Reliable localization methods for intelligent vehicles based on environment perception

Francisco Miguel Moreno Olivo

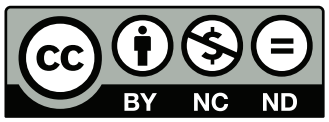
A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

Universidad Carlos III de Madrid

Advisor and Tutor
Dr. Fernando García Fernández

June, 2022



Agradecimientos

Cada paso que he tomado en la vida me ha conducido a donde estoy hoy, y me gustaría agradecer a todas las personas que me han acompañado en este camino; tanto en los buenos momentos, como en las cuestas arriba.

Aunque estos últimos años han venido cargados de curvas, he tenido la suerte de compartir grupo de investigación con personas excepcionales de las que he aprendido mucho, y que se han convertido en grandes amigos. Con ellos he compartido largas jornadas de trabajo, estupendos momentos de ocio, proyectos extraordinarios, y viajes increíbles por tierra, mar, e incluso aire. Este buen ambiente fue uno de los motivos por los que finalmente me decidí a embarcarme en esta tesis, y gracias a vuestra compañía he conseguido mantenerme cuerdo y no perderme por el camino. Y aunque no pueda mencionar a todo el mundo, en realidad no necesito hacerlo... porque vosotros ya sabéis quiénes sois.

Dentro de este gran grupo de increíbles personas, he de dar mil gracias a mi director de tesis, Fernando, por haber estado ahí en todo momento, y por haber cuidado de mi y de que esta tesis saliese adelante. Por haber sido capaz de llevarlo todo a buen puerto, aún remando a contracorriente, se merece un monumento. Y además, me gustaría agradecer especialmente a Ahmed, que aunque no aparezca como co-director de esta tesis, ha formado parte de ella desde el principio y le tengo mucho aprecio como amigo y mentor.

Por supuesto, nunca se me podrían olvidar los agradecimientos a mi familia, ya que de una forma o de otra siempre he tenido su apoyo incondicional. A mi padre, por la motivación de intentar conseguir que con mi trabajo algún día deje de existir el suyo. A mi madre, por inculcarme desde pequeño la importancia de la educación. Y a mi hermana, por enseñarme que los títulos no lo son todo en esta vida.

Y finalmente, a Nuria, por enseñarme a ser humano y por aguantarme más de lo que me merezco. Sin ella, esta tesis no existiría y nadie podría leer estas palabras.

A todos, siempre os llevaré conmigo, sea cual sea la dirección que tome en el futuro mi camino.

Francisco Miguel Moreno Olivo
Leganés, junio de 2022

Published and submitted content

Some ideas, figures, and tables used in this thesis have appeared previously in the following publications:

Journal articles

- Francisco Miguel Moreno, Carlos Guindel, José María Armingol, and Fernando García. “Study of the Effect of Exploiting 3D Semantic Segmentation in LiDAR Odometry”. In: *Applied Sciences* 10.16 (2020), p. 5657

  Partially included in the thesis: Chapter 3. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- Miguel Ángel de Miguel, Francisco Miguel Moreno, Pablo Marín-Plaza, Abdulla Al-Kaff, Martín Palos, David Martín, Rodrigo Encinar-Martín, and Fernando García. “A Research Platform for Autonomous Vehicles Technologies Research in the Insurance Sector”. In: *Applied Sciences* 10.16 (2020)

  Partially included in the thesis: Chapter 3. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

Conference articles

- Francisco Miguel Moreno, Ahmed Hussein, and Fernando Garcia. “Landmark Placement Optimization for Accurate Localization in Autonomous Vehicles”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 128–134

  Partially included in the thesis: Chapter 4. The inclusion in the thesis of

material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

Other Research Merits

Some ideas, figures, and tables used in this thesis have appeared previously in the following publications:

Additional Published Content

Patents

- Fabian Pucks, Roman Blaschek, Stefan Rechenberger, Ahmed Hussein, and Francisco Miguel Moreno. "Verfahren, Steuergerät und Fahrzeug zur Validierung der Lokalisierung eines Fahrzeuges durch Referenzierung zu statischen Objekten im Umfeld". DE 10 2020 132 397.2. Dec. 2020

Journal articles

- Abdulla Al-Kaff, María José Gómez-Silva, Francisco Miguel Moreno, Arturo De La Escalera, and José María Armingol. "An appearance-based tracking algorithm for aerial search and rescue purposes". In: *Sensors* 19.3 (2019), p. 652
- Pablo Marin-Plaza, David Yagüe, Francisco Royo, Miguel Ángel de Miguel, Francisco Miguel Moreno, Alejandro Ruiz-de-la-Cuadra, Fernando Viadero-Monasterio, Javier Garcia, José Luis San Roman, and José María Armingol. "Project ARES: Driverless transportation system. challenges and approaches in an unstructured road". In: *Electronics* 10.15 (2021), p. 1753

Book chapters

- Abdulla Al-Kaff, Francisco Miguel Moreno, and Ahmed Hussein. "Ros-based approach for unmanned vehicles in civil applications". In: *Robot Operating System (ROS)*. Springer, 2019, pp. 155–183

Conference articles

- Bahae Abidi, Francisco Miguel Moreno, Mohamed El Haziti, Ahmed Hussein, Abdulla Al Kaff, and David Martin Gomez. "Hybrid V2X Communication Approach using WiFi and 4G Connections". In: *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2018, pp. 1–5

- Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando García, and Arturo De La Escalera. "BirdNet: A 3D Object Detection Framework from LiDAR Information". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3517–3523

- Mostafa Osman, Ricardo Alonso, Ahmed Hammam, Francisco Miguel Moreno, Abdulla Al-Kaff, and Ahmed Hussein. "Multisensor Fusion Localization using Extended Hinf Filter using Pre-filtered Sensors Measurements". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1139–1144

- Francisco Miguel Moreno, Omar El-Sobky, Fernando Garcia, and Jose Maria Armingol. "Hypergrid: A Hyper-Fast ROS-Based Framework for Local Map Generation". In: *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2019, pp. 1–6

- Abdulla Al-Kaff, Angel Madridano, Ahmed Radwan, Francisco Miguel Moreno, and Ahmed Hussein. "Heterogeneous Multiple Vehicles Cooperation Approach for Smart Roads". In: *11th International Micro Air Vehicle Competition and Conference (IMAVS)*. 2019

- Miguel Ángel de Miguel, Francisco Miguel Moreno, Fernando García, Jose María Armingol, and Rodrigo Encinar Martin. "Autonomous vehicle architecture for high automation". In: *International Conference on Computer Aided Systems Theory*. Springer. 2019, pp. 145–152

- Carlos Justo de Frías, Abdulla Al-Kaff, Francisco Miguel Moreno, Ángel Madridano, and José María Armingol. "Intelligent Cooperative System for Traffic Monitoring in Smart Cities". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 33–38

- Walter Morales Alvarez, Francisco Miguel Moreno, Oscar Sipele, Nikita Smirnov, and Cristina Olaverri-Monreal. "Autonomous Driving: Framework for Pedestrian Intention Estimation in a Real World Scenario". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 39–44

Research Visits

- Research visit to IAV Automotive Engineering in Berlin, Germany, under the supervision of Dr.-Ing. Ahmed Hussein, from 21/09/2020 to 29/01/2021.

Supervised Academic Works

Bachelor Theses

- O. Mohamed, "Generation and Analysis of Occupancy Grid Maps for Ground Autonomous Vehicles", Bachelor's Thesis, Bachelor's Degree in Computer Science and Engineering, German University in Cairo, Jul. 2018.
- J. Velado Núñez, "Ego lane localization for autonomous vehicles", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Jul. 2019.
- O. El-Sobky, "Local-Map Generation and Sensor Fusion in Real-Time", Bachelor's Thesis, Bachelor's Degree in Computer Science and Engineering, German University in Cairo, Jul. 2019.

Abstract

In the near past, we would see autonomous vehicles and Intelligent Transport Systems (ITS) as a potential future of transportation. Today, thanks to all the technological advances in recent years, the feasibility of such systems is no longer a question. Some of these autonomous driving technologies are already sharing our roads, and even commercial vehicles are including more Advanced Driver-Assistance Systems (ADAS) over the years. As a result, transportation is becoming more efficient and the roads are considerably safer.

One of the fundamental pillars of an autonomous system is self-localization. An accurate and reliable estimation of the vehicle's pose in the world is essential to navigation. Within the context of outdoor vehicles, the Global Navigation Satellite System (GNSS) is the predominant localization system. However, these systems are far from perfect, and their performance is degraded in environments with limited satellite visibility. Additionally, their dependence on the environment can make them unreliable if it were to change.

Accordingly, the goal of this thesis is to exploit the perception of the environment to enhance localization systems in intelligent vehicles, with special attention to their reliability. To this end, this thesis presents several contributions: First, a study on exploiting 3D semantic information in LiDAR odometry is presented, providing interesting insights regarding the contribution to the odometry output of each type of element in the scene. The experimental results have been obtained using a public dataset and validated on a real-world platform. Second, a method to estimate the localization error using landmark detections is proposed, which is later on exploited by a landmark placement optimization algorithm. This method, which has been validated in a simulation environment, is able to determine a set of landmarks so the localization error never exceeds a predefined limit. Finally, a cooperative localization algorithm based on a Genetic Particle Filter is proposed to utilize vehicle detections in order to enhance the estimation provided by GNSS systems. Multiple experiments are carried out in different simulation environments to validate the proposed method.

Resumen

En un pasado no muy lejano, los vehículos autónomos y los Sistemas Inteligentes del Transporte (ITS) se veían como un futuro para el transporte con gran potencial. Hoy, gracias a todos los avances tecnológicos de los últimos años, la viabilidad de estos sistemas ha dejado de ser una incógnita. Algunas de estas tecnologías de conducción autónoma ya están compartiendo nuestras carreteras, e incluso los vehículos comerciales cada vez incluyen más Sistemas Avanzados de Asistencia a la Conducción (ADAS) con el paso de los años. Como resultado, el transporte es cada vez más eficiente y las carreteras son considerablemente más seguras.

Uno de los pilares fundamentales de un sistema autónomo es la autolocalización. Una estimación precisa y fiable de la posición del vehículo en el mundo es esencial para la navegación. En el contexto de los vehículos circulando en exteriores, el Sistema Global de Navegación por Satélite (GNSS) es el sistema de localización predominante. Sin embargo, estos sistemas están lejos de ser perfectos, y su rendimiento se degrada en entornos donde la visibilidad de los satélites es limitada. Además, los cambios en el entorno pueden provocar cambios en la estimación, lo que los hace poco fiables en ciertas situaciones.

Por ello, el objetivo de esta tesis es utilizar la percepción del entorno para mejorar los sistemas de localización en vehículos inteligentes, con una especial atención a la fiabilidad de estos sistemas. Para ello, esta tesis presenta varias aportaciones: En primer lugar, se presenta un estudio sobre cómo aprovechar la información semántica 3D en la odometría LiDAR, generando una base de conocimiento sobre la contribución de cada tipo de elemento del entorno a la salida de la odometría. Los resultados experimentales se han obtenido utilizando una base de datos pública y se han validado en una plataforma de conducción del mundo real. En segundo lugar, se propone un método para estimar el error de localización utilizando detecciones de puntos de referencia, que posteriormente es explotado por un algoritmo de optimización de posicionamiento de puntos de referencia. Este método, que ha sido validado en un entorno de simulación, es capaz de determinar un conjunto de puntos de referencia para el cual el error de localización nunca supere un límite previamente fijado. Por último, se propone un algoritmo de localización cooperativa basado en un Filtro Genético de Partículas para utilizar las detecciones de vehículos con el fin de mejorar la estimación proporcionada por los sistemas GNSS. El método propuesto ha sido validado mediante múltiples experimentos en diferentes entornos de simulación.

Contents

List of Acronyms xix

List of Figures xxiii

List of Tables xxvii

List of Algorithms xxix

1 Introduction 1
- 1.1 Road safety . . . 2
  - 1.1.1 Vehicle safety systems . . . 3
  - 1.1.2 Advanced Driver-Assistance Systems . . . 4
- 1.2 Autonomous vehicles . . . 5
  - 1.2.1 Control and path planning systems . . . 7
  - 1.2.2 Perception systems . . . 8
  - 1.2.3 Localization systems . . . 9
- 1.3 Motivation . . . 10
- 1.4 Objectives . . . 12
- 1.5 Thesis Structure . . . 13

2 Related Work 15
- 2.1 Autonomous driving platforms . . . 15
- 2.2 Perception technologies . . . 18
- 2.3 Self localization . . . 21
  - 2.3.1 Wheel odometry . . . 23
  - 2.3.2 IMU odometry . . . 24
  - 2.3.3 Visual odometry . . . 24
  - 2.3.4 LiDAR odometry . . . 25
  - 2.3.5 GNSS-based localization solutions . . . 27
  - 2.3.6 Landmark-based localization solutions . . . 28
- 2.4 Data filtering and fusion . . . 30
- 2.5 Localization evaluation systems and metrics . . . 33
- 2.6 Cooperative systems . . . 37
  - 2.6.1 Communication technologies . . . 37

2.6.2 Cooperative algorithms . . . 38
2.7 Concluding remarks . . . 40

3 LiDAR Odometry Based on 3D Semantic Information 41
3.1 Introduction . . . 41
3.1.1 Odometry algorithm . . . 43
3.2 Methodology . . . 43
3.2.1 Data . . . 43
3.2.2 Input filter configurations . . . 44
3.2.3 Evaluation metrics . . . 46
3.3 Results and discussion . . . 47
3.3.1 Results . . . 47
3.3.2 Discussion . . . 51
3.4 Real-world validation . . . 55
3.4.1 Research driving platform: ATLAS . . . 55
3.4.2 3D semantic segmentation . . . 55
3.4.3 Experimental results . . . 57
3.5 Conclusions . . . 60

4 Validation of Localization Systems 63
4.1 Problem statement . . . 64
4.2 Proposed approach . . . 67
4.2.1 Simulation environment . . . 68
4.2.2 Landmark detection algorithm . . . 69
4.2.3 Landmark measurement model . . . 72
4.2.4 Visibility model . . . 80
4.2.5 Landmark-based localization . . . 80
4.2.6 Accuracy metric . . . 82
4.2.7 Landmark placement optimization . . . 84
4.3 Experiments and results . . . 90
4.3.1 Experimental setup . . . 91
4.3.2 Results . . . 94
4.4 Concluding remarks . . . 98

5 Cooperative Localization 101
5.1 Introduction . . . 101
5.2 Proposed approach . . . 104
5.2.1 Problem statement . . . 104
5.2.2 LoCo framework . . . 105
5.2.3 Particle Filter . . . 108

5.2.4 Genetic resampling 112
5.3 Experimental results 115
5.3.1 Simulation environments 116
5.3.2 Processing time 121
5.3.3 Parameter tuning 124
5.3.4 Genetic resample validation 130
5.3.5 Performance analysis 133
5.4 Concluding remarks 136

6 Conclusion and Future Work **137**
6.1 Conclusion 137
6.2 Future Work 139

Bibliography **141**

List of Acronyms

ABS Anti-Lock Braking System. 1, 3, 4

ACC Adaptive Cruise Control. 1, 5, 8

ADAS Advanced Driver-Assistance Systems. 1, 4, 5

APE Absolute Pose Error. 1, 34–36, 46, 47, 49, 54, 59, 60, 96, 125, 134, 135

CNN Convolutional Neural Network. 1, 26, 55

DARPA Defense Advanced Research Projects Agency. 1, 15–18

DDS Data Distribution Service. 1, 105, 106

DOF Degrees of Freedom. 1, 21, 33

ECU Electronic Control Unit. 1

EKF Extended Kalman Filter. 1, 31, 32, 39

ESC Electronic Stability Control. 1, 3, 4

GDOP Geometric Dilution of Precision. 1, 11, 64

GNSS Global Navigation Satellite System. 1, 5, 9–12, 16, 17, 24, 27–30, 32–34, 36, 37, 39, 40, 55, 57, 63, 64, 101–104, 110, 114, 115, 117, 120, 125–132, 134–137, 139, 140

GPUs Graphical Procesing Units. 1

ICP Iterative Closest Point. 1, 26

IMLS Implicit Moving Least Squares. 1, 26

IMU Inertial Measurement Unit. 1, 9, 16, 23, 24, 27, 29, 32, 33, 36, 37, 55, 57, 120, 134

IoT Internet of Things. 1, 37, 101

ITS Intelligent Transport Systems. 1, 5, 101

LiDAR Light Detection and Ranging. 1, 4, 9–13, 15, 16, 19–21, 23, 25–27, 32, 33, 38, 41–46, 50, 54–57, 60, 61, 66, 68–71, 73, 74, 80, 94, 120, 138, 140

LSTM Long Short-Term Memory. 1, 26

MCMC Markov Chain Monte Carlo. 1, 83

Mo-Cap Motion Capture. 1, 12, 33, 34, 63, 64

MPC Model Predictive Control. 1, 8

NDT Normal Distributions Transform. 1, 25

NLLS Non-Linear Least Squares. 1, 28, 81–83, 85, 86, 97

NTP Network Time Protocol. 1, 107

OICA International Organization of Motor Vehicle Manufacturers. 1

PID Proportional Integral Derivative. 1, 8

RNNs Recurrent Neural Networks. 1, 26

ROS Robotic Operating System. 1, 68, 69, 94, 134

RPE Relative Pose Error. 1, 35–37, 46, 47, 49, 54, 59

RSSI Received Signal Strength Indicator. 1, 39, 40, 103

RTK Real-Time Kinematic. 1, 11, 27, 28, 34, 36, 37, 55, 63, 127, 128

SLAM Simulatneous Localization and Mapping. 1, 24, 26, 27, 29, 34, 41–43, 46

SVM Support Vector Machine. 1, 20, 21

UAVs Umnanned Aerial Vehicles. 1, 21, 22, 27, 33

UKF Unscented Kalman Filter. 1, 16, 31, 32, 55, 133–136

UWB Ultra Wide Band. 1, 32, 40

VANET Vehicular Ad-Hoc Network. 1, 103

VRU Vulnerable Road Users. 1, 2, 37, 38, 102

WHO World Health Organization. 1, 2

List of Figures

1.1 Vehicles in use by world region, according to OICA data [19]. 2
1.2 Number of motor vehicles and rate of road traffic death per 100,000 vehicles: 2000-2016 [22]. 3
1.3 Comparison of the number of vehicle crashes, injured people and fatalities in traffic accidents. EU, 2010-2019 4
1.4 Levels of driving automation proposed by SAE International [31]. 7

