presents different datasets on which the algorithms developed during this thesis have been tested. Data were gathered on 2015/07/22 using Heudiasyc's test vehicles, presented in Section 2.6.1.

B.1 Acquisition Area

The following datasets have been collected while driving manually in the city of Compiègne, France. The acquisitions focused on areas that have been mapped in the high accuracy lane-level road map format.

B.2 System Setup

The recorded sensors used during this thesis are:

- Septentrio AsterX GNSS receiver (no external corrections), operating at 10 Hz.
- Mobileye EyeQ2 smart camera (lane detection), operating at 10 Hz.
- Vehicle CAN bus frames (yaw rate and rear wheel speed), operating at 50 Hz

B.3 Travel 1: "CI_Rocade"

B.3.1 Overview

This travel began in a peri-urban area, at the exit of UTC's *Centre d'Innovation (CI)* site. The car then headed towards the city bypass (*Rocade*, in French) southward

Figure B.1 – Overview of the travel (ground truth): 4 km-long in Compiègne, France. It is representative of a peri-urban trip passing in areas with multipath conditions (near high-rise buildings) as well as areas with open sky (urban bypass): see Section B.3.2 for more details, the "Bx" labels refers to events described in the following corresponding figures (map data from OpenStreetMap).

taking one-way 2-lane roads and completed the trip back taking the same roads in the other direction (see Fig. B.1 for an overview). This travel was well balanced, spending half the time in peri-urban areas and the other half in open sky conditions.

The weather was sunny and high, not causing lens flare (see Fig. B.2). The traffic flow was fluid, although the test car was driven conservatively slowly (21 km/h mean speed).

System	IRIS
Framework	Pacpus
Date	2015/07/22-15:16
Run Time	approx. 10 min
Distance Driven	4 km (see Fig. B.1)

B.3.2 Events of Interest

During this travel, some events of interest occur, notably: a stop at a red light, multiple roundabouts, lane changing (more details further).

- **At** $t = 57\,\mathbf{s}$**: Red Light**.
- **At** $t = [93, 102]\,\mathbf{s}$**: GNSS Loss** as the car pass under a small concrete bridge, causing interference (see Fig. B.3).
- **At** $t = 122\,\mathbf{s}$**: Roundabout**, taking the opposite exit (first exit).

Figure B.2 – Image from the contextual camera (front facing dashboard camera).

- **At** $t = [165, 183]$ **s: Multipath** caused by a nearby high-rise building (see Fig. B.4).
- **At** $t = 206$ **s: Roundabout**, entering the bypass (third exit).
- **At** $t = 304$ **s: Lane Change**, going from right to left lane (see Fig. B.5).
- **At** $t = 340$ **s: Roundabout**, making a U-turn (fourth exit).
- **At** $t = 455$ **s: Exit Ramp**, keeping right at the bifurcation (see Fig. B.6).
- **At** $t = [517, 525]$ **s: Multipath** caused by a nearby high-rise building.
- **At** $t = [535, 550]$ **s: GNSS Loss** caused by a nearby building.
- **At** $t = 604\,s$**: Final Roundabout**

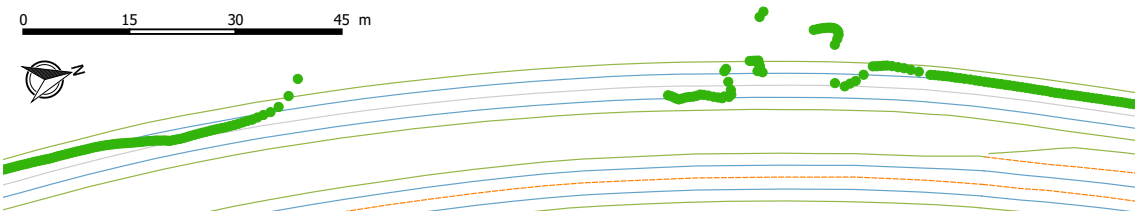


Figure B.3 – Details of the GNSS fixes, during GNSS loss at $t = [93, 102]$ s, with multipath-induced errors. The car was traveling southward (right to left).

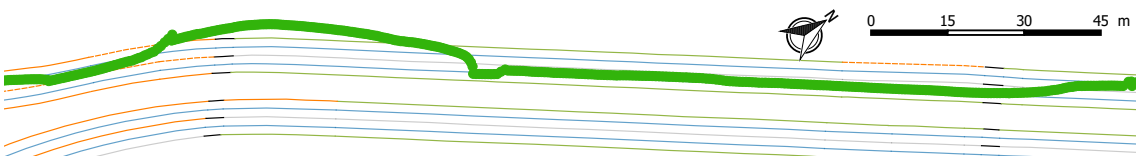


Figure B.4 – Details of the GNSS fixes, during multipath at $t = [165, 183]$ s. Note that the car stayed constantly on the right lane, the variations were caused by multipath.

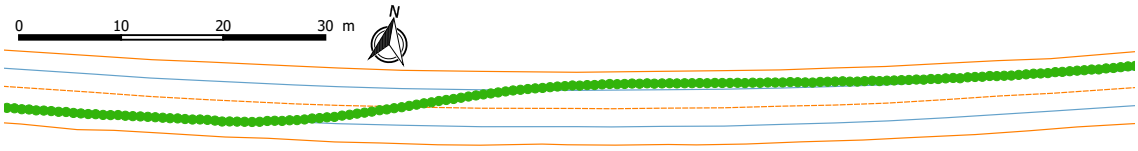


Figure B.5 – Details of the GNSS fixes, during lane change on the bypass at $t = 304\,\mathrm{s}$.

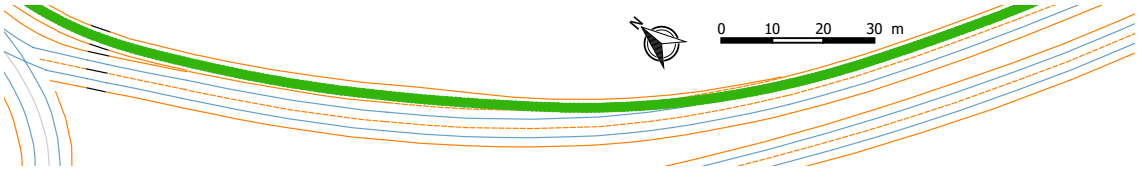


Figure B.6 – Details of the GNSS fixes, reaching an exit ramp at $t = 455\,\mathrm{s}$, and taking it.

B.4 Travel 2: "Rocade"

B.4.1 Overview

This recording was performed exclusively on the city bypass. It started at the exit of the west roundabout and ended at the entrance of the same roundabout, after driving back and forth on the bypass (see Fig. B.7). The driving conditions (weather and traffic) were identical to the first travel.

The travel is performed in open sky conditions: it aims to represent nominal driving conditions to assess the performance of the algorithm in "stationary operation" (opposed to converging phase).