2.1 Stanley vehicle from Stanford University in DARPA Grand Challenge 2005. ©2005 Standford University. 16
2.2 BOSS vehicle from Carnegie Melon University in DARPA Urban Challenge 2007 [54]. ©2009 Springer. 17
2.3 Problematic of the APE metric. The green and blue circles represent the reference and estimated poses, respectively. The error of each pair of poses is drawn as a red arrow. 35

3.1 Samples of filtered point clouds from different test configurations. Letters (A, B, E, F) refer to the filtering configuration used, described in Section 3.2.2. 42
3.2 Labelled classes in Semantic-KITTI dataset [262] and number of points from each class. *Image source: semantic-kitti.org/dataset*. 44
3.3 Point cloud downsizing for each filtering configurations. 46
3.4 Distribution of RPE translation metric for each filtering configuration across all KITTI odometry sequences. 50
3.5 Distribution of RPE rotation metric for each filtering configuration across all KITTI odometry sequences. 50
3.6 Computational time required by LOAM to process each measurement for each of the filtering configurations. 51
3.7 Distance to ground points and distance to object points after vehicle translation. 54
3.8 ATLAS research platform. 56
3.9 3D semantic segmentation in ATLAS research platform. 57
3.10 Driving environment from the vehicle's perspective. 58
3.11 Results for each localization metric on the ATLAS experiment. 59

4.1 Gazebo simulation environment. 68
4.2 Lamp post model used as a pole-like landmark. 69
4.3 Pole-like landmark detection process. 70
4.4 Offset between center of mass and actual pole center. 71
4.5 Simulation environment to validate the landmark detection algorithm. . 72
4.6 Every landmark position used in the measurement model calibration process. 75
4.7 Angle error with respect to the landmark angle. Each point corresponds with the average error of each landmark position. Points are colored depending on the distance to the landmark. 76
4.8 Distance error mean and standard deviation with respect to the landmark angle. Each point corresponds with the mean (top) and standard deviation (bottom) of all measurements from each landmark position. Points are colored depending on the distance to the landmark. 76
4.9 Angle error and standard deviation with respect to the distance to the landmark. Each point in the top subplot corresponds with the mean of all measurements from each landmark position. The subplot at the bottom shows the standard deviation of all the average error points displayed above. 77
4.10 Distance error and standard deviation with respect to the distance to the landmark. Each point in the top subplot corresponds with the mean of all measurements from each landmark position. The subplot at the bottom shows the standard deviation of all the average error points displayed above and the linear model of the standard deviation, estimated by linear regression. 78
4.11 Histogram of distance error at different distances. 79
4.12 Left: Received measurement from $\mathbf{x}$ and expected measurements from $\hat{\mathbf{x}}$. Right: Difference (error) $\mathbf{e}_i$ between received and expected measurements, with respect to $\hat{\mathbf{x}}$. 81
4.13 Sampling the posterior of the NLLS pose estimation. 84
4.14 Accuracy heatmap for one of the testing environments. The color of the cells represent the standard deviation of the localization error. 85
4.15 Coverage map for one of the testing environments. The value of each cell represents the number of visible landmarks from that point. 86
4.16 Local optimization process. 89

4.17 Score maps for the auxiliary greedy algorithm. Brighter cells (more orange) have a higher score. The top map is empty; hence the score is higher towards the center with cells with higher coverage. The bottom map already has some landmarks, so the score is higher in the right area, which is less populated. 91
4.18 Gazebo environment for scenario 1. 92
4.19 Gazebo environment for scenario 2. 92
4.20 Occupancy grid map for scenario 2. 93
4.21 Gazebo environment used in the validation experiments of scenario 2. . 94
4.22 Accuracy and coverage maps obtained with the landmark configuration calculated for scenario 2. 95
4.23 Validation trajectories followed by the vehicle in scenario 1. Landmarks are displayed as magenta triangles. 96
4.24 Validation trajectories followed by the vehicle in scenario 2. Landmarks are displayed as magenta triangles. 97

5.1 Proposed cooperative localization approach. 105
5.2 Inputs and outputs of the proposed cooperative localization method. . . 106
5.3 Population state encoding for the Genetic Particle Filter. 110
5.4 Genetic crossover example. 113
5.5 Genetic mutation example. 114
5.6 *Agent emulator* environment. 117
5.7 Carla simulation scenario 1: Straight line. 119
5.8 Carla simulation scenario 2: Curved line. 119
5.9 Carla simulation scenario 3: Real-world location. 120
5.10 Total processing time of each iteration, averaged for 1 minute-long sequences. 123
5.11 Processing time of the likelihood process, averaged for 1 minute-long sequences. 124
5.12 Average error mean for the initial tuning test, averaged on 10 sequences. 126
5.13 Average error mean for the second tuning test, divided by GNSS noise and averaged on 10 sequences. 128
5.14 Average error std for the second tuning test, divided by GNSS noise and averaged on 10 sequences. 129
5.15 Error mean analysis with $M = 0$, divided by GNSS noise and averaged on 10 sequences. 130
5.16 Error distribution with each resampling approach. 132
5.17 Error distribution with each resampling approach (zoomed). 133

5.18 Error distribution in Carla's simulation experiments. 135

List of Tables

3.1 Absolute Pose Error (APE) obtained for each configuration on the available KITTI odometry sequences. 48
3.2 Relative Pose Error (RPE) obtained for each configuration on the available KITTI odometry sequences. 48
3.3 Best filtering configuration grouped for each metric and each KITTI odometry sequence. 49
3.4 Results for each error metric on the ATLAS experiment. 58

4.1 Translation error of each validation sequence. 97
4.2 Rotation error of each validation sequence. 98

5.1 GNSS noise settings. 127
5.2 Performance results on Carla's simulations. 135

List of Algorithms

1 Genetic algorithm for the landmark placement problem 88

2 Particle Filter process 111

1 Introduction

> *Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.*
>
> **— Marie Curie**

Over the years in human history, the ability to move efficiently from one point to another has always been of great importance. Since the invention of the wheel, we have been continuously improving our techniques and developing new technologies, creating new means of transport and allowing us to travel, commerce, and discover the world. Transportation is one of the pillars of the modern economy, where the production and consumption/usage of goods are usually geographically separated [16]. Furthermore, a high percentage of transportation is done by ground vehicles, most of it being carried by road. As presented in the latest report on energy, transport, and environment statistics by the European Union [17], road transportation accounted for 75.3% of the total inland freight transportation. Moreover, the total volume of transported goods kept increasing in most countries during the last ten years, according to a study performed by the European Union [18].

Nevertheless, we are not using our roads for transport purposes only. In addition to freight transportation, we are familiar with covering great distances daily: to commute to/from work, to visit family or friends, or even to buy groceries. Personal cars are used every day all around the globe, and their numbers are only increasing. According to the data published by the International Organization of Motor Vehicle Manufacturers (OICA), in 2015, there were more than 1200 million motor vehicles in use [19]. Figure 1.1 shows the growth by region in the period available from the OICA website. Unfortunately, this is the latest data available regarding vehicles in use, which was presented in 2015.

In the European Union, car ownership is increasing as well, with an average of around one car for every two persons in 2019 [20]. Regarding the distance traveled by car, although it is slowly decreasing, the average European still travels 11,300 km per year by car [21].

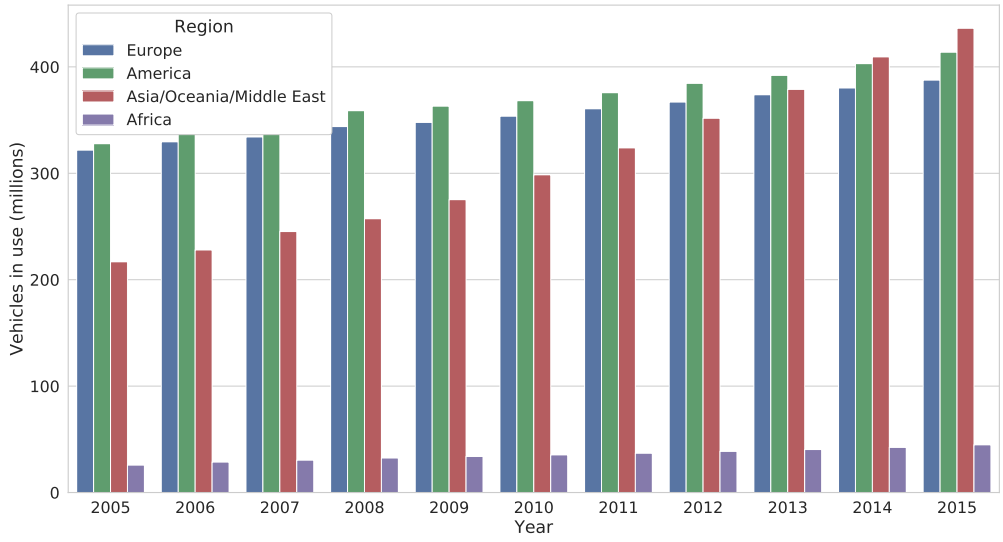


Fig. 1.1: Vehicles in use by world region, according to OICA data [19].

1.1 Road safety

Despite all the benefits and advantages that motor vehicles grant us, they come with a significant problem: road accidents. According to the latest global status report on road safety from the World Health Organization (WHO) in 2018 [22], road traffic injuries are the 8th most common cause of death in the world. Note that, albeit old, this is the latest report available on global road safety by the WHO. This tragic statistic is strongly related to the rise in the number of vehicles in use; presented before. For this reason, among many others, vehicle manufacturers are constantly improving the safety of vehicles, not only for their occupants but also for the Vulnerable Road Users (VRU). This is reflected in Figure 1.2, also from [22], which shows that the rate of traffic related deaths per 100,000 vehicles is in recession. Nonetheless, the total number of deaths remains unacceptable, with 1.35 million people dying each year on our roads due to traffic accidents.

If the focus is placed on the European Union, the numbers are more optimistic. The latest report on road safety [23] shows that the number of crashes involving vehicles, as well as the number of injured people and deaths, are all decreasing. Figure 1.3 presents a comparison between these variables, where their decrease becomes clear. Although the death toll of traffic accidents is still a considerable problem, the advances in safety systems for vehicles are helping to reduce it.

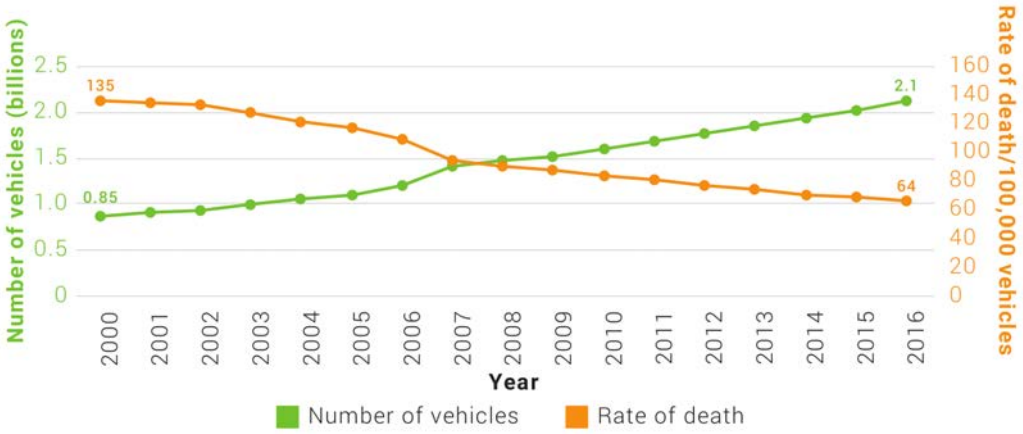


Fig. 1.2: Number of motor vehicles and rate of road traffic death per 100,000 vehicles: 2000-2016 [22].

1.1.1 Vehicle safety systems

Since the invention of the first automobile in 1886, the Benz Patent-Motorwagen [24], motor vehicles have received several improvements in terms of safety. Next, the most relevant safety features introduced in cars (the most common motor vehicle) in the latest decades are presented:

- **Seat belts**: One of the first safety features for cars, from the early 1900s. The initial seatbelt invention consisted of a simple two-point design around the waist. Then Volvo patented the three-point seatbelt in 1959 [25], which quickly became a standard in all cars after that patent was released for free.

- **Anti-Lock Braking System (ABS)**: The ABS was initially developed for aircraft; in order to prevent sliding during landing due to the wheels locking up. This technology began to be used in cars in the 1970s, becoming a standard in the late 1980s.

- **Crumple zones**: Crumple zones turn the structure of the car into a protective shell, which is crushed in case of impact, absorbing the crash energy and protecting the passengers.

- **Electronic Stability Control (ESC)**: The advances in electronics in the latest years made possible new safety features for vehicles, such as the ESC. The ESC is capable of detecting traction loss in a vehicle. Then, it automatically acts on the brake of each wheel in order to take stability control back.

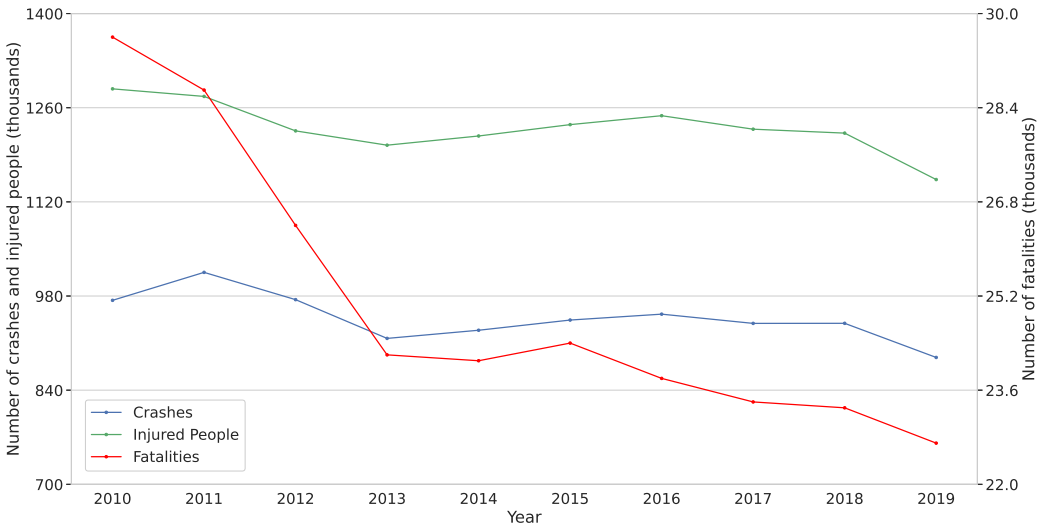


Fig. 1.3: Comparison of the number of vehicle crashes, injured people and fatalities in traffic accidents. EU, 2010-2019

1.1.2 Advanced Driver-Assistance Systems

Advanced Driver-Assistance Systems (ADAS) are the outcome of years of research on vehicle systems engineering, electronics, and computer science. They use sensors, such as cameras, radar, ultrasound, and Light Detection and Ranging (LiDAR) sensors. Although some ADAS features are primarily focused on driver comfort, most contribute actively to enhancing the vehicle's safety [26]. ADAS are classified into passive or active systems. Passive ADAS include the aforementioned ABS and ESC safety systems, in addition to other features such as the following:

- **Back-up Camera**: Helps the driver in parking maneuvers by providing a view from the back of the vehicle.
- **Lane departure warning**: The vehicle is capable of detecting the road lines using cameras; in order to warn the driver if the vehicle is not keeping within its lane.
- **Forward collision warning**: Radar and LiDAR sensors can be used to detect obstacles and other objects like vehicles, cyclists, and pedestrians. If there is a risk of collision, the vehicle system alerts the driver.
- **Blind spot detection**: Sometimes, the driver cannot see other vehicles passing around it through the mirrors because of blind spots. This issue is mitigated

by using sensors to detect vehicles in these areas and inform the driver about their existence.

- **Driver Monitoring**: Danger is not only outside of the vehicle, as the driver can also malfunction. A monitoring system inside the vehicle can sense if the driver is distracted or falling asleep, triggering a warning to keep attention on the road [27].

- **Navigation System**: Modern navigation systems acquire the position of the vehicle via Global Navigation Satellite System (GNSS) and can compute the best route considering the map and the current traffic levels. These systems help the driver find the optimal way to their destination while indirectly reducing congestion on the roads.

Other systems act directly on the car systems, taking partial control of some of them to help the driver control the vehicle or prevent dangerous situations:

- **Adaptive Cruise Control (ACC)**: A comfort feature in modern cars. The system takes control of the vehicle throttle and brake; in order to maintain a fixed velocity or match the velocity of the leading vehicle. The vehicle in front is commonly detected using a radar sensor.

- **Automatic Emergency Braking**: The evolution of the forward collision warning. Instead of merely warning the driver, this system operates directly on the brakes.

- **Lane Keeping Assistance**: Lane-keeping systems assist in controlling the steering wheel, ensuring that the vehicle stays within its lane.

1.2 Autonomous vehicles

Following the research on vehicle systems, and after assessing the benefits brought by ADAS, we can recognize that integrating sensing capabilities and automation in our vehicles makes them more comfortable and safe. One of the reasons is that these systems try to eliminate the driver from the control loop. The primary cause of traffic accidents is due to human errors. Drivers tend to get distracted by the phone, the radio, and drinking or eating, for example. Therefore, the most natural evolution of ADAS are Intelligent Transport Systems (ITS). The goal of ITS is to make vehicles more autonomous, thus reducing the driver's actions even further.

Nevertheless, making vehicles completely autonomous is not an easy task, as it requires handling very complex situations that might seem simple for humans but not for machines. Hence, the arrival of self-driving cars on our roads is expected to be progressive rather than instantaneous [28]. The reasons behind this are twofold: On the one hand, new technology must be developed and tested. Additionally, these systems rely on expensive sensors and equipment and will have to become affordable. On the other hand, the legislation around these vehicles advances slowly. Having autonomous vehicles roaming the world requires certification; to guarantee that the systems are safe. In addition, the vehicles will also need insurance, and there is still debate on who would be the responsible entity in case of an accident involving an autonomous vehicle [29].

In 2013, SAE International [1] proposed a taxonomy to classify the grade of autonomy in a vehicle that is composed of six levels [30]. The different levels, summarized in Figure 1.4, are as follow:

0. *Driver only*: The driver must take both lateral and longitudinal control. There is no automation.

1. *Assisted Driving*: The vehicle systems have longitudinal or lateral control. The driver is still continuously operating the vehicle.

2. *Partial automation*: The system can take both longitudinal and lateral control of the vehicle in some defined use cases, but the driver must always monitor the system.

3. *Conditional automation*: Same as level two, but now the driver does not need to monitor the system at all times. The system, however, might require the driver to resume manual control.

4. *High automation*: The vehicle systems can fully manage all situations defined in a specific use case. Inside that use case, the driver is not required.

5. *Full automation*: A fully autonomous vehicle does not require human intervention in any case and is able to handle all traffic situations in the journey without any specifically designed use case.

In order to reach high levels of autonomy, a vehicle must be equipped with several sensors and actuators that allow sensing the environment around it and performing actions accordingly. A review of the multiple sensors and actuators used in autonomous driving platforms can be found in [29].

[1] AE International is an automotive standardization entity based in the United States.

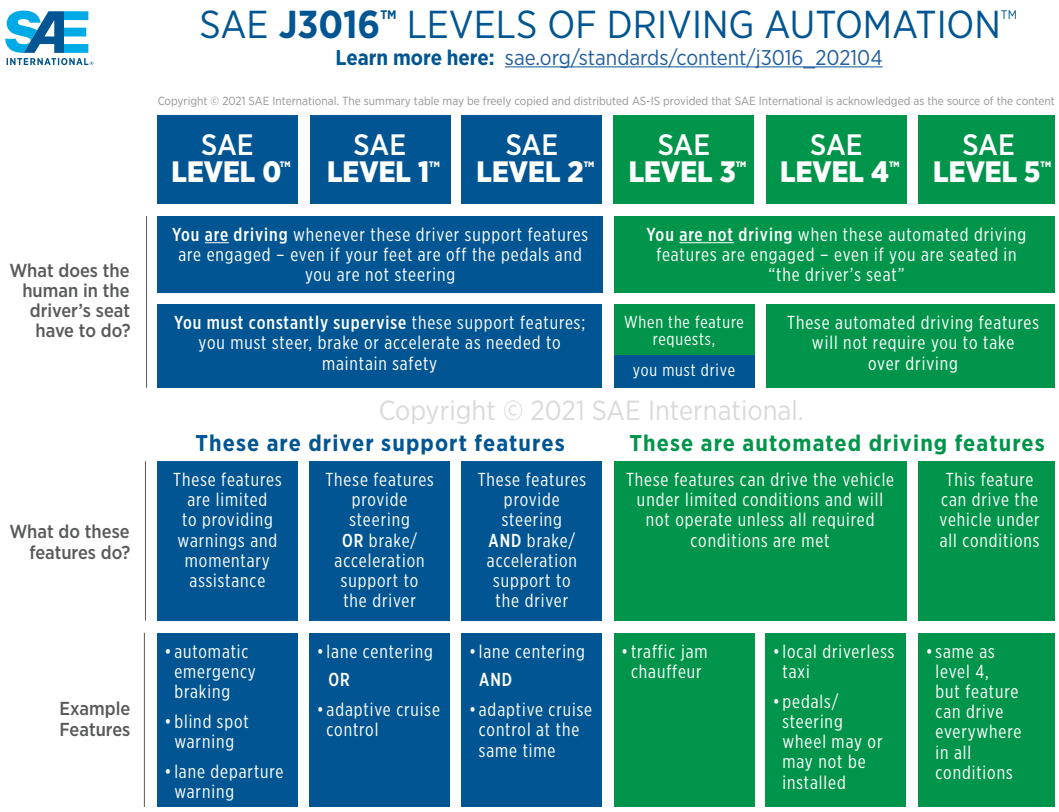


Fig. 1.4: Levels of driving automation proposed by SAE International [31].

The principal subsystems of an autonomous vehicle are the control, perception, and localization systems. The following sections will detail the main characteristics of each of these subsystems.

1.2.1 Control and path planning systems

The control subsystem of an autonomous vehicle can be divided into high-level and low-level control. While the former is responsible for the high-level navigation and the interactions between the vehicle and the world, the latter is responsible for the physical movement of the vehicle, following the goals directed by the high-level control.

Additionally, the low-level control of a vehicle can be divided into lateral and longitudinal control. On the one hand, longitudinal control systems control the velocity and acceleration of the vehicle through the throttle and brakes. The goal of these modules is to maintain a target velocity, keep a safe distance from the leading vehicle, or adjust the velocity in certain maneuvers when required. Moreover, the

vehicle's acceleration is also often controlled, so the movement is smooth, thus making the trip more comfortable for the passengers. An example of such a system is the ACC.

On the other hand, lateral control systems take control of the steering wheel and are responsible for keeping the vehicle within its lane (like Lane Keeping Assistance systems), changing lanes, or making turns in an intersection. Due to the complexity of the vehicle dynamics, lateral control is more difficult to solve than longitudinal control and requires more complex algorithms [32]. Multiple solutions to this problem are based on the dynamic and kinematic model of the vehicle [33, 34], using a Model Predictive Control (MPC) algorithm to predict the future state of the vehicle according to the current state and the models and taking the best control action to reach the desired goal. Other approaches operate without knowing the exact models of the vehicle movement, which is the case of the Proportional Integral Derivative (PID) controller or controllers based on fuzzy logic [35]. Furthermore, with the rise of techniques based on neural networks and learning algorithms, there are novel methods based on deep learning algorithms that are able to learn how to control the vehicle [36].

On top of the lateral and longitudinal control, we have higher-level control systems that are responsible for generating trajectories, performing complex maneuvers (e.g., lane changing or parking), and leading the vehicle to its destination [37].

1.2.2 Perception systems

Perceiving the environment, the road elements, other vehicles, pedestrians, cyclists, and unexpected objects on the road, and detecting the many threats that a vehicle might encounter on its way is not a straightforward task. Perception systems analyze the data acquired from the sensors to understand the scene and allow the vehicle to perform safe actions accordingly. The tasks performed by these systems include object detection and classification, semantic segmentation of the scene, data fusion, and object tracking.

3D object detection methods [38] provide the location and size of the detected objects relative to the ego vehicle. In addition, classification algorithms can also identify the class of the objects to determine if that object is a pedestrian, a traffic sign, or another vehicle. Most object detection modules can also identify the object's type.

A different approach to modeling the environment is through scene segmentation techniques. These can either be image [39] or pointcloud based [40]. If the data comes from a camera in image format, the output is another image where each pixel is assigned a label that represents the object's class. When working with 3D sensors like LiDAR or stereo cameras, it is common to obtain the data in a cloud of points, where each point represents each one of the 3D measurements. Then, 3D segmentation algorithms compute the object's class for each one of the points.

In addition to the complexity of object detection and classification tasks, perception algorithms must also be robust enough to work under any weather condition. Therefore, it is common to combine different sensor types, so the weakness of one sensor is covered by a different one [41]. The benefits of data fusion are many. Besides covering weaknesses, combining data from multiple sensors often produces better results than processing them independently because more information is used to solve the problems.

Finally, it is also important to remark that perception systems have grown immensely in the latest years due to advances in image processing and 3D scene understanding, and thanks to the advances in machine learning that have been adopted in the field of autonomous vehicle research.

1.2.3 Localization systems

Localization systems aim to answer the question: "Where am I?". Regarding intelligent vehicles, we could divide localization techniques into global and local.

Global localization Having a global localization system means having a shared reference system with the rest of the vehicles and elements of the environment. For instance, global latitude and longitude coordinates, or coordinates in a map, are good examples of global localization data. This type of localization is the most important because the destination, the path to follow, and the world information will be expressed in this reference system.

The most common technology used to obtain a position in global coordinates is the GNSS system. This one is an essential system in any vehicle and is often combined with Inertial Measurement Unit (IMU)s to provide both the position and orientation of the vehicle.

Another method to obtain global coordinates is estimating the vehicle's position and orientation with respect to a map. Map-based localization techniques use perception

algorithms to detect specific points or landmarks that appear on the map; in order to determine the location of the vehicle with respect to those points If the map is georeferenced, it is even possible to combine the localization estimated relative to the map with the coordinates obtained from the GNSS service.

Local localization The term "local localization" is associated with localization techniques that do not provide coordinates in a global reference frame. This is the case for odometry methods, for example. An odometry method estimates the movement of the vehicle by integrating the relative changes in the position or integrating the velocities. For this reason, the estimated position will always be relative to the origin of the movement, hence the term "local" in localization.

One of the possible odometry sources, and perhaps the most available in any vehicle, is the odometry from the wheels. All vehicles include encoders on the wheels to measure the velocity with an odometer and display it to the driver. A wheel odometry system can exploit this information, combined with the steering angle, to estimate the relative movement of the vehicle.

While odometry from the wheels is considered a proprioceptive method (it measures the inner vehicle's state), there are other methods considered exteroceptive because they use sensors to measure the changes on the outside of the vehicle. On the one hand, camera data can be used in visual odometry, algorithms that analyze the changes in consecutive images to estimate the movement. Likewise, the same approach can also be followed with LiDARs by processing consecutive point clouds to provide a source of local localization.

The main drawback of odometry methods is that they suffer from drift. The cause is that the vehicle pose is obtained by integrating small increments in the position. Thus, the error (even if small) accumulates over time, causing the drift.

1.3 Motivation

Accurate and reliable localization is essential for autonomous vehicles. Prior to any move, maneuver, or even route planning, a vehicle must first know its position. If the vehicle's pose is not accurate or is very noisy, proper control of the vehicle is unfeasible. Moreover, most obstacle detection systems have a final object tracking step, where the movement of the detected objects is predicted and their position estimated. In that tracking process, the movement of the ego vehicle is considered

to compensate for it. Thus, a wrong estimation of the ego localization can also cause wrongful estimations in the pose of the obstacles around the vehicle.

Nowadays, GNSS systems are the unquestionable go-to solutions to solve the localization problem in vehicles. A good, accurate GNSS system is, however, expensive. High-end systems include multiple antennas and support different satellite constellations (GPS, GALILEO, GLONASS, BEIDOU, or QZSS). Moreover, most of the high-end GNSS systems can receive differential corrections to reduce the estimation error and also support Real-Time Kinematic (RTK) positioning to reach centimeter accuracy.

Nevertheless, GNSS solutions are not flawless since they require good satellite coverage in order to obtain an accurate estimate. Plus, they cannot work indoors, so applications that include underground parking lots or tunnels would require a different solution because a GNSS will not be able to work. Another challenging scenario for GNSS systems is navigating inside cities; due to urban canyons. If the vehicle is traversing an area with tall buildings or narrow streets, the sky visibility is reduced. Moreover, the remaining visible satellites will likely be concentrated. Thus, the system will suffer from poor geometry for Geometric Dilution of Precision (GDOP) [42].

Finally, although the error from the GNSS position can be estimated, it cannot be controlled. Satellites are constantly moving, and vehicles might enter an area with poor coverage, so the localization error cannot be predicted. That makes autonomous vehicles unable to anticipate a situation with a poor localization estimate.

Alternatives to pure GNSS solutions include fusing it with odometry sources or using a different global localization approach based on digital maps. Integrating odometry measurements with GNSS is a common practice and helps mitigate some of its problems. However, these local methods suffer from drift and, therefore, are not a valid option for long times GNSS denial.

On the other hand, localization solutions based on digital maps rely on the availability and quality of the map. They use the data from cameras and LiDARs to detect landmarks and try to match the vehicle's surroundings with the map. If the sensors are accurate and the map has a high resolution and quality, the localization obtained by these methods can be superior to GNSS systems.

In addition, newer advances in sensor technologies and perception algorithms can be exploited to improve localization systems. Generally, perception and localization systems are independent, while many localization systems (either sensor-based odometry or map-based methods) could benefit from the processed data obtained

from perception algorithms (road detection, obstacles, or traffic signs, for example).

After all, What does having a good localization even mean? In the real world, it is not straightforward to determine if the localization output of an algorithm is the actual vehicle's position. The problem of a lacking ground-truth system for localization has been tackled before in indoor mobile robotics. In those applications, high accuracy Motion Capture (Mo-Cap) systems are installed in testing areas to provide a localization reference. Sadly, these systems do not work outside and have a limited range making them unsuitable for experiments with road vehicles. Previously, localization reference systems for road vehicles consisted of high accuracy GNSS systems; still, they have the drawbacks mentioned before.

Last, the advances in communication technologies allow vehicles to cooperate with other vehicles and with the city's infrastructure. This integration of technologies enables new cooperative algorithms for autonomous driving, including cooperative localization methods with potential benefits.

1.4 Objectives

As stated before, there is room for improvement in localization systems for autonomous vehicles. The work done in this thesis revolves around the idea of integrating the advances in perception algorithms; in order to improve localization systems in autonomous vehicles. On the one hand, the systems that allow autonomous driving are complex enough, so using the information already processed by perception systems helps reduce the overall system's complexity. On the other hand, this thesis also aims to provide alternatives to GNSS systems, with the purpose of diversifying the viable solutions to the global localization problem in intelligent vehicles. Finally, another objective of this work is to lay some basics for cooperative localization algorithms based on environment perception.

Subsequently, this work is done according to the following list of objectives:

- Exploit the advances in perception technologies applied to autonomous vehicles in the latest years; to improve their localization systems.
- Study the benefits of integrating advances in 3D scene understanding with LiDAR-based odometry algorithms.

- Provide novel methods and systems for localization validation. Design such systems to guarantee an upper-bounded localization error.
- Enhance localization algorithms by the means of a multimodal cooperative localization system for autonomous vehicles, combining information from vehicles and infrastructure.

1.5 Thesis Structure

This thesis is structured in six chapters and two appendices. The central chapters that present the main contributions of this work have their own results sections. Plus, they are opened with a brief introduction and closed with some concluding remarks specific to that chapter.

The remainder of this thesis is summarized below:

Chapter 2

The second chapter is a continuation of the thesis introduction, where state-of-the-art works are analyzed and discussed. The literature reviewed starts with a background of autonomous driving platforms and demonstrations. Then, a brief review of perception algorithms for object and infrastructure detection is presented. Afterward, a deep analysis of the different localization methods is performed, including data filtering and fusion techniques. Next, vehicle cooperation mechanisms and cooperative localization algorithms are reviewed. Finally, the most common evaluation systems are reviewed, including a deep analysis of the several metrics that can be used to measure the performance of localization systems.

Chapter 3

This chapter presents a study of the performance of LiDAR-based odometry algorithms after filtering the input point clouds using 3D semantic information. Results show the advantages and disadvantages o multiple filtering configurations, obtaining the importance of each semantic element in the odometry output.

Chapter 4

The content in this chapter is focused on localization validation and reference systems based on landmark detection from LiDAR sensors, and it is divided in two parts. First, an odometry reference-less validation method is presented. The second part details a landmark placement optimization algorithm that can be used to generate a reference trajectory with a guaranteed bounded error.

Chapter 5

This chapter is focused on cooperative localization for connected autonomous vehicles. An estimation framework to solve this problem is presented, and its performance is evaluated.

Chapter 6

Finally, the conclusions of this work are presented, and the possible future work lines are discussed.

Related Work

2

> *If I have seen further, it is by standing on the shoulders of Giants.*
>
> **— Issac Newton**

This chapter details the related work in the context of the objectives presented in Section 1.4. First, a historical background is given in Section 2.1, where a review of the most famous autonomous driving platforms is provided. This review includes the application of each platform and its sensor setup. Section 2.2 presents an analysis of different perception technologies used in autonomous vehicles. The studied technologies are those that can benefit localization systems, including common LiDAR-based and camera-based object detection techniques and landmark detection algorithms. Then, Sections 2.3 and 2.4 present a deep review of localization techniques and filtering and data fusion methods, respectively. The works reviewed in those two sections are not only centered on autonomous vehicles. Other works from the fields of mobile robotics and state estimation are included. Afterward, localization evaluation systems are presented in Section 2.5, including the metrics used to analyze the performance of localization systems. Finally, Section 2.6 presents a review of communication technologies for vehicles and different cooperative solutions for perception and localization problems.

2.1 Autonomous driving platforms

Autonomous driving platforms have gained considerable interest in the last two decades, and this interest is only increasing over the years. The first autonomous driving vehicles started appearing during the 20th century with works such as ALVINN from the Carnegie Melon University in 1988, [43, 44], Navlab project, from the CMU as well [45], or the Eureka-PROMETHEUS project from the European Union [46]. Nevertheless, the bigger jump happened in 2004, when Defense Advanced Research Projects Agency (DARPA) from the U.S. held a competition for autonomous ground vehicles, the DARPA Grand Challenge [47, 48]. This challenge consisted in following a 240 km route in autonomous mode through the desert from California

to Nevada. The off-road route included multiple difficulties, as it required driving through rough terrain with several obstacles. Sadly, none of the teams were able to finish the challenge, but the competition was repeated the year after, in the 2005 DARPA Grand Challenge with more successful results. On this occasion, 5 participants managed to finish the course. The winner of the competition was the team from Stanford University with their vehicle Stanley [49], presented in Figure 2.1.

Fig. 2.1: Stanley vehicle from Stanford University in DARPA Grand Challenge 2005. ©2005 Standford University.

Stanley was equipped with an array of LiDAR scanners, radars, and cameras on top of the roof. Those sensors were used to determine the drivable space by detecting the road limits and also to detect the possible obstacles on its path. Additionally, the localization of the vehicle was estimated using a GPS sensor, an IMU, and the odometry from the wheels. The data acquired from those sensors were then processed and fused by an Unscented Kalman Filter (UKF). In general, most participants also relied on GNSS systems as the main localization source [50, 51, 52, 53]. Nonetheless, some of them did include different mechanisms to handle GNSS signal outages, which could lead to a navigation error.

After the success of the DARPA challenge, a new type of challenge was launched in 2007: the DARPA Urban Challenge. In this challenge, the vehicles did not have to cope with the physical difficulties of the rough off-road terrain, and the length of the route was only 96 km. Notwithstanding, this challenge brought new complexities, such as moving in a cluttered urban area while sharing the road with other vehicles

and following all the traffic circulation rules. In addition, vehicles must be able to operate under any weather condition (e.g., rain or fog) and with a degraded GPS signal. Accordingly, all participants made their vehicles even more intelligent than before, with new sensors suited for better perception in urban environments and more complex decision-making algorithms. The winner of this challenge was the team from Carnegie Melon University with their vehicle BOSS [54], shown in Figure 2.2.