Figure B.7 – Overview of the travel (ground truth): 2 km-long on Compiègne bypass, France. This dataset is aimed to validate the behavior of the algorithm in nominal conditions.

System	IRIS
Framework	Pacpus
Date	2015/07/22-15:29
Run Time	approx. 5 min
Distance Driven	2 km (see Fig. B.7)

B.4.2 Events of Interest

The travel being aimed to represent nominal conditions, no much events occurs. Only a lane change happens at $t = 270\,\mathrm{s}$ (see Fig. B.8): the car changed from the

right lane to the left, near the exit ramp (not taking it).

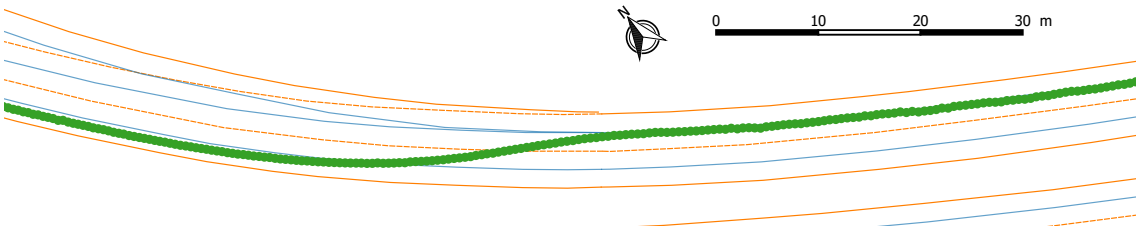


Figure B.8 – Lane change event during travel 2. The car was traveling westward (right to left).

Appendix C

MA-HUL Along-Track and Cross-Track Separation

Contents

C.1 Raw Distance 173
C.2 Along and Cross-Track Separation 174
C.3 Results 175

As presented in Section 4.4.1, the GNSS receiver often underestimates its uncertainty, potentially leading to wrong covariance matrices. Likewise, a PF, depending on its tuning, can also return incorrect covariance estimates. Based on this observation, this section presents a measurement removing the covariance from the calculation, in order to be unaffected by its possible error.

The goal is also to characterize the confidence that can be given to the positioning system. The same data are used, but with this rationale the GNSS fix and the map-matching hypotheses are now considered as plain geographic positions, i.e. using the mean value of the distribution, without covariances.

The calculation is based loosely on the aeronautical RAIM principle. Classical HUL calculation requires access to raw GNSS measurements (i.e. pseudo-ranges, that are not available in the scope of this research) and compares them to the computed position, estimating the uncertainty of the fix at the given time. The developed method relies on the map-matching data to provide some redundancy to the system: the multiple hypotheses are used in place of the pseudo-ranges, yielding consistency metric named Map-Aided Horizontal Uncertainty Level (MA-HUL) [Li et al., 2017b].

C.1 Raw Distance

The Cartesian distances between the GNSS fix and the position hypotheses (see Section 3.3.8) are indicators of the error between the GNSS fix and the results from the map-matching. It illustrates the discrepancy between the two sources of positioning and is thus an estimation of the GNSS position uncertainty relative to the map data.

The MA-HUL is set to be the largest of the computed distances (see Fig. C.1):

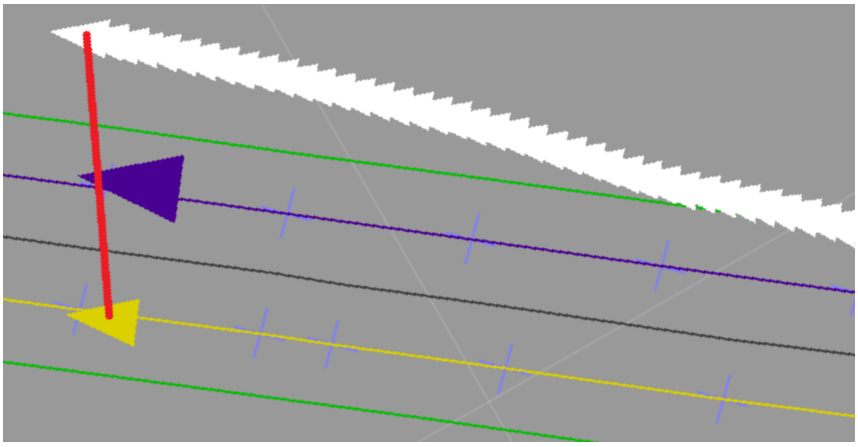


Figure C.1 – Biased GNSS position (the GNSS track over time is displayed by white triangles, a deviation is clearly visible) along with the associated matching hypotheses at the current time index (colored triangles). The MA-HUL is denoted by the red line, corresponding to the distance between the GNSS position and the farthest matching hypothesis.

$$\text{MA-HUL} \ = \max_j ||X_{\text{GNSS}} X_{\text{hyp}_j}||^2 \tag{C.1}$$

Choosing the largest distance is motivated by integrity considerations. It then corresponds to the minimum radius of a circle centered on the GNSS fix enclosing all hypotheses, being comparable to an HUL. The MA-HUL thus characterizes the positioning system with a conservative level of consistency linked to the map-matching process.

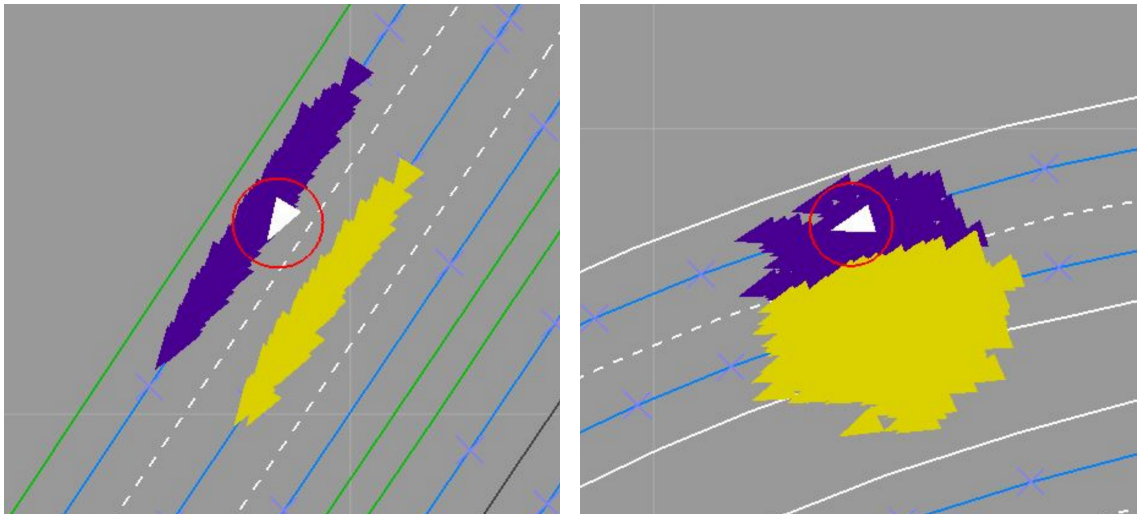
C.2 Along and Cross-Track Separation

A characteristic of the PF developed in this research, shared with all other solutions using dead-reckoning, is a tendency to drift over time. In the case of the PF, it manifests itself as a particle spreading, generally oriented in the less constrained direction (see Fig. C.2).