Fig. 2.2: BOSS vehicle from Carnegie Melon University in DARPA Urban Challenge 2007 [54]. ©2009 Springer.

Regarding the localization system, this vehicle did not rely only on a GNSS-based system. In this challenge, the teams had access to a map of the roads, and some teams also decided to exploit that extra information to help improve the accuracy of the GNSS sensor. Two example vehicles are BOSS and Junior [55] from the Standford team, which finished in second place, clearly showing that using the road map for localization purposes had an impact on the final ranking.

Following the DARPA challenges, the Grand Cooperative Driving Challenge (GCDC) was held in Europe in 2011 [56] and 2016 [57]. While the first edition was mainly focused on vehicle platoons and cooperative adaptive cruise control, the second edition included two different challenges: cooperative intersection passing and platoon merging. In this challenge, vehicles from different teams must cooperate

with each other while following the traffic rules. This event also showed the benefits of cooperative driving.

After the DARPA and GCDC challenges, there has been a boom in the autonomous driving world, bringing new technologies and creating new companies and many different vehicles used for testing and demonstration purposes. For instance, 125 years after the first journey on a motor vehicle by Bertha Benz in 1888, a prototype vehicle from Mercedes drove the same route autonomously without any intervention [58]. The route had a length of 103 km and was composed of multiple villages, rural areas, and major cities, all with shared traffic and following the traffic rules. Furthermore, *VisLab*, from Parma University, which already participated in both DARPA challenges, completed the *VisLab Intercontinental Autonomous Challenge (VIAC)* [59]. Three autonomous vans completed a 16,000km route from Parma to Shanghai, where the 2010 World Expo took place.

In the last decade, we have started to see an initial development of commercial platforms, moved by private initiatives that aim to create a Level 4 / Level 5 autonomous vehicle in the next years. These advances are leaded by Waymo, which acquired and automatized a fleet of Chrysler vehicles, which has already covered more than 32 million kilometers. Nowadays, some of these vehicles; which are based on Google's self-driving car [60], have been successfully deployed as "robotaxies" in some areas of San Francisco. Another big company that is leading the development of autonomous driving technologies is Tesla. Tesla vehicles started offering Level 2 autonomy features and recently have started releasing its "full self-driving" algorithm, including Level 3 features. Classical car manufacturers are one step behind but advancing steadily. For instance, BMW has been planning for a while to release a Level 3 autonomous copilot but recently announced that it will be postponed until 2025 [61]. On the other hand, Mercedes, which built the Bertha prototype in 2014 [58], has already released this year the first Level 3 vehicle approved for European public roads [62]. Volvo also announced its new autonomous driving system at the latest CES consumer electronics show in 2022 [63], although its release date is still unknown.

2.2 Perception technologies

The perception systems used in intelligent vehicles have evolved over the years as the sensing technologies have improved [29]. Monocular cameras are the most common perception sensor in intelligent vehicles; for multiple reasons: They resemble human

vision, and artificial vision algorithms are easy to understand and develop. Moreover, they are relatively cheap, easy to use, easy to install in the vehicle, and have a low power consumption. For this reason, there are numerous methods for object detection using monocular cameras [64, 65]. Lately, thanks to the rise in machine learning and deep learning techniques, new computer vision methods based on neural networks are being used in intelligent vehicles with a substantial performance boost [66]. Some examples of deep learning techniques (i.e. convolutional neural networks) are implemented in Detectron 2 framework [67], from Facebook Research; such as the FASTER R-CNN [68] or the MASK R-CNN [69].

The only drawback of this type of sensor is that detections are obtained in the image plane, which is 2D. Thus, the 3D position of the detected objects is not directly known, including the distance to obstacles. In order to deal with this problem, some works estimate the distance of each pixel [70, 71], creating a depth map from the monocular camera. However, the most common approach is to use a different sensor to sense the objects in 3D and obtain the depth of the obstacles detected from the images [72]. The sensors that fill this gap are stereo cameras and LiDAR sensors. While the former requires a method to compute the image disparity and obtain the depth of the image [73, 74], the latter directly provides the 3D coordinates of the obstacle points, while also being more accurate.

The advantage of stereo vision techniques is that combining the 2D obstacle detections from images with the computed depth is straightforward [75]. Moreover, the generated depth map is dense and has enough information to detect vehicles, other obstacles, or even the road limits [76, 77].

On the other hand, LiDAR sensors provide a somewhat sparser point cloud, so identifying objects is a more challenging task. However, while they lack information density, the accuracy of the measurements is very high, hence LiDAR-based detections tend to be very accurate. The information extracted from LiDAR point clouds can be combined with camera-based detections or can be used independently to obtain the 3D detections. Detecting objects is generally easier in images. Camera-LiDAR fusion techniques try to exploit this by using LiDAR to obtain the 3D position of the detected obstacles and improve the detections [78, 79, 80]. Yet, detection solutions based on just LiDAR are simple, do not require sensor synchronization or calibration and can also provide acceptable results [81, 82]. Finally, other works that use only the LiDAR point cloud take a different approach when processing the data. Instead of directly processing the 3D data, the information is encoded into a 2D image in a bird's eye view. Then, the same deep learning methods that are used to process camera images are trained to process the data from the LiDAR [83, 9].

In addition to object detection techniques, another method to understand the scene is through instance segmentation. This method consists in assigning a label to each pixel (in images) or point (in point clouds). Initial approaches used image processing and deep learning methods to obtain the semantic segmentation of the scene [84, 85]. Nevertheless, newer techniques are able to process 3D information and produce a similar output, assigning a label to each point [86, 87].

In addition to detecting potential obstacles, intelligent vehicles must also detect the road infrastructure to navigate safely. Road infrastructure includes the road surface (drivable area, lanes, line types, road topology) and other elements, such as traffic signs or traffic lights. Additionally, detecting other infrastructure elements can be helpful to localize the vehicle within a map. Some examples of such elements are lamp posts, trees, or other static objects like buildings or fire hydrants.

Detecting the road markings and identifying the drivable area and the road lanes is crucial for autonomous navigation, and it has been a perception task since the first vehicle prototypes. Most works are based on image processing methods and exploit the brightness difference between the lines and the rest of the road [88, 89]. In [90], map information and data from a stereo camera are used to detect road markings and curbs. Additionally, in [91] camera and LiDAR sensors are combined in a feature-level fusion method to detect the drivable area and the road markings. Lastly, it is also possible to detect the road lines with only a LiDAR sensor. In addition to the 3D coordinates of the points, LiDAR sensors also provide the returned intensity of the laser beam. That information can be exploited to differentiate the lane markings from the asphalt [92]. Other works use deep learning methods to learn the line features from LiDAR data [93].

Finally, the detection of traffic signs and traffic lights is also important, albeit not critical, for navigation. This is required to comply with the circulation rules and drive safely. Together, traffic signs and other elements such as lamp posts or trees can be considered landmarks and be used as key points in localization algorithms, hence the relevance of such perception methods in this thesis. The detection and recognition of traffic signs are always computed using a camera because identifying the type of sign is not possible without visual information. This last step is usually performed by machine learning methods, such as Support Vector Machine (SVM) algorithms or neural networks that learn the possible sign types and assign the corresponding class to the sign detection. In [94], a neural network is used to classify the traffic signs, and a Kalman Filter is applied afterward to provide consistency and memory in the detections. Other works, such as [95], perform a color-based segmentation to segment the signs instead of analyzing the shape of the objects. An example of a

SVM-based traffic sign classifier can be dound in [96], and similar methods are used to detect traffic lights [97]. Furthermore, landmark detection is generally applied in mapping and map-based localization processes. In [98], natural landmarks (i.e., trees) are detected from a 2D LiDAR for localization purposes. Urban environments are richer in landmarks than other places like highways or rural areas. Therefore, most of the works focused on landmark detection are oriented to this type of cluttered environment. For example authors in [99] propose a semi-automated extraction of light poles using a LiDAR sensor. In [100], an automatic detection and classification of pole-like structures method is presented. Moreover, a stereo camera is used in [101] to detect poles and trees based on the occlusions generated between the two cameras. Due to those occlusions, the disparity map shows a black area next to the poles on the disparity image, and the poles are then extracted by edge detection. Lastly, authors in [102] recently presented a landmark-based global localization algorithm with a custom point cloud processing algorithm to extract pole-like landmarks from LiDAR data. Then, a map of the landmarks is created to obtain a global reference within the map.

2.3 Self localization

Any system capable of autonomous or semi-autonomous navigation needs a localization system. Examples are small mobile robots like small intelligent vacuum cleaners, industrial AGVs, self-driving vehicles, or Umnanned Aerial Vehicles (UAVs). For this reason, localization is a common problem in multiple areas, and the techniques and solutions are usually similar.

The localization problem is a well-studied topic in robotics, which consists in finding the robot's pose (position and orientation). In order to be complete, this positioning problem must be formulated in 3D, although sometimes a simplification to 2D is enough when the environment is flat. The 3D position of the robot is represented in cartesian coordinates: $\{x, y, z\} \in \mathbb{R}$. Accordingly, the orientation consists of the three angles: $\{roll, pitch, yaw\} \in [0, 2\pi)$. Together, they form a pose with 6 Degrees of Freedom (DOF). The robot's orientation can also be represented by Euler angles [103], or by quaternions [104]. Furthermore, another common representation of a pose is by a 4×4 transformation matrix from the $\mathbf{SE}(3)$ group[1] [105].

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbf{SE}(3) \tag{2.1}$$

[1] $\mathbf{SE}(3)$ is the Special Euclidean Group in 3 dimensions that represents rigid motions.

Where $\mathbf{R} \in \mathbf{SO}(3)$[2] is a 3×3 rotation matrix that encodes the robot's orientation angles and $\mathbf{t} \in \mathbb{R}^3$ is the 3D translation vector that represents the robot's position.

Due to the nature of rotations, measuring the difference, or distance, between two rotations in a matrix form is not straightforward. In [106], six different metrics to evaluate the distance between 3D rotations are analyzed. In robotics, more particularly in the field of pose estimation, the use of the Lie groups $\mathbf{SE}(3)$ and $\mathbf{SE}(2)$ have gained interest in recent years. The reason is that these groups are nonlinear because of the rotations, and Lie theory provides a mechanism, the Lie algebra, to linearize the nonlinear operations performed in the Lie group [107]. This Lie algebra is a linear vector space, tangent to the group's manifold, which allows the linearization of such operations through exponential and logarithmic transformations.

Another important topic to consider when representing the robot's pose is uncertainty. Everything has uncertainty. Sensors, measurements, the robot kinematic and dynamic models, or extrapolation operations can introduce errors in the pose estimation process. Measuring and modeling this uncertainty has been one of the main study cases for decades. In order to take into account these errors, most works adopt probabilistic approaches [108]. This way, the localization problem is turned to finding the most likely pose of the robot, and the uncertainty can be estimated by determining how likely is that pose to be. Regarding error covariance estimation, in [109] a method is proposed to estimate the uncertainty of a visual odometry algorithm for UAVs. In addition, authors in [110] proposed a generic method to model the drift error of an odometry method and estimate the covariance of its error.

Localization is a complex problem with multiple possible solutions. A review of the numerous localization techniques is presented in [111]; applied to mobile robotics, albeit still accurate for other mobile vehicles. In [112], different localization methods applied to autonomous vehicles in highway environments are presented. Generally, the way to obtain the robot's pose is to either infer the pose by measuring external references or estimate how much the robot has moved by measuring its displacement or integrating acceleration/velocities. Furthermore, most localization methods rely on data fusion techniques to combine the information perceived from different sensors to improve accuracy and robustness.

As introduced in 1.2, localization methods can be local or global. Local methods are also called odometry or dead-reckoning. They are based on incremental techniques that compute the pose by integrating the movement. Global methods estimate

[2] $\mathbf{SO}(3)$ is the Special Orthogonal Group that represents rigid rotations in 3 dimensions.

the pose within a common coordinates system, usually using external references. The remainder of this section will present and analyze multiple local and global localization solutions, divided by the sensor and techniques used.

2.3.1 Wheel odometry

Wheel odometry methods work by measuring the velocity of the wheels through encoders. The velocity is then integrated, and the robot's pose is calculated through its kinematic model. In vehicles with steering wheels like cars, the kinematics are usually modeled with the Ackermann model [113]. In those cases, the steering angle must also be measured and introduced in the model responsible for generating the odometry.

Wheel odometry, or dead-reckoning, techniques date back to the first mobile robots due to encoders being simple and inexpensive sensors. In [114], a wheel odometry system for a differential-drive mobile robot is presented. That work also proposes a calibration mechanism to mitigate systematic errors in the odometry process. Wheel odometry is generally more complex for road vehicles than indoor mobile robots. First, the kinematic model is more complex. Also, indoor robots usually move in flat environments, while road vehicles are more likely to move on uneven terrain. Authors in [115] also presented a calibration method to deal with systematic errors, but this time applied to Ackermann vehicles. Moreover, authors in [116] even consider the effect that the dynamic load in the vehicle has on the wheels. That effect is modeled and considered in the wheel odometry algorithm, thus improving the system's accuracy.

One of the most common sources of error in wheel odometry systems is when the robot is turning, i.e., moving in a curve. For that reason, most works combine the odometry from the wheels with a IMU sensor. The IMU provides a more accurate estimation of the robot's orientation, which can help correct the error that wheel odometry systems have in curves. In [117], authors combine a gyroscope with a wheel odometry system, using the gyroscope data when the odometry drifts. Furthermore, authors in [118] propose a method to provide an accurate localization estimation based on wheel odometry and IMU; to work when LiDAR and cameras are unavailable. Their work exploits deep learning techniques to learn the vehicle model and use the learned model to generate the odometry from encoders and IMU data. Finally, another method combining wheel odometry and IMU is presented in [119]. Then, the generated wheel odometry is fused with a visual odometry system to improve its performance.

2.3.2 IMU odometry

IMU sensors are a combination of accelerometers, gyroscopes, and magnetometers that can measure linear acceleration, angular velocity, and orientation to the magnetic north, respectively. These sensors are generally greatly affected by external interferences and are rather noisy. Therefore, a thorough calibration process is always required. For this reason, their usage in localization systems is usually to complement another system instead of being the main sensor. In addition to the works presented in the previous subsection, IMU sensors are also often combined with visual odometry systems [120, 121]. However, the best synergy is probably obtained by combining a GNSS unit with an IMU [122, 123, 124]. GNSS systems can provide accurate positioning, but they are not able to obtain the orientation of the vehicle, which is provided by the inertial system. That is the explanation to why most commercial GNSS solutions include an IMU as well.

Although IMU systems are most often seen as complementary sensors, recent advances in machine learning techniques are enabling different works that generate acceptable odometry from IMU sensors. Authors in [125] proposed a learning framework to generate an odometry from low-cost IMU sensors. Finally, the work presented in [126] proposes a deep learning method to learn the error model of an IMU and improve the accuracy of the odometry system.

2.3.3 Visual odometry

Since cameras are one of the most popular sensors for intelligent vehicles and also mobile robots, the field of visual odometry has received significant interest in the last two decades.

Classical visual odometry techniques, or optical flow techniques, measure the camera movement by detecting specific features in consecutive images and performing a matching step between the extracted features. With that information, these methods are able to estimate the movement of the camera by analyzing the displacement of the features in the images. In [127], this approach is applied to an omnidirectional camera mounted on a car. Authors in [128] propose a method to handle motion blur in a visual odometry process. Direct methods, such as [129] and [130] extract information from the intensity gradients in the image. Therefore, these techniques work with the whole image rather than depending on keypoint detection and extraction methods. Other works, such as [131], perform Simulatneous Localization

and Mapping (SLAM) to keep track of the detected features over time, building a map of the environment.

Furthermore, depth information is of significant help in these methods. For this reason, some works suggest using stereo cameras or depth cameras to aid in the generation of visual odometry [132, 133].

In cases where only a monocular camera is available, it is possible to use deep learning techniques to estimate a depth map from images and exploit the generated depth information in the visual odometry step [134, 135].

Finally, visual odometry methods have also received significant benefits from the advances in machine learning and deep learning techniques. In [136], a deep learning method is proposed with a neural network that learns the image features and infers the camera poses from a sequence of images. Instead of using image features, authors in [137] propose another deep learning method that exploits the semantic information in images to generate the odometry.

2.3.4 LiDAR odometry

The principles of LiDAR odometry are very similar to those of visual odometry. The relative movement of the sensor is incrementally computed by analyzing the displacement of the points from consecutive measurements. The advantage of LiDAR sensors is that they measure the distance to objects directly with very high accuracy. However, multilayer LiDAR produce point clouds with numerous points, which in addition to the 3D coordinates, also encode other information such as the returned intensity. Processing this sheer amount of data requires a significant amount of computational power and very efficient methods.

In order to mitigate this processing burden, some works propose different strategies to reduce the scale of the data. In [138] authors propose an odometry method based on 2.5D data by projecting the 3D points into multiple 2D height maps. Other work suggested the use of a siamese network that was capable of learning dimension-reduced representations of the 3D point cloud [139]

The classical incremental odometry approach is performed by matching consecutive point clouds through a registration method that estimates the most likely transformation between the two scans. In [140], a Normal Distributions Transform (NDT) algorithm is selected to perform this matching process and generate theodometry from consecutive point clouds. Additionally, in [141] a different matching algorithm

based on Collar Line Segments is adopted to perform the registration between scans. This method is designed to efficiently handle the sparsity and quantity of points in LiDAR data. Also, following the trends, new approaches based on deep learning methods are being studied. In [142], a Convolutional Neural Network (CNN) is used to extract features, and the transformation between consecutive point clouds is estimated afterward. Other techniques to infer the movement between scans include Recurrent Neural Networks (RNNs), such as the Long Short-Term Memory (LSTM), as was suggested in [143].