For instance, in straight lines with a lane marking detection, the particles are well constrained laterally by the markings, but no longitudinal information is available: the particles tend to spread along the lane (the along-track direction, see Fig. C.2a).

The opposite phenomenon is observed in sharp bends: the substantial change of direction eliminates the particles too far longitudinally (as their position is now inconsistent with the map). This can cause the spread to change direction and to be more prominent laterally. It is amplified by the fact that lane markings are harder to detect robustly in important curves. The larger component of the spread is then in the cross-track direction (see Fig. C.2b).

This directional asymmetry can lead to situations where the MA-HUL, as computed in (C.1), may not represent accurately the positioning uncertainty when considering its longitudinal and lateral components independently. This separation is useful, if not necessary, when considering some functionalities that only require accurate localization in a single direction. For example, basic Lane Keeping Assist systems only need lateral localization. Using a MA-HUL as described previously



(a) (b)

Figure C.2 – Particles spread due to dead-reckoning drift in a PF. Depending on the situation, the spread can be more prominent in the *along-track* direction (C.2a) or in the *cross-track* direction (C.2b).

could be prejudicial for such systems, as the metric aggregates both directions by computing the Cartesian distance. In order to have a more discerning view of the uncertainty in a single given direction, it is important to consider separately two metrics: the along-track MA-HUL, following the lane direction, and the cross-track MA-HUL, being orthogonal to the former [Li et al., 2017a].

The principle is to simply express the vectors $\overrightarrow{X_{\text{GNSS}}X_{\text{hyp}_j}}$ in a Frenet reference frame following the road. Also called a Tangent-Normal frame, its first axis matches the direction of the lane, while the second is in a orthogonal direction. The tangent and normal components of the MA-HUL coordinates expressed in this reference frame corresponds respectively to the along-track and cross-track MA-HUL. Fig. C.3 shows a situation that benefits from this separation. The particles are spread longitudinally leading to the mean hypotheses being shifted in this direction, in comparison with the GNSS fix. The along-track uncertainty accounts for the majority of the total MA-HUL, while the cross-track value is relatively small. If the system were to consider the raw MA-HUL, system such as LKA would not be able to confidently rely on the localization information, but taken separately, it appears that the cross-track uncertainty value can be sufficient for a lateral guiding function (half a lane width).

Note that these calculations and map-matching hypotheses are only used here to characterize the positioning system uncertainty. To be able to process it for an application such as LKA, another positioning algorithm must be run, as the present multi-hypothesis result is not usable directly.

C.3 Results

The MA-HUL is analyzed on the mixed test travel. The raw value, compared with the GNSS error, is shown in Fig. C.4. The MA-HUL acts 94% of the time as an

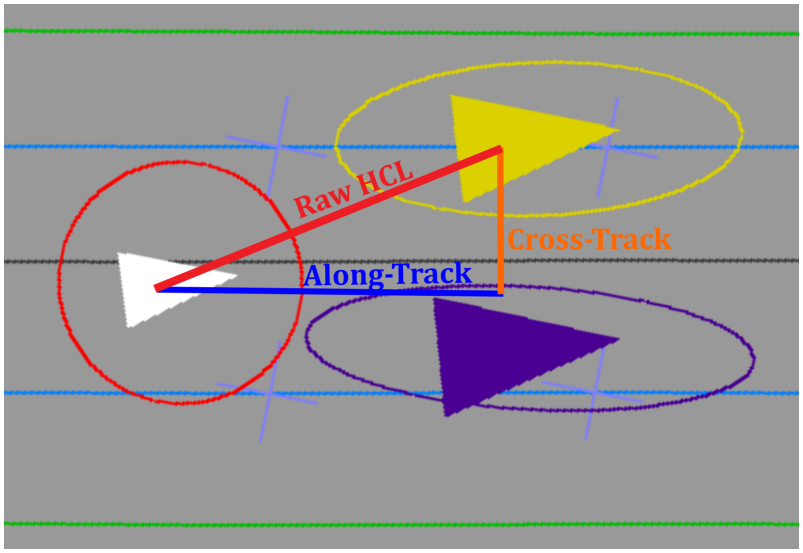


Figure C.3 – Separating the MA-HUL into cross-track and along-track components allows to characterize more finely the error. Here, the cross-track error is much smaller than the raw HCL.

upper-bound of the GNSS error, confirming the integrity origins of its design inspired from the HUL. Additionally, the MA-HUL does not exceed the 15 m HPL value, as it relies on the map-matching solution that does not allow particles to cross that HPL limit.

The along/cross-track problem is clearly observable between $t = [230, 330]\,s$: during this time frame, the car travels on the bypass. The markings detection allows a good lateral accuracy but, as the road follows approximately a straight line (it is actually a curve with a very large curvature radius—around 512 m), some longitudinal spread appears. This is causing the MA-HUL to increase progressively. At $t = 310\,s$ the road curvature changes to a sharper bend: this evolution tightens the particle longitudinal spread and thus decreases drastically the MA-HUL value ($t = [310, 330]\,s$).

Fig. C.5 shows the MA-HUL with its along/cross-track separated variants. Studying it on the same time interval demonstrates the interest of the separation: during $t = [230, 260]$ the situation still presents some lateral spreading. Afterwards, the filter converges to a good position, corresponding laterally to the GNSS fix: the cross-track MA-HUL decreases. Meanwhile, the along-track value increases due to the particle spreading, nearly accounting for the total MA-HUL described above. With the road bend at $t = 310\,s$, camera loses track of the lane markings, removing the lateral constrains from the filter: an strong cross-track MA-HUL increase is observed.

Quantitatively, this separation is beneficial as it allows the processing of smaller HCLs. Fig. C.6a and C.6c show respectively the distribution of the raw and the cross-track HCL. The raw MA-HUL distribution is shifted higher of nearly $2\,m$ compared to the cross-track one.

Moreover, a lateral threshold of $3.5\,m$, corresponding to the width of a lane, is respected 77% of the time when using the cross-track MA-HUL, against only 25% for the raw value (see Fig. C.6b and C.6d). The gap is even larger when considering a $1.75\,m$ threshold (half a lane width): it is respected 21% of the time for the

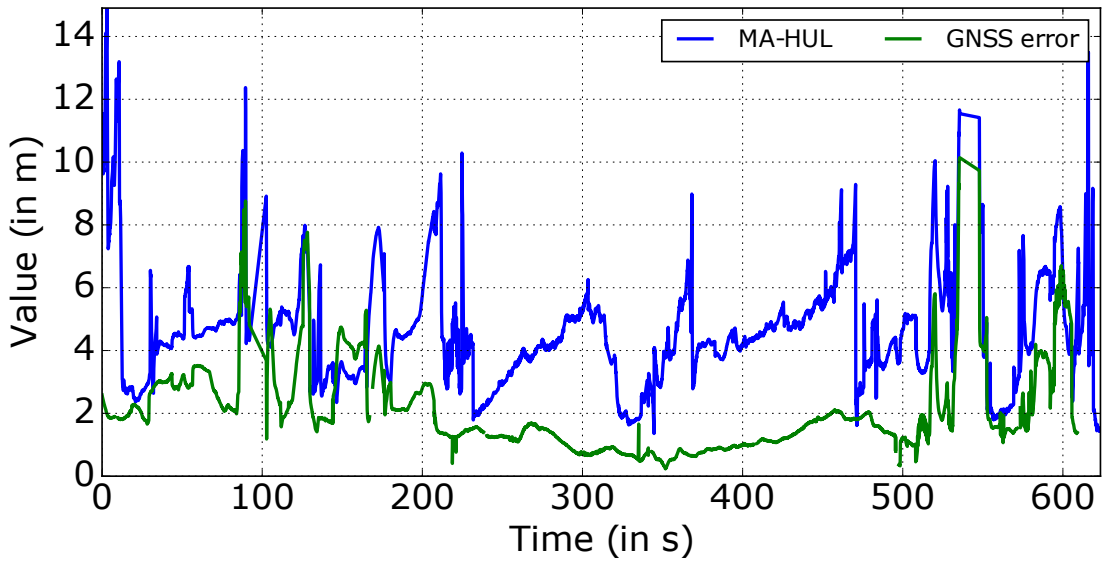


Figure C.4 – The MA-HUL (in blue) appears as an upper bound of the GNSS error (in green).

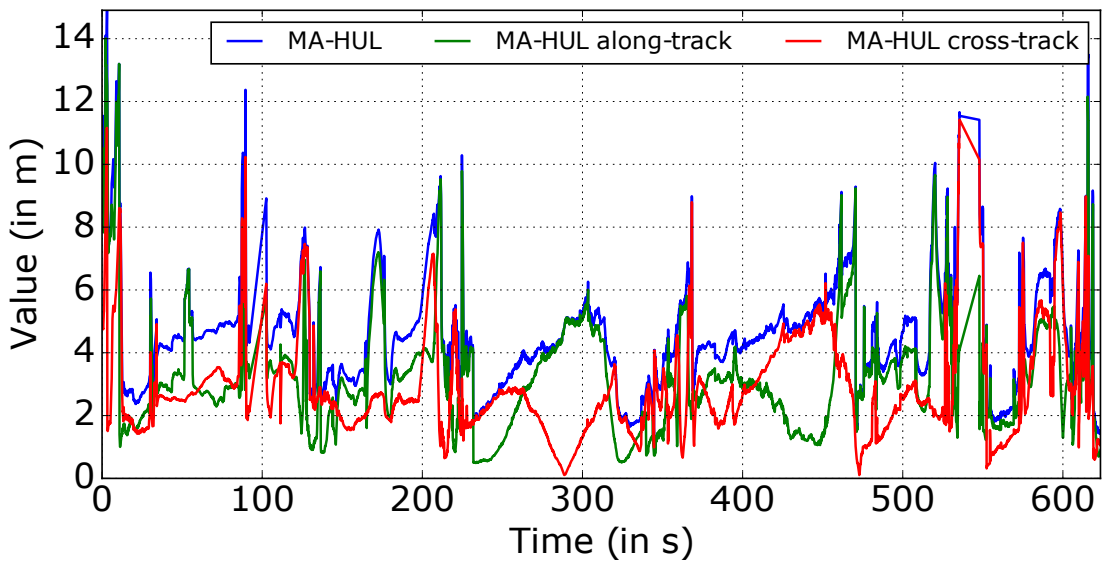


Figure C.5 – Separating the MA-HUL in two: along-track (in cyan) and cross-track (in green). The camera detections are used, improving the cross-track uncertainty.

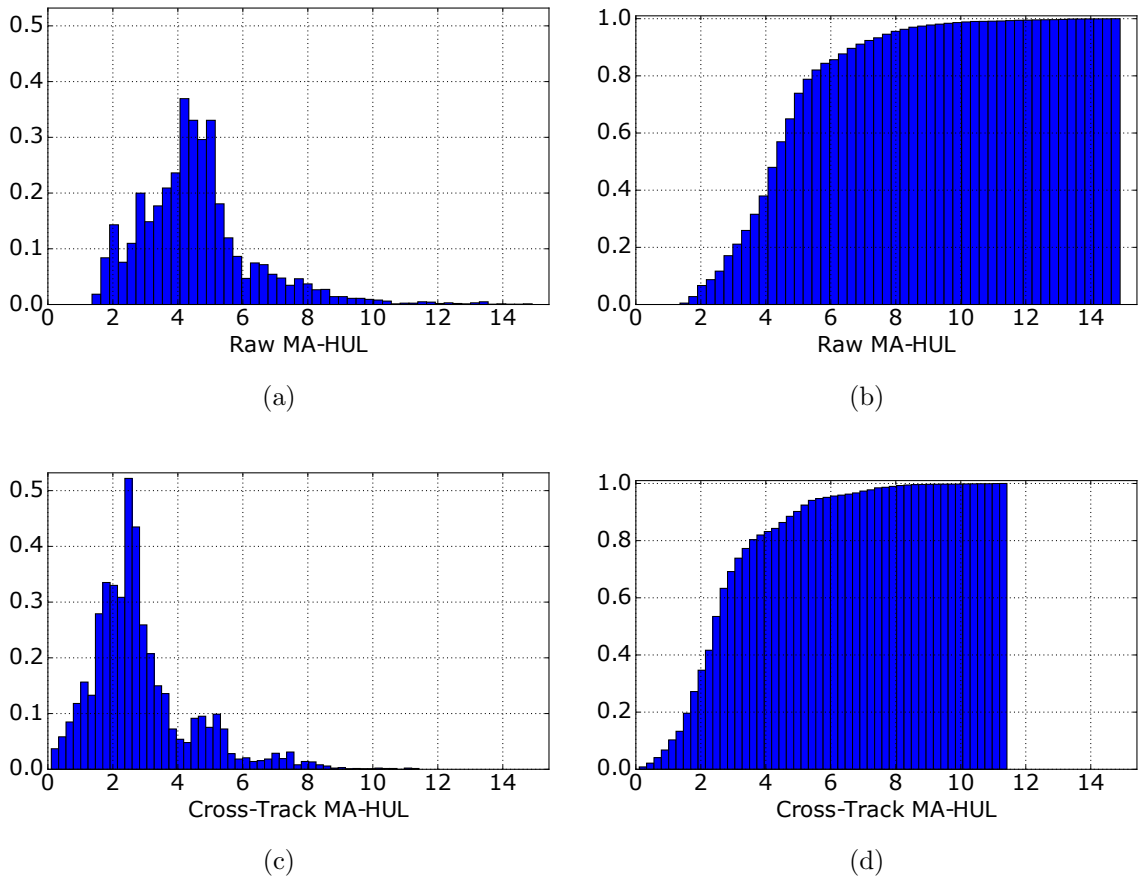


Figure C.6 – Histograms showing the distributions of the MA-HUL (on the left) and their cumulative density functions (on the right). The raw value is presented on top and the cross-track is at the bottom.

separated metric against only 1% for the raw one. The separation in cross-track MA-HUL would provide a greater availability for applications needing only lateral accuracy such as LKA. This is especially true with the $1.75\,m$ threshold, as the lateral error is less than half a lane width meaning the correct lane is identifiable.