As happens with other odometry techniques, these methods are prone to drift due to their incremental nature. Consequently, many LiDAR odometry approaches actually implement a complete SLAM method. Building a map of the environment while estimating the odometry provides the system with memory, and the map can be used to correct the drift in the generated trajectory. For the last decade, LOAM [144] has been, and still is, the most popular LiDAR-based SLAM method. The idea behind LOAM is to separate the odometry and mapping into two parallel processes. First, a low-accuracy preliminary odometry estimation is performed with high frequency. Concurrently, a low-frequency mapping process the point clouds with the estimated odometry and performs a fine matching and registration of the scans. The 3D data processing performs a feature extraction step, where feature points from planar surfaces and sharp edges are obtained. Afterward, those features are used to compute point-to-plane and point-to-lane distances and estimate the transformation between the two point clouds. Later, a lightwave version of the LOAM algorithm was presented in [145]; a method named LeGO-LOAM. This version follows the same methodology but was adapted to run on embedded computers like the Nvidia Jetson machines. In LeGO-LOAM, the ground plane is segmented and exploited to estimate the pitch and roll angles. Similar to LOAM, planar and edge features are extracted to estimate the movement of the sensor. However, a 3D segmentation is performed beforehand to filter the point cloud and reduce the number of points, thus speeding up the process. Another LiDAR-based SLAM algorithm is presented in [146], that applies a scan-to-model matching method based on the Implicit Moving Least Squares (IMLS) surface representation.

The recent advances in 3D semantic segmentation techniques have been incorporated into some works to improve the performance of the LiDAR odometry and SLAM methods. In [147], a point cloud alignment based on Iterative Closest Point (ICP) algorithm is presented. Before matching the points, 3D semantic segmentation is performed, restricting the point association to those objects of the same type. A semantic SLAM method is presented in [148], where a CNN is used to extract the semantic information. Then, that semantic information is used to remove moving

objects from the scene and apply semantic constraints in the scan matching process. Other work that exploits 3D semantic segmentation to generate LiDAR-based odometry can be found in [149]; this time applied to UAVs in forest environments.

In order to improve accuracy and robustness, LiDAR sensors are also often combined with other systems. Authors from LOAM presented a second version of the famous algorithm that also included visual information in the odometry generation [150]. In [151], a powerful SLAM framework combining LiDAR and a monocular camera is presented, where the LiDAR information is used to find the depth of visual features. LiDARs are also typically complemented by IMU sensors to improve the reliability of the localization solutions. A LiDAR odometry algorithm is presented in [152], where an IMU is used to constrain the estimated rotations and improve the performance of the odometry system. Finally, IMU sensors are also integrated in LiDAR-based SLAM methods to preprocess the scans [153] and to compensate motion when aligning two point clouds [154].

2.3.5 GNSS-based localization solutions

GNSS are the most used localization systems in autonomous vehicles. They are globally available and flexible. While very high-end systems can provide centimetric precision, it is also possible to obtain a position estimate (with some meters of error) with low-cost sensors.

Since GNSS systems directly provide the position of the receiver, they usually do not require any extra computation. However, autonomous systems that rely on these systems need to estimate the quality of the GNSS estimation. A map-aided integrity monitoring system is presented in [155]. Additionally, other GNSS integrity monitoring systems are reviewed in [156].

In order to improve the accuracy and robustness of GNSS in environments with low signal coverage, GNSS receivers with antenna arrays are proposed in [157]; for applications in urban scenarios.

The precision of GNSS systems can be improved by receiving corrections from other receivers, with Differential GNSS or RTK systems. Although both methods require a base station, Differential GNSS [158] has a higher availability, a lower cost, and allows a greater distance between the base station and the vehicle receiver. RTK systems have a more limited range but a much higher precision. In [159] a low-cost RTK-GNSS system is used for UAV applications. Authors in [160] propose a calibration method for GLONASS GNSS receivers with RTK corrections. Furthermore,

in [161] different positioning strategies with multiple GNSS constellations and RTK correction are reviewed.

Finally, it should be noted that GNSS systems are usually combined with other information sources, to improve their accuracy or to provide information about the vehicle's orientation. Some of the most common data fusion approaches used in autonomous vehicles will be presented in Section 2.4, including data fusion systems based on GNSS.

2.3.6 Landmark-based localization solutions

Typically, a localization solution based on landmarks requires previous knowledge about the landmarks' positions. Therefore, a map of the environment containing information about landmarks is usually precomputed, either automatically or manually. Then, the robot must be able to detect the landmarks and obtain the relative position of each detected landmark with respect to the robot. Since the global position of a landmark is known (relative to the map), every measurement contains information about the global robot pose within the map. Combining all measurements produces a system of equations, which are nonlinear due to the nature of rotation angles. Such equation systems are treated as Non-Linear Least Squares (NLLS) problems, which solution can be estimated by optimization algorithms such as the Levenberg-Marquardt method [162].

The use of external landmarks (both natural and artificial) for localization purposes has been widely studied in the worlds of mobile robotics and intelligent transportation systems. The first formulation of a landmark-based localization solution dates back to 1988 [163], where a camera is used on a mobile robot to detect vertical edges and match them with the vertical poles given in a map. Indoor mobile robots have benefited from this global localization approach because GNSS is not available inside buildings. In [164], a camera mounted on a mobile robot is used to detect landmarks based on their appearance and localize itself on a map. Another landmark-based localization approach based on visual information is presented in [165]; applied to robots participating in a RoboCup[3] event.

In some environments, it is possible to install human-made landmarks specifically designed to be used in a localization system. In [166], a laser sensor is used to measure the bearing angle to artificial landmarks. The final robot position is estimated by a triangulation method. Authors in [167] specifically designed unique

[3]RoboCup is a well-known competition where small mobile robots play soccer.

landmarks and placed them on the ceiling to be easily detected. Multiple other works prefer the use of QR codes as unique landmarks [168, 169, 170].

The landmark-based localization approach is sensor-agnostic and does not even need a sensor at all. Communication devices can also be exploited to determine the distance between the receiver mounted on the robot and other communication nodes based on the transmission time. For example, the signal strength received at a WiFi interface can be used to estimate the distance to access points and utilize them as landmark measurements [171, 172]. A similar approach is taken when using Ultra Wide Band (UWB) radio technology for wireless positioning [173, 174].

Landmarks can not only be used to localize the robot inside a map but also to update or generate the global map. In [175], authors propose a visual-based localization based on landmark detections; that is also capable of updating the global map. If the map is not available, the localization and the map can be estimated jointly with SLAM techniques based on landmarks [176].

A famous landmark-based SLAM method is GraphSLAM [177, 178], which builds a topological map based on landmark detections and odometry. An example application of this method applied to autonomous vehicles is shown in [179], where landmarks are detected using a radar sensor.

Intelligent vehicles navigate in outdoor environments, so the approaches previously presented for mobile robots must be adapted to work in less structured and changing environments, such as cities, rural areas, or highways. In [180], a camera is used to extract visual features from the driving environment and match them with a previously computed map. An IMU is integrated to estimate the movement of the vehicle and complete the localization method. Some recurrent elements in these environments are vertical pole-like objects such as lamp posts, trees, traffic lights, or traffic signs. These elements can be easily detected and identified, becoming potential landmarks. In [181] a stereo camera is used to reliably detect pole structures. Then a landmark-based localization method is applied to find the vehicle's position within a map. Other works use semantic information to identify street lamps, traffic signs, and trees [182] for localization purposes. Finally, a tailored landmark representation was recently proposed in [183] to help in the automatic mapping of driving environments, using traffic signs and traffic lights as landmarks.

Notwithstanding, road vehicles can also use a different type of landmark: road markings. Road lines and other markings (e.g., pedestrian crossings) are available on most roads and can also be used as landmarks for localization purposes. In [184], authors combine HD maps, a stereo camera, and a low-cost GNSS module to obtain

an accurate localization using road markings as landmarks. Furthermore, semantic information is exploited in [185] to solve the data association problem between landmark measurements and map elements. In addition to road markings, building facades and pole-like structures are also considered in [186], increasing the amount of information that can be matched with the global map. Because road markings can produce very similar measurements at different points in the map, some of the presented works rely on GNSS measurements to estimate the map area where the vehicle is located, and, later on, the fine pose estimation is obtained from the landmark measurements. In [187], however, the initial estimation of the map area is obtained by a topological localization step that recognizes the scene and estimates the map area where the vehicle is located.

2.4 Data filtering and fusion

As we introduced before, one of the most important tasks of localization systems is dealing with uncertainty. Modeling and estimating the error noise in the system is paramount for a reliable system. In localization systems, there are two types of models to consider. On the one hand is the kinematic model that characterizes the robot movement, or the motion model. On the other hand are the sensor models, which are used to estimate the nature and the magnitude of the measurement noise. While the motion models depend solely on the robot or vehicle, sensor, or measurement, models depend on the sensor type, the detection algorithm applied, and the elements that are being detected. Multiple probabilistic approaches to motion and sensor models can be found in [108].

In the literature, the noise in both the motion and measurement models is usually assumed to be Gaussian (i.e., following a normal distribution). Even when the error noise follows a different distribution, Gaussian approximations have proven to be very effective [188]. Nevertheless, it is also possible to model error noise with different distributions [189, 190]. As an example, a probabilistic model is used in [191] to estimate the error noise when detecting road markings from cameras, with its posterior integration in a landmark-based localization system.

The most common way to represent data uncertainty is through data variance and covariance matrices, which represent the magnitude of the noise and the correlation between variables. Filters are methods that receive noisy data and reduce/mitigate the noise. In order to perform the filtering, it is important to know the covariance

of the data to be filtered. Then, the noisy data is fitted to a model, considering its uncertainty, and the filter outputs the corrected data.

The most famous and most used filter in robotics and signal processing is the Kalman filter [192]. The Kalman filter is designed to work with linear systems under Gaussian noise. It has applications in noise filtering, object tracking, and data fusion. The process consists of two parts: First, a prediction step is performed, where the state is updated based on the motion model and the control inputs. Afterward, an update step is executed, where the sensor measurements are used to correct the predicted estimation and generate the filtered output.

Several variations of the Kalman filter have been introduced over the last decades to address its limitations and improve its performance in more complex systems. One of the first major variations is the Extended Kalman Filter (EKF) [193], which introduces a linearization method to deal with nonlinear systems. However, this approach still requires the motion and observation models to be based on differentiable functions, and the noise must be Gaussian. Another determinant variation of the Kalman filter is the UKF [194], which can deal with systems with high nonlinearities. Moreover, the UKF does not require differentiable model functions. Instead of computing the function Jacobians, this method is based on a deterministic sampling technique, the unscented transformation, to obtain several sample points called sigma points around its mean. This filter can also be implemented in a square root form [195], which guarantees the semi-definiteness of the final state covariances.

Instead of following a probabilistic approach, it is also possible to follow a statistical approach with information filters. Information filters are based on the Fisher information theory and can be considered the dual of Kalman filters. These filters represent Gaussian data in its canonical form by an information vector and an information matrix, which is the inverse of the covariance matrix.

Nonetheless, even when nonlinearities in systems can be dealt with, these types of filters still struggle when the a posteriori error noise is not Gaussian. One alternative to Kalman filters regarding this aspect is the H-Infinity (H_∞) filter [196]. The H_∞ filter does not try to reduce the data variance, and therefore it does not require knowing the statistical distribution of the data beforehand. Instead, what it does is assume the worst-case disturbances and minimize the maximum expected error. That provides a robust estimator that compensates for inaccuracies in the system models.

Another alternative consists in using random samples from the posterior distribution; to approximate the state space. That is the approach followed by Sequential Monte

Carlo sampling methods [197]. This method, also known as the particle filter, represents the a posteriori distribution by a set of weighted particles [198]. This set of particles leads to a nonparametric representation of the posterior, which can represent more distributions than just the Gaussian. Each particle is associated with an importance factor, which can be seen as a weight. The importance factor assigned to each particle is computed according to the probability of having that sample with the given measurements. Finally, a resampling process is performed to keep the particles with a higher importance factor and discard the less relevant ones.

All these estimation techniques are not only used to filter noisy data but also to combine the information received by multiple sources. Multisensor fusion techniques are paramount for intelligent vehicles, mainly for perception and localization systems. In the past decades, data fusion approaches have been profoundly studied and adopted in multiple engineering areas but particularly in robotics and intelligent transportation systems [199]. Some examples of multisensor fusion methods have already been introduced in the previous sections of this thesis. Additionally, several works provide a review of the multiple techniques and applications to autonomous vehicles [200, 201], including deep learning techniques applied to sensor fusion [202].

The field of localization has received special interest in these techniques for multiple reasons previously mentioned. In [203], an EKF is implemented to provide the state estimation for an aerial vehicle based on sensor fusion. An adapted EKF is also used in [204] to combine a GNSS and an IMU for localization in road vehicles. Furthermore, authors in [205] apply an UKF to combine GNSS, IMU and digital maps. The UKF is also used in [206], to fuse Ultra Wide Band (UWB) and IMU readings from a UAV. Both, EKF and UKFare compared in [207] when implemented in road vehicles.

The H_∞ filter has received less attention in the state estimation field than the more tested Kalman filters. However, it has been shown that it can outperform its Kalman counterpart when applied in autonomous vehicles [10]. In [208], the H_∞ filter is combined with an EKF to estimate the state of a soft robot.

On the other hand, particle filters are widely used in state estimation for intelligent vehicles, especially with map-based approaches. An adaptive Monte Carlo localization method was initially presented in [209]. In [210] a similar method is presented, with theextra integration of a GNSS sensor. This method combines LiDAR and GNSS sensors with a digital map to obtain a robust and accurate localization estimation in autonomous vehicles. A particle filter is also applied in [211] to estimate the vehicle position based on road markings detections from multiple cameras. Additionally,

authors in [212] proposed a similar method to fuse GNSS, LiDAR, IMU and wheel odometry for autonomous driving in urban areas. Finally, a particle filter was also applied in [213] to estimate the position of an aerial vehicle based on wireless signals.

2.5 Localization evaluation systems and metrics

In order to determine how good a localization method is, we must evaluate how close are the estimated poses to the actual pose of the vehicle. In addition, it is also interesting to evaluate other characteristics, such as the processing time, the robustness and reliability under different scenarios, or its availability. For example, an odometry system based on infrastructure will not be available if the required infrastructure elements are not present in the environment. Also, a localization system could be very accurate, but if the processing time is too high, it would not be suitable for operating in real-time on an actual vehicle. Nonetheless, the metric that receives the most attention is always the accuracy of the system. Calculating the difference between the estimated trajectory and the actual vehicle poses is not straightforward. In order to do that, knowing the exact ground truth of the vehicle poses is required, which is a very complex task in real-world scenarios. While simulation environments do provide the actual localization ground truth, using them makes the evaluation results depend on the quality of the simulation models. In real-world scenarios, however, it is not possible to determine the exact pose of the vehicle because any system will have a small amount of error. However, the go-to approach is to use a high-accuracy system to generate a reference trajectory; that is used later on to evaluate other localization systems with lower accuracy. If the system under test has an accuracy closer to the reference system, then it is not possible to determine if the obtained error comes from the system under test or the reference system. In that case, such a reference system is not suitable to evaluate the localization system under test.

In applications that take place indoors, the most common reference systems are Mo-Cap systems [214]. These systems require installing small markers on the robot, providing a 6DOF reference system with millimetric and even sub-millimetric accuracy, which is enough for most localization systems because the sensors that are equipped usually on robots cannot reach those accuracy levels. The use of Mo-Cap systems has been widely used in the field of UAVs for testing localization and control systems. However, these systems rely on infrared cameras that do not work outdoors, so they are not suitable for evaluating localization systems in road vehicles. In the

area of intelligent transportation systems, outdoor localization systems were typically evaluated using high-end RTK-GNSS systems with high accuracy [215]. Sometimes the generated trajectory is post-processed offline to improve its accuracy. These approaches can reach centimeter-level accuracy, which is far from Mo-Cap systems but still acceptable. Lately, localization systems for autonomous vehicles have greatly improved, and GNSS-based reference solutions are becoming obsolete. Furthermore, GNSS sensors perform poorly in urban areas and environments with reduced satellite coverage, which limits the availability of such evaluation systems. An alternative approach is to exploit SLAM systems, where the poses and the generated map are optimized offline. In some cases, these methods are supervised by a human who can intervene by manually adjusting the measurement matching. In [216], a Monte Carlo method is used as the localization reference, combined with manual supervision to correct the matching between the detected landmarks and a digital map.

Before describing how trajectories are compared, it is important to understand how to compare two different poses. Let us assume we have two poses $P_{ref,t}, P_{est,t} \in \mathbf{SE}(3)$, where $P_{ref,t}$ is the reference pose and $P_{est,t}$ is the estimated pose from the system under test at the same timestamp t. The error in the estimation can be seen as the difference between the two poses, i.e., the transformation from P_{est} to P_{ref}:

$$E_t = P_{ref,t} \ominus P_{est,t} \tag{2.2}$$

Where $\oplus$ is the pose composition operator, and $\ominus$ is defined as its inverse [217]. Afterward, the translation and rotation components of E_t can be combined or evaluated separately, as suggested in [218] and [219]. The translation component of the measurement error is computed as the Euclidean norm of the translation vector. Regarding the rotation component, different rotation metrics in $\mathbf{SO}(3)$ can be found in [106].

Besides the definitions of the relation between poses, there are also different ways to analyze and evaluate the whole trajectory. In order to compare the trajectories, both should be synchronized in time, so each estimated pose is associated with a reference pose. If a common reference system is not available, it is also possible to align both trajectories using the Umeyama method [220]. Then, a first approach could be computing the translation and rotation error for each pair of poses, extracting several statistics on that data such as the mean, median, maximum error, or the RMSE. This technique is known as the Absolute Pose Error (APE) because it measures the total error in the whole trajectory at each timestamp in an end-to-end manner

[221]. However, while this approach is convenient for evaluating global localization methods, it has a significant flaw when trying to evaluate the performance of local methods. As it is mentioned in [218], the APE evaluation metric is not appropriate for odometry evaluation for the following reason: If the system has a small drift error at the beginning of the trajectory, the whole trajectory will be globally wrong, even if the movement is estimated correctly. However, if that same drift error happened at the end of the trajectory, the total APE will be much smaller, as only the latest poses will be affected by that error. Figure 2.3 shows this problem in a simple 1D scenario. There, it can be seen how the error obtained with the APE metric is much higher if the small estimation error happened at the start of the trajectory. After that small error, the estimated trajectory is still locally good; hence the APE is not suitable for evaluating odometry methods, which are meant to be local.