Bibliography

Aitken, A. C. (1936). IV.–On Least Squares and Linear Combination of Observations. *Proceedings of the Royal Society of Edinburgh*, 55:42–48. 105

Akaike, H. (1998). Information Theory and an Extension of the Maximum Likelihood Principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer. 127

Al Hage, J., E. El Najjar, M., and Pomorski, D. (2017). Multi-sensor fusion approach with fault detection and exclusion based on the Kullback-Leibler Divergence: Application on collaborative multi-robot system. *Information Fusion*, 37:61–76. 128

Anderson, J., Kalra, N., Stanley, K., Sorensen, P., Samaras, C., and Oluwatola, O. (2016). *Autonomous Vehicle Technology: A Guide for Policymakers*. RAND Corporation. 7

Armand, A., Filliat, D., and Ibañez-Guzmán, J. (2014). Ontology-based context awareness for driving assistance systems. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, volume 1, pages 227–233. IEEE. 7

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188. 62, 65, 69

Baarda, W. (1968). A Testing Procedure for Use in Geodetic Networks. *Publications on Geodesy*, 2(5):97. 107

Banerjee, A., Merugu, S., Dhillon, I., and Ghosh, J. (2004). Clustering with Bregman Divergences. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, volume 6, pages 234–245. Society for Industrial and Applied Mathematics, Philadelphia, PA. 128

Bar-Shalom, Y. and Li, X.-R. (1993). *Estimation and tracking: Principles, techniques, and software*. Artech House, Inc. 118

Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, USA. 97

Bar-Shalom, Y., Willett, P. K., and Tian, X. (2011). *Tracking and data fusion*. YBS Publishing. 11

Barrios, J. M., Hochberg, Y. V., and Yi, H. (2018). The Cost of Convenience: Ridesharing and Traffic Fatalities. *SSRN Electronic Journal*, (27). 3

Bender, P., Ziegler, J., and Stiller, C. (2014). Lanelets: Efficient map representation for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, number IV, pages 420–425. IEEE. 36

Bergman, N. (1999). *Recursive Bayesian Estimation Navigation and Tracking Applications*. PhD thesis, Linköping University. 66

Betaille, D. (2017). Paving the way for future use of the Urban Trench model along with a lane level road map. In *2017 European Navigation Conference (ENC)*, pages 143–149. IEEE. 23

Betaille, D. and Toledo-Moreo, R. (2010). Creating Enhanced Maps for Lane-Level Vehicle Navigation. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):786–798. 30, 32, 36, 69

Blackman, S. S. (2004). Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18. 62

Blanch, J., Ene, A., Walter, T., and Enge, P. (2007). An Optimized Multiple Hypothesis RAIM Algorithm for Vertical Guidance. *Proceedings of ION GPS*, pages 2924–2933. 104

Bonnifait, P., Laneurit, J., Fouque, C., and Dherbomez, G. (2009). Multi-Hypothesis Map-Matching Using Particle Filtering. In *16th World Congress for ITS Systems and Services*, number September, pages 1–8. 46, 47, 72

Borenstein, J. and Liqiang Feng (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880. 49

Braasch, M. S. (1996). Multipath Effects. In *Global Positioning System: Theory and Applications, Volume I*, pages 547–568. American Institute of Aeronautics and Astronautics, Washington DC. 23

Brenner, M. (1996). Integrated GPS/Inertial Fault Detection Availability. *Navigation*, 43(2):111–130. 104

Bressler, J., Reisdorf, P., Obst, M., and Wanielik, G. (2016). GNSS positioning in non-line-of-sight context—A survey. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1147–1154. IEEE. 23

Broadcom (2017). Broadcom BCM4775X - Product Brief. Technical report. 50

Brown, R. G. (1996). Receiver Autonomous Integrity Monitoring. In *Global Positioning System: Theory and Applications, Volume II*, pages 143–165. American Institute of Aeronautics and Astronautics, Washington DC. 26, 104

Castellanos, J. A., Neira, J., and Tardós, J. D. (2004). Limits to the consistency of EKF-based SLAM. *IFAC Proceedings Volumes*, 37(8):716–721. 97

Chen, Z. (2003). Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. *Statistics*, 182(1):1–69. 58

Cosmen-Schortmann, J., Azaola-Saenz, M., Martinez-Olague, M., and Toledo-Lopez, M. (2008). Integrity in urban and road environments and its use in liability critical applications. In *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 972–983. IEEE. 28, 56, 72, 83

DARPA (2007). Urban Challenge Route Network Definition File (RNDF) and Mission Data File (MDF) Formats. *Urban Challenge*. 34

Dickmanns, E. D. and Mysliwetz, B. D. (1992). Recursive 3-D road and relative ego-state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):199–213. 36

Douc, R. and Cappe, O. (2005). Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE. 66

Drevelle, V. (2011). *Étude de méthodes ensemblistes robustes pour une localisation multisensorielle intègre Application à la navigation des véhicules en milieu urbain.* PhD thesis. xvi, 106

Drevelle, V. and Bonnifait, P. (2011). A set-membership approach for high integrity height-aided satellite positioning. *GPS Solutions*, 15(4):357–368. 27, 32

Du, J. and Barth, M. J. (2008). Next-Generation Automated Vehicle Location Systems: Positioning at the Lane Level. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):48–57. 30

Duchi, J. (2014). Derivations for Linear Algebra and Optimization. Technical report, Stanford University. 128

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110. 81

Durrant-Whyte, H., Majumder, S., Thrun, S., de Battista, M., and Scheding, S. (2003). A Bayesian Algorithm for Simultaneous Localisation and Map Building. In *Robotics Research*, pages 49–60. 57

Engelson, S. P. and McDermott, D. V. (1992). Error correction in mobile robot map learning. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2555–2560. IEEE Comput. Soc. Press. 45

European Commission (2016). Road Safety Country Overview – France. Technical Report September 2016, European Commission, Directorate General for Transport. 5