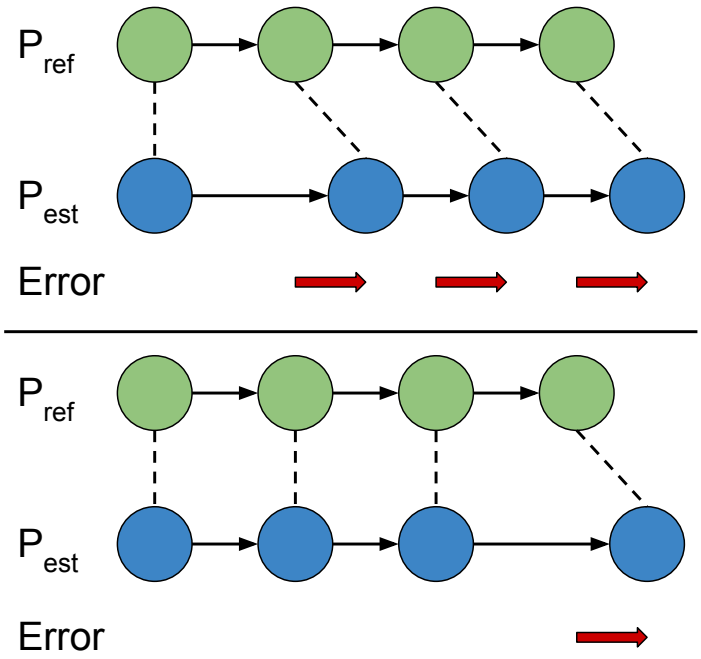


Fig. 2.3: Problematic of the APE metric. The green and blue circles represent the reference and estimated poses, respectively. The error of each pair of poses is drawn as a red arrow.

In order to solve this issue, a new metric is proposed in [218]: the Relative Pose Error (RPE). Instead of evaluating each pair of corresponding poses, this metric compares the relative displacement between poses, which is computed between two

poses of the same trajectory. Then, the RPE is obtained from the difference of deltas between reference and estimated trajectories:

$$\begin{aligned} \delta_{i,j} &= P_j \ominus P_i \quad i < j \\ E_i &= \delta_{ref,i,j} \ominus \delta_{est,i,j} \end{aligned} \tag{2.3}$$

Both metrics, APE and RPE are deeply studied in [222] for evaluating trajectories. Note that each metric provides different insights about the estimated trajectory. To conclude, the APE metric is preferable to analyze how good is the performance globally, which makes it preferable to evaluate global localization methods. On the other hand, the RPE metric is more suitable for the evaluation of odometry methods because it focuses the analysis on local segments of the trajectory.

Nowadays, multiple public datasets are available online to test, validate, and evaluate technologies related to autonomous driving. Due to the rise of deep learning techniques in this area and the gigantic amount of data required to train the models, There have been great efforts in the last decade to generate and label data. While most of these efforts have been generally focused on perception technologies, many of them include basic positioning data. Even some of them are specifically designed to evaluate localization methods.

Some datasets include a variety of sensors, different scenes, and multiple kilometers driven [223, 224]. However, the localization data is obtained directly from a simple GNSS unit, which cannot be used as ground truth for evaluation purposes. Recently, Waymo released its own dataset [225], but it is also centered around perception tasks. While the vehicle poses are provided, the paper does not mention how they were obtained, so their accuracy is doubtful. When datasets are more focused on providing a localization reference, RTK-GNSS systems are used, as in [226].

One of the principal datasets focused on localization is the Oxford RobotCar dataset [227, 228], which also includes an online challenge to evaluate localization results. This set of sequences includes data recorded from multiple sensors at different seasons under diverse weather conditions. The localization ground truth is obtained by post-processing GNSS with a RTK base station and an IMU. Authors claim to have a centimeter-accurate reference system.

Finally, the most famous dataset for autonomous driving technologies is, without doubt, the KITTI dataset [219], which also includes multiple online challenges on different topics such as object detection, semantic segmentation, or odometry. The

localization reference for the odometry challenge is obtained from a GNSS unit with RTK corrections, combined with an IMU. In order to evaluate the submitted trajectories, the authors implemented an RPE metric based on the one proposed in [218]. On the one hand, translation and rotation are evaluated separately. On the other hand, the RPE metric is designed to evaluate errors depending on the trajectory length and velocity.

2.6 Cooperative systems

This section provides a review of the different communication technologies that allow vehicles to communicate with other road agents and some of the multiple cooperative algorithms for perception and localization problems.

2.6.1 Communication technologies

Human drivers consider the actions of other road users and, in a way, cooperate with them when sharing the roads. Some cooperative actions are based on traffic rules, such as turn signals or traffic lights. Others are based on non-written rules, like a pedestrian visually checking the driver's attention before crossing the street. Intelligent vehicles are destined to share the roads with other users like human-driven vehicles and VRUs. Thus, it is required to grant intelligent vehicles with the capabilities to communicate with other road agents such as other vehicles, road infrastructure, or pedestrians [229]. The trends in Internet of Things (IoT) technologies and smart cities are pushing the development of new communication technologies, such as 5G, which will also bring multiple benefits to autonomous vehicles [230, 231].

Generally, the term associated with communication technologies involving connected vehicles is Vehicle-to-Everything (V2X) [232]. These communication technologies, however, can be divided into three groups: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Pedestrian (V2P).

- **V2V**: The form of cooperation that has received the most attention is based on vehicle-to-vehicle communications. One way to enable V2V communications is through Vehicular Ad-hoc Networks (VANETs) [233, 234]. A different approach consists in using specific communication infrastructure or mobile

networks to connect the vehicles [235]. The benefit of mobile networks is their high availability worldwide and how simple it is to connect to them.

The range of V2V applications is broad. This type of communication allows vehicles to share their position and their perception of the environment. For example, this information can allow vehicles to be aware of road threats in advance. Furthermore, vehicle cooperation schemes can provide benefits to localization and mapping systems.

- **V2I**: Communication with infrastructure is essential in smart cities and can be considered one of its cores. Vehicles can benefit from V2I communication by receiving information about traffic lights or traffic congestion. Moreover, traffic management solutions based on V2I and I2V communication can achieve high-level traffic coordination in urban areas [236].

  In order to enable this type of communication, mobile networks, such as 4G/5G, are a good option due to their high availability in urban areas. Other options include local networks based on WiFi or WiMax technologies [237].

- **V2P**: VRUs, such as pedestrians, are the group most involved in traffic accidents with injuries or deaths. For this reason, multiple works have been focused on exploiting communication technologies and connected vehicles to raise awareness among pedestrians of dangerous situations [238]. This type of communication is usually performed over WiFi networks [239], although it can also be carried out over mobile networks [240], and generally relies on connected devices such as smartphones to provide the pedestrians with alerts. Furthermore, these types of V2P applications allow warning pedestrians about possible collisions, even when there is no clear line of sight between the vehicle and the pedestrian.

2.6.2 Cooperative algorithms

Communication technologies and the V2X schemes described previously enable a plethora of cooperative algorithms that can improve all systems in an autonomous vehicle, including perception, localization, and mapping.

Regarding perception systems, cooperation between vehicles allows sharing the obstacles information, even if they are not in the line of sight [241, 242]. Furthermore, in [243] a cooperative perception approach is proposed, based on extending the sensing area by combining and aligning the 3D point clouds acquired by the LiDAR

sensor of each vehicle. One of the drawbacks of sharing sensor data is that it is usually very inefficient because it requires a large bandwidth. In [244], authors analyze the multiple parameters in WiFi-based V2V communication to effectively implement a cooperative perception system. Moreover, an approach to determine what information should be shared and when is presented in [245], allowing more efficient use of the communication channels. Authors in [246] analyzed the impact of such cooperative perception approaches in the final decision-making and planning of autonomous vehicles.

Cooperative localization approaches rely on sensor measurements that provide the relation between the poses of the involved vehicles. Vehicles share their initial localization estimate (generally from GNSS) and the detections from their sensor measurements. The distance between vehicles can be obtained from multiple sources. In [247], GPS pseudo-range measurements are used to estimate the distance between vehicles, resulting in a cooperative method that does not require additional sensors. However, cooperative solutions rely on wireless communication technologies, and the Received Signal Strength Indicator (RSSI) can also be exploited to estimate the inter-vehicle distance. This is a much more common choice due to its high availability [248]. Finally, onboard sensors are another option to detect the other vehicles [249]. Onboard sensors can also be used to detect elements external to the vehicles and use them as common reference points for cooperative localization [250].

In order to solve the localization problem, Gaussian filters are studied in [251]. A similar approach is adopted in [252], where an adaptive EKF is presented for cooperative localization in underwater vehicles. Another variant of the Kalman filter is presented in [253], combining GNSS and dedicated short-range communication for ITS applications. Additionally, authors in [254] proposed an EKF to combine odometry and inter-vehicle distances to cooperatively improve the localization when GNSS is not available in urban scenarios. Generally, the cooperative localization problem can be seen as a graph, where least-squares solutions can outperform Gaussian filter approaches [255].

Particle filters are also very common estimators to solve the cooperative localization problem, which is often combined with cooperative mapping [256, 257]. In [258], a Particle filter is used to combine GNSS and ranging data from terrestrial receivers and satellites. Furthermore, the use of a similar approach is studied in [259] to solve the localization problem inside tunnels, where the GNSS signal is not available. In order to select which measurements are better, a link selection method is presented in [260], using the vehicles with the best GNSS signal as virtual anchors. This approach

is later exploited by the same authors in [261], where the RSSI measurements from UWB communication receiver is combined with a prior GNSS estimation. The approach based on virtual anchors is proven to reduce the propagation of the initial bias.

2.7 Concluding remarks

In this chapter, the main technologies related to autonomous driving have been introduced, from the early developments to the current state-of-the-art techniques. The great advances that we have seen in the latest years show that these technologies can aid in making the roads a safer place.

Furthermore, this chapter has presented a study on the state of the art in perception and localization systems for intelligent vehicles, as well as an analysis of data fusion techniques and cooperative systems. The multiple state-of-the-art techniques described in this chapter are intended to serve as a broad context for the different works that will be presented in the next chapters of this thesis.

3 LiDAR Odometry Based on 3D Semantic Information

> *If we knew what it was we were doing, it would not be called research, would it?*
>
> **— Albert Einstein**

This chapter presents a study about LiDAR SLAM algorithms and how changes in the input point clouds do affect their performance. Semantic information is exploited to segment the original point cloud, and the generated odometry is evaluated with data from a public database. This work has been published in [1].

3.1 Introduction

LiDAR-based localization systems are very common in autonomous vehicles because of their compelling set of features, including outstanding measurement accuracy, a broad field of view, which usually spans $360°$, and the ability to operate under some challenging weather conditions. In addition, modern LiDAR sensors achieve higher resolutions, which was one of their weaknesses in the past, allowing them to provide more details from the detected objects, even at far distances. This set of features makes this type of sensor an attractive option for environment perception and localization in autonomous vehicles.

Modern LiDAR sensors have a long detection range that can easily reach up to 100m. Additionally, most devices cover a field of view of $360°$. This, combined with a high point density, allows extracting multiple features from the vehicle surroundings that can be used to register multiple point clouds and compute the vehicle odometry. These characteristics are also easily exploited by SLAM algorithms, which have been proven to obtain excellent results with LiDAR sensors in autonomous vehicles.

This chapter includes content from [1] and [2].

However, not all the environment features are equally beneficial to the scan registration process in the SLAM algorithm. Traffic scenarios are very dynamic and inconsistent. On the one hand, the vehicle's surroundings can vary depending on the driving area. Elements such as buildings, vegetation, or the road's structure are different in urban areas, rural areas, or highways. This lack of predictability makes it difficult to rely on specific types of features. On the other hand, traffic environments have multiple dynamic objects, such as other vehicles, bikes, or pedestrians. Moving objects may introduce spurious correspondences into the registration process.

Consequently, the work presented in this chapter aims to introduce a systematic analysis of the input sensitivity of LiDAR-based SLAM algorithms. Before feeding the sensor data to the SLAM algorithm, the point clouds are filtered based on semantic information. Some examples of this pre-filtering are presented in Figure 3.1, where each point is colored based on its semantic class.

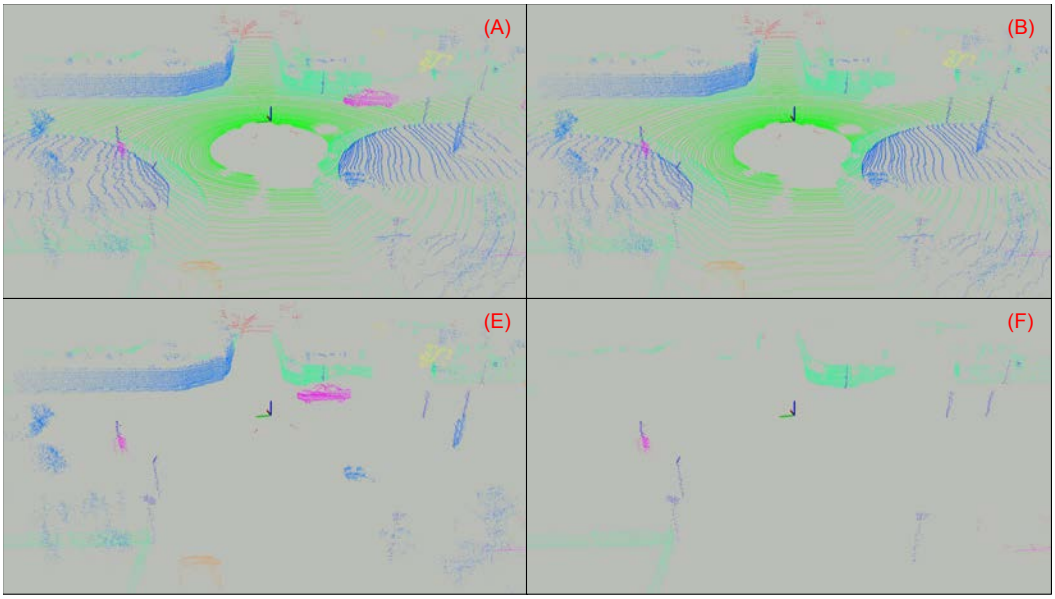


Fig. 3.1: Samples of filtered point clouds from different test configurations. Letters (A, B, E, F) refer to the filtering configuration used, described in Section 3.2.2.

In order to decouple the negative effect of the semantic segmentation accuracy, the semantic labels are extracted from the SemanticKITTI dataset [262]. This dataset is based on the odometry sequences from the original KITTI dataset [219], extending it with point-wise 3D semantic annotations.

The insights obtained from this analysis should provide a deeper understanding of how each element in the driving environment affects the registration of LiDAR point clouds in the odometry process. We believe this study will be useful in the future design of new LiDAR-based odometry algorithms.

3.1.1 Odometry algorithm

While there are many different LiDAR-based odometry methods, we chose to evaluate LOAM algorithm [144], which is a state-of-the-art approach based only on LiDAR data. This method has been selected for three main reasons. Firstly, it is a solid and well-known method that has been continuously used and cited since it was published in 2014. Secondly, it has stayed on top of the KITTI odometry leaderboard since it was published for both translation and rotation metrics; until the end of 2021. Finally, the algorithm has an open-source implementation, which is available in [263].

LOAM separates the SLAM problem into a high-frequency odometry estimation and low-frequency map registration steps. In order to estimate the vehicle movement, this method extracts edge and planar features from the point clouds; based on the smoothness of the points. Then, these features are processed independently, and a registration process is applied to find the relative transformation between the current sensor measurement and the computed map. This approach is followed by most SLAM methods; thus, the collected results from our proposed analysis can be extrapolated to other LiDAR-based algorithms.

3.2 Methodology

This section provides a detailed description of the evaluation procedure. First, the dataset selection, and the structure of the input data, are described. Then, multiple filtering configurations based on semantic information are presented. Finally, the evaluation metrics considered in the study are detailed.

3.2.1 Data

The data source selected for this study is the KITTI dataset [219]. Particularly, the odometry benchmark is the most appropriate for this type of evaluation, as it provides multiple challenging sequences in different driving environments with an accurate localization reference that can be used as ground truth. These sequences provide LiDAR input date from a Velodyne HDL-64E sensor, which has 64 vertical layers and a range up to 120m, producing more than 2.2 million points per second. Additionally, the SemanticKITTI dataset [262], which is built upon the KITTI's odometry sequences, is used in this study. While the main purpose of this dataset

is to provide a benchmark for 3D semantic segmentation, we are mainly interested in using its data to extract the 3D semantic information that has been manually annotated. Such annotations span different categories, which are presented in Figure 3.2. The labels include different vehicle categories, and other classes such as ground, vegetation, or structures, for example. In addition, moving objects are also labeled as dynamic, allowing us to consider all dynamic objects in the scene.

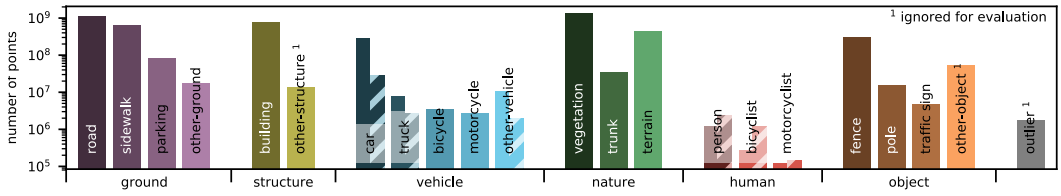


Fig. 3.2: Labelled classes in Semantic-KITTI dataset [262] and number of points from each class. *Image source: semantic-kitti.org/dataset*.

For this study, we will use the semantic annotations obtained from the SemanticKITTI dataset; in order to filter the original LiDAR point clouds and evaluate the odometry output using the localization reference obtained from the KITTI odometry benchmark.

3.2.2 Input filter configurations

In order to evaluate the performance of LiDAR odometry, we propose six different filtering configurations. A new set of filtered point clouds is created by applying each filtering configuration to the original LiDAR point clouds. Afterward, the filtered point clouds are fed to the selected odometry algorithm, and the localization output is evaluated. The six different filtering configurations used in the experiments and the motivation to include each one of them are presented below.

(A) **Raw**: Original point cloud from the LiDAR sensor in the KITTI odometry sequences. This configuration constitutes the baseline of the comparison.

(B) **Dynamic**: All dynamic objects are removed from the point cloud. Moving objects are a common source of correspondence errors, and other works already implement this type of filter before computing the point cloud registration [146]. Therefore, we expect improvement in the odometry accuracy after removing points from this type of object.

(C) **Dynamic Vehicles**: In this configuration, only the dynamic vehicles are removed. Other dynamic objects (i.e., pedestrians and cyclists) are kept. This

filtering configuration is motivated by the fact that pedestrians should have a minimal contribution to the overall error due to their small size and low velocity.

(D) **Far**: The points with a distance from the vehicle greater than 30m are removed. LiDAR sensors can provide a high point density and detailed measurements at close distances, but point clouds become more sparse with the distance. This fact leads us to think that objects that are far away will have a very small number of points; and, hence, may introduce more noise to the system. This configuration keeps the points that are close to the vehicle, where the point cloud is dense and rich in details.

(E) **Ground**: The ground points are removed, including drivable and non-drivable areas. This filtering configuration has a special interest because the effect of the ground points might be beneficial and harmful for different reasons. On the one hand, LiDAR point clouds are arranged by rings, and those rings that hit the ground will always have a similar appearance; if the ground is flat. Unfortunately, most roads are flat, so this effect might result in misconceiving the vehicle translation because all those ground points will always have the same local coordinates, even if the vehicle is actually moving. On the other hand, the points that lay on the ground should be useful to estimate the rotation of the vehicle, especially the pitch and roll angles.

(F) **Structures**: Nothing but the structures are left in the point cloud. All points are removed, except those points laying on buildings and objects such as traffic signs or poles. The motivation of this test case is to analyze the outcome when only the truly static objects and structures are present in the point clouds. This configuration removes most points from the point cloud, including the ground, vegetation, and all vehicles.

After the pre-filtering process, the newly generated data is used as the input of the LiDAR odometry method, i.e., LOAM. This filtering process also reduces the total number of points contained in each point cloud, leaving smaller point clouds that are easier to process. The obtained downsizing with respect to the original point cloud is presented in Figure 3.3, which shows the percentage of the total number of points for each configuration, averaged over all the KITTI odometry sequences.

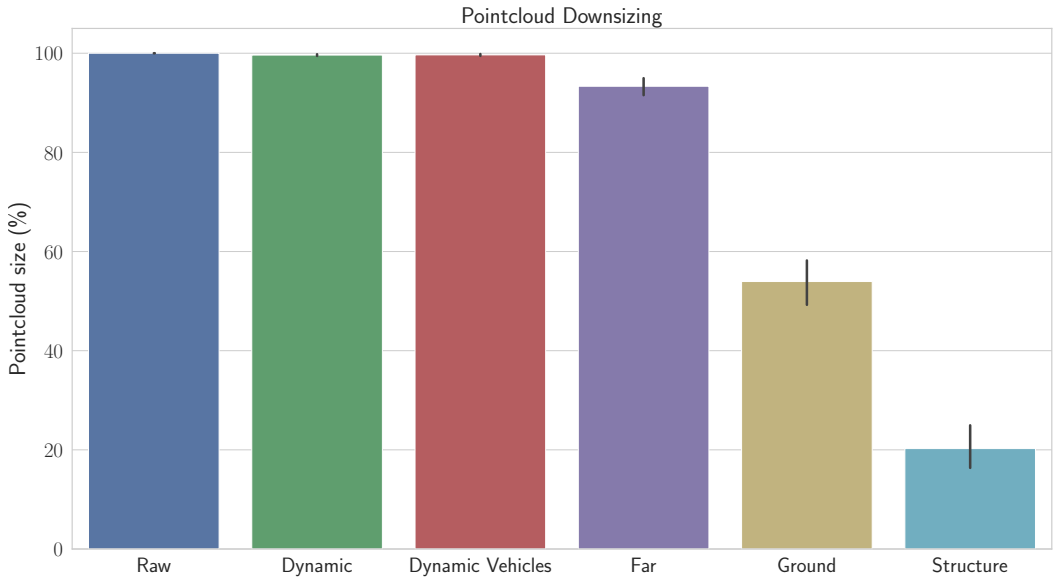


Fig. 3.3: Point cloud downsizing for each filtering configurations.

3.2.3 Evaluation metrics

After generating the new filtered sequences and executing the odometry algorithm for each data set, the estimated poses are evaluated by comparing them with the reference poses extracted from KITTI odometry ground truth. This evaluation process has been performed with the aid of *evo* [264], which is an open-source software tool to analyze and evaluate odometry data. *Evo* has multiple features to work with odometry data, allowing us to visualize the trajectories, compare and evaluate the data based on multiple metrics and finally perform an insightful analysis of the obtained odometry estimation.

As previously noted in Section 2.5, metrics for localization systems are a subject being in a continuous study. Most LiDAR-based odometry algorithms are based on SLAM methods. Therefore, we propose two different metrics that are often adopted to evaluate localization algorithms. The first one is the APE metric; to directly compare the absolute differences between each pair of poses. This metric is suitable for evaluating how good is the performance on a global scale, providing meaningful information about how consistent is the SLAM algorithm because the error will increase if the estimation separates from the reference. However, APE metric has several problematics with drift-prone methods; since even a small error at the beginning of the trajectory will affect the whole sequence of poses. For that reason, we also propose to use the RPE metric, which is more suitable for

the evaluation of the estimation's drift. The RPE metric compares the odometry deltas instead of comparing the poses directly, resulting in a local evaluation of the estimated trajectory.

Let $\hat{x}_i$ be a pose from the estimated trajectory, where $i \in 1:T$ is the timestamp of each pose in the trajectory. Accordingly, let x_i be the reference pose from the ground truth associated with that estimation. Furthermore, if we define $\delta_{i,j}$ as the relative transformation from pose x_i to pose x_j, we can formulate APE and RPE metrics using the following equations:

$$\begin{aligned} APE_i &= \hat{x}_i \ominus x_i \quad \forall iin1:T \\ RPE_{i,j} &= \hat{\delta}_{i,j} \ominus \delta_{i,j} \end{aligned} \tag{3.1}$$

Note that, while the APE metric considers all poses when computing the error, this is not the case for the RPE metric. The set of pose pairs $\{i,j\}, i<j$ is defined as the list of all contiguous sub-sequences of a fixed length in the vehicle's trajectory, in a similar manner as the KITTI benchmark does.

Finally, the error obtained from each one of the selected metrics is decomposed into its translation and rotation components:

$$\begin{aligned} E_{trans} &= \|E\|_2 \\ E_{rot} &= \angle[E] \end{aligned} \tag{3.2}$$

3.3 Results and discussion

All the filtering configurations have been tested on the first eleven sequences from the KITTI odometry benchmark, which are the sequences where ground truth is provided. All these experiments result in a voluminous amount of odometry data that must be properly analyzed. This section will first synthesize and display the extracted results. Then, a thorough analysis is performed, and the results from each configuration are discussed.

3.3.1 Results

The results from each testing configuration are presented in Table 3.1 and Table 3.2 for APE and RPE, respectively.

Tab. 3.1: Absolute Pose Error (APE) obtained for each configuration on the available KITTI odometry sequences.

		Configuration					
	Sequence	A	B	C	D	E	F
Translation (m)	00 (Urban)	21.8 (18.5)	**21.3 (17.4)**	22.1 (18.4)	41.2 (36.4)	26.8 (26.2)	36.6 (26.2)
	01 (Highway)	95.5 (69.1)	93.6 (67.9)	93.4 (67.6)	770.9 (569.1)	97.4 (71.6)	**41.0 (34.4)**
	02 (Urban)	142.3 (115.6)	143.3 (116.1)	142.9 (115.4)	179.4 (143.3)	**91.5 (65.0)**	16 656 (12 752)
	03 (Country)	4.8 (3.5)	4.7 (3.4)	4.7 (3.4)	6.2 (5.0)	**3.8 (2.5)**	15.8 (16.5)
	04 (Country)	2.8 (2.0)	3.0 (1.9)	3.0 (1.9)	17.1 (3.9)	**2.7 (1.3)**	5.4 (5.8)
	05 (Country)	12.2 (9.6)	12.1 (9.6)	12.3 (9.7)	22.2 (20.8)	10.0 (8.7)	**8.6 (8.3)**
	06 (Urban)	2.8 (1.9)	2.8 (1.9)	2.8 (1.7)	4.3 (1.9)	**1.6 (0.8)**	4.5 (6.1)
	07 (Urban)	2.8 (1.7)	2.8 (1.7)	2.8 (1.7)	4.1 (2.0)	**2.5 (1.2)**	7.3 (3.7)
	08 (Urban)	35.6 (30.2)	35.0 (30.0)	35.0 (30.0)	72.0 (59.3)	**34.2 (29.4)**	202.3 (162.2)
	09 (Urban)	15.6 (9.0)	**15.3 (8.8)**	15.5 (9.0)	28.6 (17.9)	17.1 (9.4)	2672 (1725)
	10 (Country)	12.9 (7.5)	12.9 (7.5)	12.9 (7.5)	12.9 (7.6)	**5.5 (2.4)**	141.2 (85.1)
Rotation (°)	00 (Urban)	6.0 (3.5)	**5.6 (3.3)**	6.2 (3.6)	11.7 (7.0)	7.8 (5.1)	11.7 (4.0)
	01 (Highway)	4.3 (1.8)	4.2 (1.6)	4.1 (1.6)	19.3 (19.6)	**3.1 (1.7)**	4.3 (1.6)
	02 (Urban)	31.0 (21.8)	31.2 (21.9)	31.1 (21.8)	38.2 (25.8)	**19.6 (11.8)**	118.9 (44.6)
	03 (Country)	1.9 (0.8)	1.9 (0.8)	1.9 (0.8)	2.5 (1.0)	**1.5 (0.6)**	3.7 (2.5)
	04 (Country)	**0.7 (0.3)**	0.8 (0.3)	0.8 (0.3)	2.9 (0.8)	0.8 (0.2)	6.3 (8.3)
	05 (Country)	3.9 (2.1)	3.9 (2.1)	3.9 (2.1)	7.5 (4.6)	**3.6 (2.0)**	4.4 (2.4)
	06 (Urban)	1.4 (0.6)	1.3 (0.6)	1.3 (0.6)	1.8 (0.7)	**1.2 (0.6)**	2.5 (1.1)
	07 (Urban)	**1.7 (0.7)**	1.7 (0.8)	1.7 (0.7)	2.3 (1.1)	1.8 (0.6)	4.9 (1.8)
	08 (Urban)	7.4 (4.0)	7.3 (3.9)	7.3 (4.0)	14.4 (7.2)	**7.1 (3.9)**	81.3 (32.7)
	09 (Urban)	3.7 (1.8)	**3.6 (1.7)**	3.7 (1.8)	6.9 (3.6)	4.1 (1.8)	123.0 (40.4)
	10 (Country)	2.6 (1.0)	2.6 (1.0)	2.6 (1.0)	3.2 (1.6)	**1.6 (0.8)**	30.3 (5.2)

Tab. 3.2: Relative Pose Error (RPE) obtained for each configuration on the available KITTI odometry sequences.

		Configuration					
	Sequence	A	B	C	D	E	F
Translation (m)	00 (Urban)	1.38 (0.87)	1.39 (0.85)	1.38 (0.85)	1.52 (0.73)	**1.26 (0.79)**	2.89 (1.72)
	01 (Highway)	5.93 (14.58)	5.93 (14.90)	5.97 (14.99)	430.36 (472.06)	9.83 (23.05)	**1.78 (0.75)**
	02 (Urban)	7.68 (22.98)	7.71 (23.05)	7.66 (22.92)	9.28 (27.71)	**2.58 (5.00)**	45.84 (32.42)
	03 (Country)	0.92 (0.44)	0.93 (0.45)	**0.92 (0.44)**	1.29 (0.49)	0.97 (0.48)	4.69 (2.41)
	04 (Country)	**1.25 (0.31)**	1.27 (0.31)	1.27 (0.31)	2.09 (2.26)	1.39 (0.37)	4.37 (5.13)
	05 (Country)	1.26 (0.53)	1.26 (0.54)	1.26 (0.54)	1.44 (0.59)	**1.16 (0.57)**	1.44 (0.80)
	06 (Urban)	1.21 (0.43)	1.21 (0.43)	1.20 (0.43)	1.41 (0.52)	**1.12 (0.48)**	1.36 (0.75)
	07 (Urban)	1.10 (0.59)	1.09 (0.59)	**1.09 (0.59)**	1.24 (0.59)	1.23 (0.68)	2.42 (1.40)
	08 (Urban)	1.48 (0.70)	1.49 (0.70)	1.49 (0.71)	1.68 (0.81)	**1.34 (0.72)**	4.77 (5.84)
	09 (Urban)	1.28 (0.41)	1.27 (0.42)	1.28 (0.42)	1.62 (0.55)	**1.11 (0.48)**	60.84 (23.32)
	10 (Country)	1.44 (0.52)	1.43 (0.52)	1.43 (0.52)	1.59 (0.48)	**1.28 (0.59)**	1.63 (1.36)
Rotation (°)	00 (Urban)	**1.28 (0.76)**	1.29 (0.76)	1.28 (0.75)	1.36 (0.68)	1.46 (0.70)	4.10 (1.85)
	01 (Highway)	0.92 (0.77)	0.89 (0.71)	0.89 (0.69)	30.77 (23.92)	**0.82 (0.52)**	1.05 (0.63)
	02 (Urban)	3.35 (7.18)	3.37 (7.21)	3.35 (7.18)	3.73 (7.29)	**1.71 (1.80)**	79.58 (54.90)
	03 (Country)	0.92 (0.34)	0.92 (0.33)	0.92 (0.33)	1.25 (0.40)	**0.90 (0.35)**	3.62 (2.31)
	04 (Country)	**0.47 (0.25)**	0.50 (0.22)	0.50 (0.22)	1.08 (0.54)	0.53 (0.28)	7.18 (8.05)
	05 (Country)	**1.00 (0.58)**	1.01 (0.58)	1.01 (0.58)	1.19 (0.69)	1.32 (0.65)	2.34 (0.92)
	06 (Urban)	0.87 (0.41)	0.86 (0.41)	**0.86 (0.41)**	1.05 (0.41)	0.87 (0.54)	1.6 (0.79)
	07 (Urban)	1.08 (0.46)	1.08 (0.46)	**1.07 (0.46)**	1.09 (0.48)	1.69 (0.63)	4.17 (1.30)
	08 (Urban)	1.26 (0.69)	**1.26 (0.69)**	1.27 (0.69)	1.50 (0.81)	1.41 (0.66)	9.50 (13.50)
	09 (Urban)	0.91 (0.52)	**0.91 (0.51)**	0.91 (0.52)	1.33 (0.64)	1.15 (0.59)	100.13 (54.49)
	10 (Country)	0.95 (0.59)	**0.95 (0.60)**	0.95 (0.60)	1.08 (0.59)	1.02 (0.57)	2.55 (3.82)

Each table is divided into two sections for the translation and rotation components of the error. Then, each column contains the results of each of the filtering con-

figurations (A-F) previously defined in Section 3.2.2 and each row corresponds to each KITTI odometry sequence. While the APE metric is computed for each pose in the vehicle trajectory, as stated before, the RPE is obtained from each sub-sequence with a fixed length of 100m, following the same metric setup as the KITTI odometry benchmark. Therefore, both tables present the mean and standard deviation of the poses across each trajectory, showing the standard deviation between parenthesis. In addition, the filtering configuration with the best performance for each sequence is highlighted in bold text.

Finally, Table 3.3 displays a summary of the filtering configurations with the best performance for each metric in each odometry sequence.

Tab. 3.3: Best filtering configuration grouped for each metric and each KITTI odometry sequence.

Sequence	Metric			
	APE (Trans.)	**APE (Rot.)**	**RPE (Trans.)**	**RPE (Rot.)**
00 (Urban)	B (Dynamic)	B (Dynamic)	E (Ground)	A (Raw)
01 (Highway)	F (Structure)	E (Ground)	F (Structure)	E (Ground)
02 (Urban)	E (Ground)	E (Ground)	E (Ground)	E (Ground)
03 (Country)	E (Ground)	E (Ground)	C (Dyn. Vehicles)	E (Ground)
04 (Country)	E (Ground)	A (Raw)	A (Raw)	A (Raw)
05 (Country)	F (Structure)	E (Ground)	E (Ground)	A (Raw)
06 (Urban)	E (Ground)	E (Ground)	E (Ground)	C (Dyn. Vehicles)
07 (Urban)	E (Ground)	A (Raw)	C (Dyn. Vehicles)	C (Dyn. Vehicles)
08 (Urban)	E (Ground)	E (Ground)	E (Ground)	B (Dynamic)
09 (Urban)	B (Dynamic)	B (Dynamic)	E (Ground)	B (Dynamic)
10 (Country)	E (Ground)	E (Ground)	E (Ground)	B (Dynamic)

Merging the results of all odometry sequences is not a straightforward task. Due to the nature of the APE metric, the error values presented in Table 3.1 depend on how long is the vehicle trajectory. Therefore, it is not possible to directly combine these measurements. However, considering that the RPE metric is computed for fixed sub-sequences of 100m, the results obtained with this metric can indeed be combined, allowing us to better analyze the performance of each configuration in a broader view. Consequently, the data from extracted from the RPE metric has been merged for each filtering configuration, and the distribution of the error is presented in Figure 3.4 and Figure 3.5 for translation and rotation, respectively. In these boxplot graphs, it is clearly appreciated that the first three configurations have very similar results, as we could intuit from Tables 3.1 and 3.2. Nevertheless, the last three filtering configurations feature more significant differences.

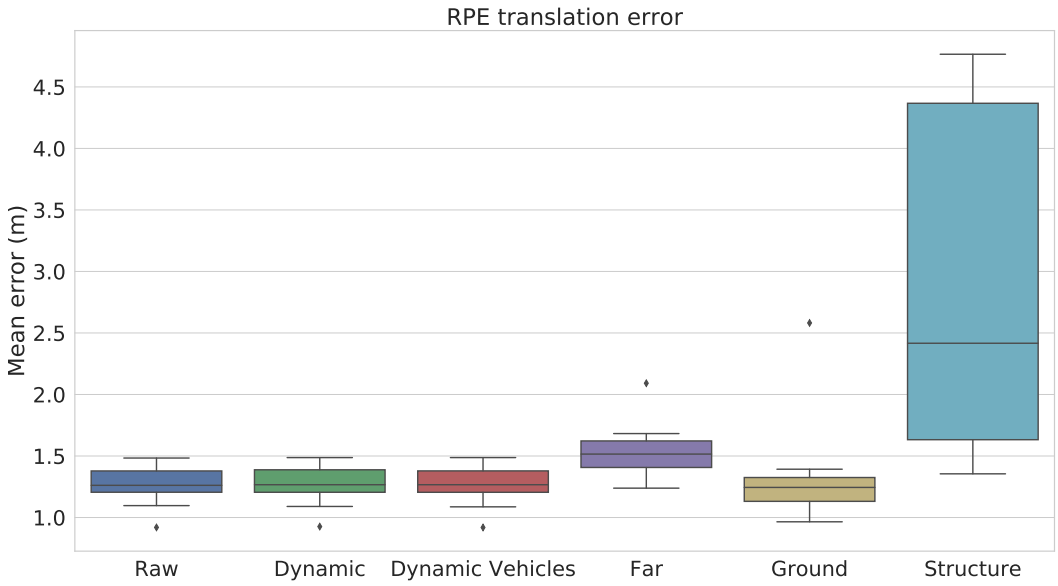


Fig. 3.4: Distribution of RPE translation metric for each filtering configuration across all KITTI odometry sequences.