Filliat, D. and Meyer, J.-A. (2003). Map-based navigation in mobile robots: I. A review of localization strategies. *Cognitive Systems Research*, 4(4):243–282. 2

Fouque, C. and Bonnifait, P. (2012). Matching Raw GPS Measurements on a Navigable Map Without Computing a Global Position. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):887–898. 44, 46

Fouque, C., Bonnifait, P., and Betaille, D. (2008). Enhancement of global vehicle localization using navigable road maps and dead-reckoning. In *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 1286–1291. IEEE. 79

Fox, D. (2003). Adapting the Sample Size in Particle Filters Through KLD-Sampling. *The International Journal of Robotics Research*, 22(12):985–1003. 128

Gackstatter, C., Heinemann, P., Thomas, S., Rosenhahn, B., and Klinker, G. (2010). Fusion of clothoid segments for a more accurate and updated prediction of the road geometry. In *13th International IEEE Conference on Intelligent Transportation Systems*, volume 2, pages 1691–1696. IEEE. 36

Gilliéron, P.-Y., Gontran, H., and Merminod, B. (2006). Cartographie routière précise pour les systèmes d'assistance à la conduite. In *Conférence GIS – SIT 2006*, pages 1–18, Zürich. xiii, 13

GMV Aerospace and Defence S.A. (2009). EP2113786A1 – Method for autonomous determination of protection levels for GNSS positioning based on navigation residuals and an isotropic confidence ratio. 28

GMV Aerospace and Defence S.A. (2013). US 20160109579A1 – Device And Method For Computing An Error Bound Of A Kalman Filter Based Gnss Position Solution. 29

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221. 34

Groves, P. D., Jiang, Z., Rudi, M., and Strode, P. (2013). A portfolio approach to NLOS and multipath mitigation in dense urban areas. *26th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2013*, 4(September):3231–3247. xiii, 24

Gu, Y., Hsu, L. T., and Kamijo, S. (2015). Passive sensor integration for vehicle self-localization in urban traffic environment. *Sensors (Switzerland)*, 15(12):30199–30220. 68

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., and Nordlund, P.-J. (2002). Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437. 63, 66, 69, 77

Gustafsson, F., Orguner, U., Schön, T. B., Skoglar, P., and Karlsson, R. (2012). Navigation and Tracking of Road-Bound Vehicles Using Map Support. In *Handbook of Intelligent Vehicles*, volume 1-2, pages 397–434. Springer London, London. xiv, 45, 46

Haklay, M. and Weber, P. (2008). OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18. 34

Harvey, B. (2007). *The Rebirth of the Russian Space Program.* Springer Praxis Books. Springer New York, New York, NY. 161

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*, volume 2 of *Springer Series in Statistics*. Springer New York, New York, NY. 136

Hata, A. and Wolf, D. (2014). Road marking detection using LIDAR reflective intensity data and its application to vehicle localization. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 584–589. IEEE. 71, 81

Hery, E., Xu, P., and Bonnifait, P. (2017). Along-track localization for cooperative autonomous vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 511–516. IEEE. 50

Hofmann-Wellenhof, B., Lichtenegger, H., and Wasle, E. (2008). *GNSS – Global Navigation Satellite Systems*. Springer Vienna, Vienna. 23

Hopfield, H. S. (1969). Two-quartic tropospheric refractivity profile for correcting satellite data. *Journal of Geophysical Research*, 74(18):4487–4499. 23

Hsu, L. T. (2017). GNSS multipath detection using a machine learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE. 23

Ibañez-Guzmán, J., Laugier, C., Yoder, J.-D., and Thrun, S. (2012). Autonomous Driving: Context and State-of-the-Art. In Eskandarian, A., editor, *Handbook of Intelligent Vehicles*, volume 1-2, pages 1271–1310. Springer London, London. xiii, 7, 10

Ibañez-Guzmán, J., Lefevre, S., Mokkadem, A., and Rodhaim, S. (2010). Vehicle to vehicle communications applied to road intersection safety, field results. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 192–197. IEEE. 4

Inside GNSS (2018). FCC Sets Vote on Galileo Approval for November Meeting. `http://insidegnss.com/fcc-sets-vote-on-galileo-approval-for-november-15-meeting/`. 164

Institut national de l'information géographique et forestière IGN (2018). `http://www.ign.fr/`. 38

International Civil Aviation Organization ICAO (2002). *World Geodetic System - 1984 (WGS-84) Manual*. xiii, 21, 22

International Civil Aviation Organization ICAO (2005). Doc 9849 AN/457 Global Navigation Satellite System (GNSS) Manual. Technical report. 25

ISO (2011). ISO26262:2011 - Road vehicles – Functional safety. Technical report, International Organization for Standardization. 9

Jabbour, M., Bonnifait, P., and Cherfaoui, V. (2008). Map-Matching Integrity Using Multihypothesis Road-Tracking. *Journal of Intelligent Transportation Systems*, 12(4):189–201. 47, 80

Jazwinski, A. H. (1970). *Stochastic Processes And Filtering Theory.* 59

Jermakian, J. S. (2011). Crash avoidance potential of four passenger vehicle technologies. *Accident Analysis & Prevention*, 43(3):732–740. 5

Joerger, M., Chan, F.-C., and Pervan, B. (2014). Solution Separation Versus Residual-Based RAIM. *Navigation*, 61(4):273–291. 104

Jorgensen, P. S. (1989). An Assessment of Ionospheric Effects on the GPS User. *Navigation*, 36(2):195–204. 23

Joshi, A. and James, M. R. (2015). Generation of Accurate Lane-Level Maps from Coarse Prior Maps and Lidar. *IEEE Intelligent Transportation Systems Magazine*, 7(1):19–29. 80

Joshi, R. R. (2001). A new approach to map matching for in-vehicle navigation systems: the rotational variation metric. In *2001 IEEE Intelligent Transportation Systems Conference Proceedings*, pages 33–38. IEEE. 44

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In Kadar, I., editor, *SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, page 182. 62

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35. 60

Kamijo, S., Okumura, K., and Kitamura, A. (1989). Digital road map database for vehicle navigation and road information systems. In *Conference Record of papers presented at the First Vehicle Navigation and Information Systems Conference (VNIS '89)*, pages 319–323. IEEE. 34

Kaplan, E. D. and Hegarty, C. J. (2006). *Understanding GPS: principles and applications.* Artech house. 12, 18, 19, 21, 22, 163

Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44. 11

Kim, W., Jee, G.-I., and Lee, J. (2000). Efficient use of digital road map in various positioning for ITS. In *IEEE 2000. Position Location and Navigation Symposium (Cat. No.00CH37062)*, pages 170–176. IEEE. 45

Kim, Z. (2008). Robust Lane Detection and Tracking in Challenging Scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26. 43, 81

Klobuchar, J. A. (1996). Ionospheric Effects On Gps. In *Global Positioning System: Theory and Applications, Volume I*, pages 485–515. American Institute of Aeronautics and Astronautics, Washington DC. 23

Krumm, J. (2012). How People Use Their Vehicles: Statistics from the 2009 National Household Travel Survey. In *SAE 2012 World Congress & Exhibition*, pages 1–12. 136

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86. 127

Lau, L. and Cross, P. (2007). Development and testing of a new ray-tracing approach to GNSS carrier-phase multipath modelling. *Journal of Geodesy*, 81(11):713–732. 23

Le Marchand, O., Bonnifait, P., Ibañez-Guzmán, J., and Betaille, D. (2009). Vehicle localization integrity based on trajectory monitoring. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3453–3458. IEEE. 27

Lee, K. W., Wijesoma, W., and Ibañez-Guzmán, J. (2004). Map aided SLAM in neighbourhood environments. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 836–841. IEEE. 136

Lefevre, S., Laugier, C., and Ibañez-Guzmán, J. (2012). Risk assessment at road intersections: Comparing intention and expectation. In *2012 IEEE Intelligent Vehicles Symposium*, pages 165–171. IEEE. 42, 45

Levinson, J., Montemerlo, M., and Thrun, S. (2007). Map-Based Precision Vehicle Localization in Urban Environments. In *Robotics: Science and Systems III*. Robotics: Science and Systems Foundation. 69

Li, F., Bonnifait, P., and Ibañez-Guzmán, J. (2017a). Estimating localization uncertainty using multi-hypothesis map-matching on high-definition road maps. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1408–1413, Yokohama. IEEE. 47, 81, 175

Li, F., Bonnifait, P., and Ibañez-Guzmán, J. (2017b). Using High Definition Maps to Estimate GNSS Positioning Uncertainty. In *2017 European Navigation Conference (ENC)*. 113, 114, 173

Li, F., Bonnifait, P., and Ibañez-Guzmán, J. (2018a). Map-Aided Dead-Reckoning With Lane-Level Maps and Integrity Monitoring. *IEEE Transactions on Intelligent Vehicles*, 3(1):81–91. 47, 118

Li, F., Bonnifait, P., and Ibañez-Guzmán, J. (2018b). Patent Application: Procédé de décision et diagnostic utilisant un calcul de cohérence. 154

Li, F., Bonnifait, P., and Ibañez-Guzmán, J. (2018c). Patent Application: Procédé de map-matching utilisant un filtre particulaire. 154

Li, F., Bonnifait, P., Ibañez-Guzmán, J., and Zinoune, C. (2017c). Lane-level map-matching with integrity on high-definition maps. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1176–1181. IEEE. 47

Maybeck, P. S. (1982). *Stochastic models, estimation, and control*, volume 141-3. 61

McDonald, R. P. (1974). Testing pattern hypotheses for covariance matrices. *Psychometrika*, 39(2):189–201. 129

McMaster, Robert, B. and Shea, K. S. (1992). *Generalization in digital cartography*. 32

Meadows, D., Randers, J., and Meadows, D. (2004). *Limits to Growth: The 30-year Update*. Earthscan. 3

Meek, D. S. and Walton, D. J. (1992). Clothoid spline transition spirals. *Mathematics of Computation*, 59(199):117–117. 35

Merriaux, P., Dupuis, Y., Vasseur, P., and Savatier, X. (2015). Fast and robust vehicle positioning on graph-based representation of drivable maps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2787–2793. IEEE. 79

Mitchell, H. B. (2012). *Data Fusion: Concepts and Ideas*. Springer Berlin Heidelberg, Berlin, Heidelberg. 11

National Coordination Office for Space-Based Positioning Navigation and Timing PNT (2018). GPS/Space Segment. `https://www.gps.gov/systems/gps/space/`. 19

Navstar (2013). Navstar GPS space Segment/Navigation User Interface. *Navstar GPS space Segment*, page 213. 23

Nebot, E. M. and Durrant-Whyte, H. (1999). A high integrity navigation architecture for outdoor autonomous vehicles. *Robotics and Autonomous Systems*, 26(2-3):81–97. xvi, 107, 108, 109, 112

NovAtel (2018). NovAtel SPAN-CPT. `https://www.novatel.com/products/span-gnss-inertial-systems/span-combined-systems/span-cpt/`. 50

Observatoire National Interministériel de la Sécurité Routière ONISR (2017). Accidentalité routière 2017 – résultats définitifs. Technical report, Observatoire National Interministériel de la Sécurité Routière. xxi, 3, 5

Observatoire National Interministériel de la Sécurité Routière ONISR (2018). ATBH mensuels. `http://www.securite-routiere.gouv.fr/la-securite-routiere/l-observatoire-national-interministeriel-de-la-securite-routiere/series-statistiques/series-longues`. xiii, 4

Ochieng, W. Y., Quddus, M. A., and Noland, R. B. (2003). Map-Matching in Complex Urban Road Networks. *Brazilian Journal of Cartography Revista Brasileira de Cartografia*, 55(2):1–18. 44

Oh, S. M., Tariq, S., Walker, B. N., and Dellaert, F. (2004). Map-based Priors for Localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2179–2184. 68

Parker, T. E. and Matsakis, D. (2004). Time and Frequency Dissemination – Advances in GPS Transfer Techniques. *GPS world Magazine*, pages 32–38. 21

Pearson, K. (1900). X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175. 106

Peker, A. U., Tosun, O., and Acarman, T. (2011). Particle filter vehicle localization and map-matching using map topology. In *IEEE Intelligent Vehicles Symposium, Proceedings*, number Iv, pages 248–253. 45, 68

Petridou, E. and Moustaki, M. (2000). Human Factors in the Causation of Road Traffic Crashes. *European Journal of Epidemiology*, 16(9):819–826. 5

Plainis, S., Murray, I. J., and Pallikaris, I. G. (2006). Road traffic casualties: Understanding the night-time death toll. *Injury Prevention*, 12(2):125–128. 5

Quddus, M. A., Noland, R. B., and Ochieng, W. Y. (2006a). A High Accuracy Fuzzy Logic Based Map Matching Algorithm for Road Transport. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 10(3):103–115. xiv, 44, 45

Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2006b). Integrity of map-matching algorithms. *Transportation Research Part C: Emerging Technologies*, 14(4):283–302. 47, 69

Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328. 46, 69

Rabe, J., Hubner, M., Necker, M., and Stiller, C. (2017). Ego-lane estimation for downtown lane-level navigation. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1152–1157. IEEE. 37, 47, 69, 82

RATP (2018). RATP Plan de la ligne 4. `https://www.ratp.fr/plans-lignes/metro/4`. 33

Renault (2013). Renault UMAP. `https://group.renault.com/en/news/blog-renault/pamu-valet-parking-of-the-future/`. 40

Renault (2017). Renault Symbioz Concept. `https://www.renault.fr/vehicules/concept-car/symbioz-concept-car.html`. xiii, 8

Renault (2018a). Renault EZ-GO Concept. `https://www.renault.fr/vehicules/concept-car/ez-go-concept.html`. xiii, 8

Renault (2018b). Renault Megane Features. `https://www.renault.co.uk/vehicles/new-vehicles/megane/features.html`. xiii, 5, 6

Rencher, A. C. (2002). *Methods of Multivariate Analysis*, volume 37 of *Wiley Series in Probability and Statistics*. John Wiley & Sons, Inc., New York, NY, USA. 129

Ristic, B., Arulampalam, M. S., and Gordon, N. (2004). *Beyond the kalman filter: Particle filters for tracking applications*. 58

Russian Space Systems JSC (2016). Interface Control Document. Technical report. 162

Saastamoinen, J. (1959). Contributions To The Theory Of Atmospheric Refraction. *Nature*, 184(4703):1916–1917. 23

SAE On-Road Automated Driving (ORAD) committee (2018). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Technical Report 2018_06, SAE International. xxi, 7, 9

SaPPART (2017). *SaPPART Handbook: Assessment of positioning performance in ITS applications*. 28

Schoettle, B. and Sivak, M. (2014). A Survey of Public Opinion about Autonomous and Self-Driving Vehicles in the U.S., the U.K., and Australia. Technical report, University of Michigan – Transportation Research Institute, Ann Arbor, Michigan. 7

Seeber, G. (2003). *Satellite Geodesy : foundations, methods, and applications (2nd ed)*. Walter de Gruyter GmbH & Co, 2 edition. 22

Selloum, A., Betaille, D., Le Carpentier, E., and Peyret, F. (2009). Lane level positioning using particle filtering. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE. 47, 69

Sétra (2006). Comprendre Les Principaux Paramètres De Conception Géométrique Des Routes. Technical report. 35

Shafer, G. R. (1976). *A Mathematical Theory of Evidence*. Princeton University Press. 11

Shao, X., Huang, B., and Lee, J. M. (2010). Constrained Bayesian state estimation - A comparative study and a new particle filter based approach. *Journal of Process Control*, 20(2):143–157. 63

Spilker, J. J. J. (1996). Troposheric Effects On Gps. In *Global Positioning System: Theory and Applications, Volume I*, pages 517–546. American Institute of Aeronautics and Astronautics, Washington DC. 23

SQLite (2018). SQLite Limits Documentation - https://www.sqlite.org/limits.html. 145

Stroustrup, B. (1991). What is "Object-Oriented Programming"? (1991 revised version). *European Software Festival*. 51

Szottka, I. (2013). Particle filtering for lane-level map-matching at road bifurcations. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 154–159. IEEE. 46, 47, 69, 77

Tao, Z. and Bonnifait, P. (2016). Sequential Data Fusion of GNSS Pseudoranges and Dopplers With Map-Based Vision Systems. *IEEE Transactions on Intelligent Vehicles*, 1(3):254–265. 80

Tao, Z., Bonnifait, P., Fremont, V., and Ibañez-Guzmán, J. (2013). Mapping and localization using GPS, lane markings and proprioceptive sensors. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 406–412. IEEE. 80, 91

Taylor, G., Blewitt, G., Steup, D., Corbett, S., and Car, A. (2001). Road Reduction Filtering for GPS-GIS Navigation. *Transactions in GIS*, 5(3):193–207. 44

Teunissen, P. (1990). Quality control in integrated navigation systems. *IEEE Aerospace and Electronic Systems Magazine*, 5(7):35–41. 104

Thrun, S. (2010). Toward robotic cars. *Communications of the ACM*, 53(4):99. 2

Thrun, S., Burgard, W., and Fox, D. (2002). *Probabilistic Robotics*. MIT Press. 31, 66

Toledo-Moreo, R., Betaille, D., and Peyret, F. (2010). Lane-Level Integrity Provision for Navigation and Map Matching With GNSS, Dead Reckoning, and Enhanced Maps. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):100–112. 30, 33, 38

Toledo-Moreo, R., Betaille, D., Peyret, F., and Laneurit, J. (2009). Fusing GNSS, Dead-Reckoning, and Enhanced Maps for Road Vehicle Lane-Level Navigation. *IEEE Journal of Selected Topics in Signal Processing*, 3(5):798–809. 69

Tossaint, M., Samson, J., Toran, F., Ventura-Traveset, J., Hernandez-Pajares, M., Juan, J., Sanz, J., and Ramos-Bosch, P. (2007). The Stanford - ESA Integrity Diagram: A New Tool for The User Domain SBAS Integrity Assessment. *Navigation*, 54(2):153–162. xiii, 28

Van Der Merwe, R., Doucet, A., De Freitas, N., and Wan, E. A. (2001). The Unscented Particle Filter. *Advances in Neural Information Processing Systems*, 96(6080):584–590. 64

Waymo (2018). Waymo Website. `https://waymo.com/ontheroad/`. 154

Wells, D., Beck, N., Delikaraoglu, D., Kleusberg, A., Krakiwsky, E. J., Lachaélle, G., Langley, R. B., Nakiboglu, M., Schwarz, K.-P., Tranquilla, J. M., and Vanicek, P. (1986). *Guide to GPS Positioning*. Department of Geodesy and Geomatics Engineering Lecture Note No. 58, University of New Brunswick, Fredericton, New Brunswick, Canada. 23

White, C. E., Bernstein, D., and Kornhauser, A. L. (2000). Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1-6):91–108. 44

William J. Hughes Technical Center (2017). Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report. Technical Report 96, Federal Aviation Administration, Washington, DC. xiii, 26, 27

Xu, P., Dherbomez, G., Hery, E., Abidli, A., and Bonnifait, P. (2018). System Architecture of a Driverless Electric Car in the Grand Cooperative Driving Challenge. *IEEE Intelligent Transportation Systems Magazine*, 10(1):47–59. 49

Zadeh, L. A. (1989). Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):89–100. 44

Zhang, W. (2010). LIDAR-based road and road-edge detection. In *2010 IEEE Intelligent Vehicles Symposium*, pages 845–848. IEEE. 71

Zhu, N., Marais, J., Betaille, D., and Berbineau, M. (2018). GNSS Position Integrity in Urban Environments: A Review of Literature. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2762–2778. 27

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Thao Dang, Franke, U., Appenrodt, N., Keller, C. G., Kaus, E., Herrtwich, R. G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knoppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., and Zeeb, E. (2014). Making Bertha Drive-An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20. xiv, 36, 37

Zinoune, C., Bonnifait, P., and Ibañez-Guzmán, J. (2012). Detection of missing roundabouts in maps for Driving Assistance Systems. In *2012 IEEE Intelligent Vehicles Symposium*, pages 123–128. IEEE. 2, 13

Zumberge, J. and Gendt, G. (2001). The demise of selective availability and implications for the international GPS service. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, 26(6-8):637–644. 19