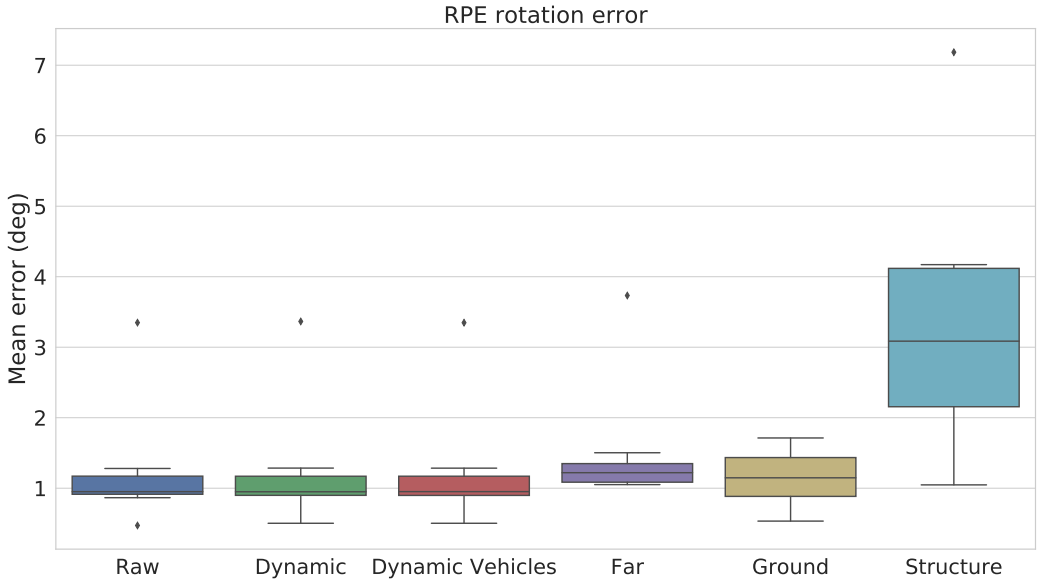


Fig. 3.5: Distribution of RPE rotation metric for each filtering configuration across all KITTI odometry sequences.

As stated before, the proposed filtering configurations reduce the total number of points in each point cloud. This should result in a decrease in the processing time; because the odometry algorithm must check a smaller amount of points. For this reason, we also analyzed the processing time that LOAM took to process each LiDAR measurement. Because LOAM implementation is divided into different processes that

are executed concurrently, the time of each individual process is aggregated. Figure 3.6 presents the collected processing times, averaged for each filtering configuration. Additionally, the standard deviation of each measurement is represented by a vertical line. Please note that the actual processing time is much smaller since this is the aggregation of the different concurrent processes. As expected, the values shown in Figure 3.6 resemble the point cloud downsizing of each configuration, as presented in Figure 3.3.

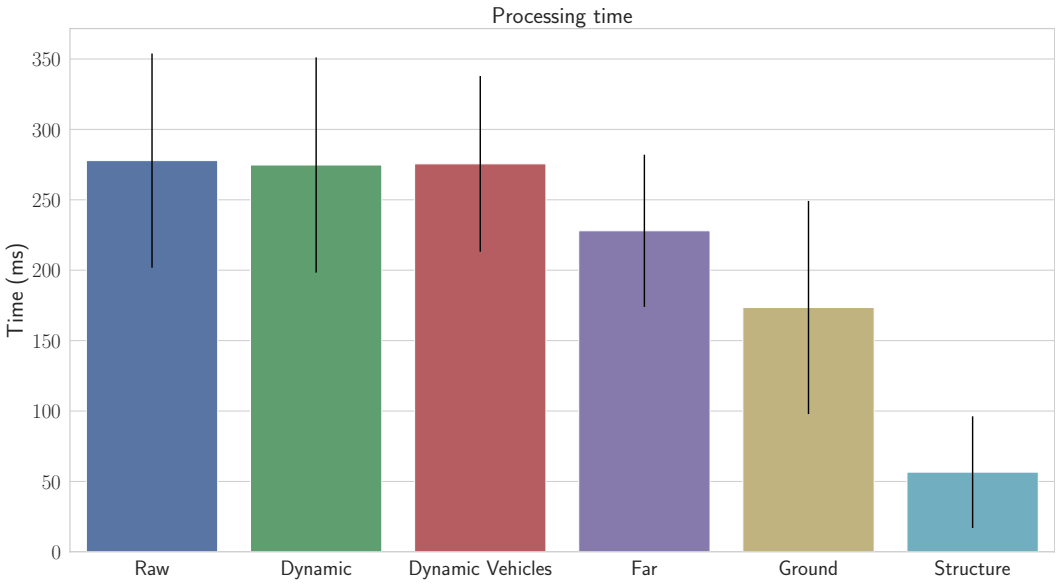


Fig. 3.6: Computational time required by LOAM to process each measurement for each of the filtering configurations.

3.3.2 Discussion

After performing a deep analysis of the experimental results obtained from all configurations, several insights can be extracted from them.

First, it has become clear that the number of filtered points has an important contribution to the odometry estimation. As shown in Figure 3.3, the number of points removed in configurations B and C (dynamic objects and dynamic vehicles) is very small. This is because the percentage of points laying on dynamic objects is minimal; due to the reduced number of dynamic objects and their relatively small size. Therefore, the point clouds for those configurations have a size close to 99% of the original point cloud size. In consequence, the errors computed for those filtering configurations are very close to the baseline. However, other filtering configurations that perform heavier filtering, such as configurations E

and F (ground and structures), present a higher fluctuation in the results. This is especially true for the last configuration, where only the points from structures are kept, reducing the original point clouds up to 10% of their original size. These filtering configurations rely on specific elements in the environment; thus, driving scenarios lacking structures will result in empty point clouds, and the odometry algorithm will be lost for multiple consecutive measurements. Nevertheless, these configurations with heavy point filtering do outperform the baseline and other methods in some specific scenarios. A possible interpretation of these scenarios is that most of the noise sources are removed from the point cloud while leaving the points from the most significant elements. Again, this outcome is only possible in very specific scenarios where a sufficient number of structures is constantly available throughout the trajectory. Finally, it must also be considered that the processing time for these types of filtering configurations is notably reduced, as displayed in Figure 3.6. Therefore, low-cost processing systems could benefit from the reduced computational load required to estimate the vehicle's odometry in real-time.

Regarding the filtering configurations where dynamic objects are removed (B and C), it can be appreciated that they show slightly better results than the baseline. Although the error difference is minimal because the filtered point clouds in these configurations are almost identical to the original ones, overall, the performance is better when removing dynamic objects. Furthermore, between configurations B and C, it seems that the first one, where all dynamic objects are removed, performs better than the second one, where only dynamic vehicles are filtered out of the point cloud. This fact confirms that the contribution of pedestrians and cyclists is not negligible and, hence, these kinds of dynamic objects are also a source of noise in the odometry process. These results seem to confirm our initial hypothesis regarding dynamic objects. However, the actual impact of these objects cannot be properly analyzed because the filtered point clouds are still almost the same as the raw data. For this reason, we think the obtained results are not meaningful enough to reach a definitive conclusion, even though a small improvement is shown. In order to completely validate our hypothesis, a larger dataset with a more dynamic environment would be required.

In regard to the filtering configuration D, where the points that are far from the vehicle are removed, all metrics on every sequence lead to the same conclusion: The odometry results are always worse than the baseline. Even though the point clouds are more sparse at further distances, removing those points limits the amount of information that the odometry algorithm can exploit. Moreover, those points at further distances might be helpful when computing the vehicle's rotation.

One of the most interesting and unexpected observations from the analysis is the variability in the results from the filtering configuration F, where only the structures are kept. As mentioned before, the results from this filtering configuration are generally bad because if the vehicle drives by an area without structures, the odometry algorithm will have no points to register, and the estimation will be lost. Such situations might happen when the vehicle is driving by a road without buildings and with few traffic signs. Nevertheless, a surprising result can be found for sequence 01, which takes place on a highway. Highways are very challenging scenarios for localization algorithms; due to the vehicle's high speed, the lack of reference points, and the amount of other dynamic objects. To our surprise, we found that this filtering configuration obtained remarkably good results, even outperforming the rest of the filtering configurations. After closely analyzing the contents of the original point clouds for that sequence, it can be observed that the highway has guardrails on each side of the road. Guardrails are also considered structures and therefore are labeled as "fence" in the SemanticKITTI dataset. The only other structures present in the highway sequence are traffic signs and traffic panels, which come and go as the vehicle moves but are not always present like the guardrails. By using only the points from these elements, LOAM was able to properly estimate the vehicle's trajectory, outperforming all other filtering configurations. One of the feasible explanations for this case is that most of the points removed with this filtering configuration in this type of scenario are likely to introduce noise into the localization system. In the end, most of the points in a highway scenario will correspond to fast-moving vehicles and the road, which, combined with the speed of the ego-vehicle, will introduce noise and might worsen the estimation quality of the other filtering configurations. Furthermore, configuration F also showed competent results in sequence 05, which consists of a country area with some buildings. The key elements found in this sequence are some small walls that are placed close to the road, in a similar manner to the guardrails on the highway. For this reason, we speculate that the existence of some continuous structures close to the road limits is highly beneficial in the odometry process. Nonetheless, despite providing us with this magnificent insight, the results obtained with this filtering configuration are far from being useful in real-world scenarios because the algorithm gets lost in most of the sequences due to the lack of points after the filtering.

Finally, we bring attention to one of the filtering configurations that initially caused some doubt regarding its potential benefits: The ground filter (filtering configuration E). In Table 3.1 and Table 3.2 it is possible to realize that the removal of ground points is generally beneficial to the odometry estimation. As presented in Table 3.3, among the 44 test scenarios that result from combining all metrics and all

odometry sequences, this filtering configuration is the best performing one in 24 cases. This can also be observed in Figure 3.4 for the RPE translation results. In the rest of the test cases, this configuration still provides competitive results. The only exception is the highway sequence, where the error metrics are higher than the baseline. Following our initial hypothesis, we believe that when the ground is flat and uniform, the ground points produce a negative effect in the translation part of the odometry estimation. When the LiDAR points (which are arranged by rings) hit a flat road, all points in each ring will have the same distance, and each point will always have very similar coordinates. In consequence, even if the vehicle is moving, all those ground points will appear with the same coordinates. Figure 3.7 provides a simplified example of this effect. In that scenario, the points that hit the ground appear to be static, even though the vehicle is moving forward. On the other hand, those points that hit the tree properly reflect the corresponding translation of the vehicle after its movement. The apparent lack of movement presented in the ground points might be incorrectly processed by an odometry algorithm, leading it to believe that the vehicle is not moving. Regarding the rotation error, our initial guess was that the ground points should be beneficial when estimating the vehicle rotation, especially pitch and roll angles. In this case, the results obtained from APE and RPE metrics are slightly contrasting. As one can clearly see in Figure 3.5, the rotation error is affected when the ground points are removed, and the error variance is greatly increased. However, this filtering configuration is the best performing configuration in 7 out of 11 sequences when evaluating the APE rotation metric. To summarize, it is reasonable to conclude that removing the ground points provides better estimations of the translation component of the vehicle's pose. In regard to the rotation error, the tests are not totally conclusive; since each metric provides different results. Nonetheless, this filtering configuration is definitely the one with the better performance overall, even after reducing about half of the points in each point cloud.

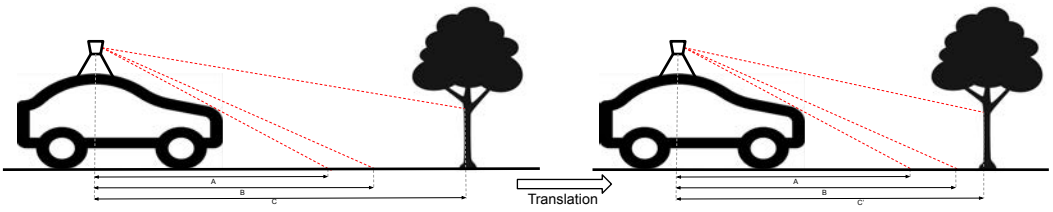


Fig. 3.7: Distance to ground points and distance to object points after vehicle translation.

3.4 Real-world validation

After studying the behavior of the odometry algorithm with the custom filtering configurations, an additional experiment is performed on a real-world platform. In this experiment, we test the performance of LOAM algorithm using the data from a LiDAR sensor mounted on the roof of an intelligent vehicle.

This section will introduce the research driving platform used in the experiment and the 3D semantic segmentation algorithm selected for filtering the point clouds. Finally, the obtained results are discussed.

3.4.1 Research driving platform: ATLAS

ATLAS is a research driving platform for testing and developing autonomous driving technologies for the insurance sector [2]. This platform was jointly developed by MAPFRE-CESVIMAP (a Spanish insurance company), the University Carlos III de Madrid (UC3M), and the Universidad Politécnica de Madrid (UPM). The vehicle platform, presented in Figure 3.8, is managed by a team from the Autonomous Mobility and Perception Laboratory (AMPL) of the UC3M, serving as a versatile platform for testing new algorithms.

Regarding the perception and localization systems, ATLAS is equipped with a GNSS with RTK corrections, an IMU,three LiDAR sensors (one with 32 layers on top and two with 16 layers at the laterals), and three monocular cameras. For the proposed test, however, only the GNSS, the IMU and the 32 layer LiDAR were used.

In order to validate the odometry generated by LOAM, a reference localization trajectory is generated by fusing the RTK-GNSS with the IMU sensor by a UKF.

3.4.2 3D semantic segmentation

While the original tests were performed using data from the KITTI and SemanticKITTI datasets, this experiment resembles a real-world situation; where the ground truth is not available. Therefore, instead of obtaining the 3D semantic segmentation labels from a dataset, this segmentation must be obtained by an algorithm.

The 3D semantic segmentation of the LiDAR point clouds is extracted using Cylinder3D [265], which is based on CNNs. This approach proposes a point cloud

Fig. 3.8: ATLAS research platform.

representation in cylindrical coordinates to extract features and obtain the segmentation of the scene elements. Furthermore, this method is one of the best performing approaches in the SemanticKITTI leaderboard for 3D semantic segmentation, with a mIoU of 67.8%. Finally, the implementation of this algorithm is publicly available, allowing us to integrate it with the rest of the vehicle's software architecture to perform this experiment.

Regarding the training process for the neural networks in Cylinder3D, this method already provides pre-trained weights, which were obtained using the data from the semanticKITTI dataset. However, the LiDAR sensor used to record the KITTI data is a 64-layer LiDAR with a specific vertical FOV, and the LiDAR sensor equipped on ATLAS has only 32 layers with a completely different vertical FOV. Unfortunately, the networks trained with a specific sensor configuration are not expected to behave that well if that configuration changes drastically because the objects in the environment are not similar in the point clouds from the two different sensors.

For this reason, the neural networks have been re-trained using a different database. We selected the *nuScenes* dataset [266], for two reasons. On the one hand, the LiDAR used to record the data in that dataset is similar to the one equipped in ATLAS and is mounted at a similar height, so the perspective is roughly the same. On the other hand, the method Cylinder3D also appears in a good position in the *nuScenes*

leaderboard for 3D semantic segmentation, so it is expected to perform well after training it with this dataset.

A oneshot example of the results of the Cylinder3D semantic segmentation with the LiDAR from ATLAS is presented in Figure 3.9.

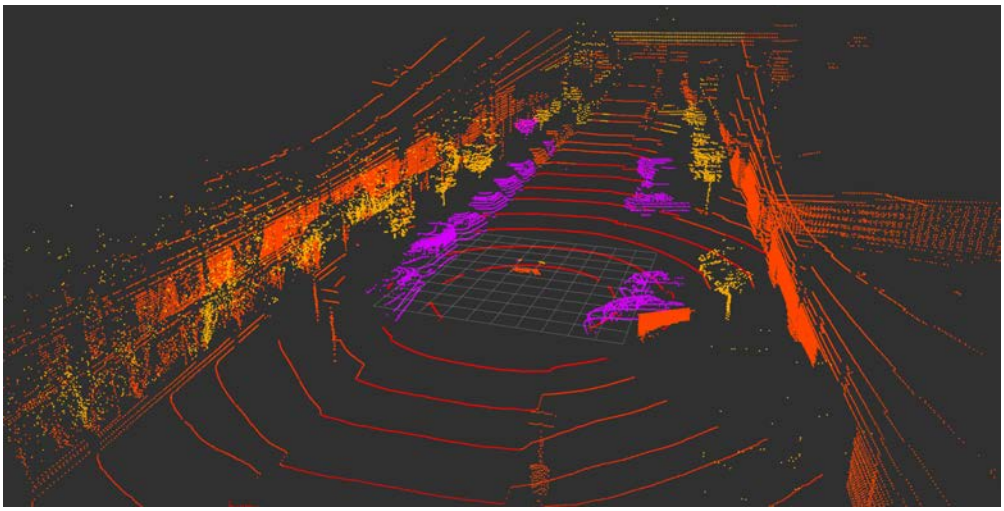


Fig. 3.9: 3D semantic segmentation in ATLAS research platform.

3.4.3 Experimental results

The proposed experiment, using the ATLAS platform, was carried out in an urban area in one suburb in Madrid. The environment was rich in elements, including lamp posts, traffic signs, vegetation, and buildings. By the time the sequence was recorded, the traffic was light, and the number of pedestrians was very limited. Figure 3.10 shows an image taken from one of the onboard cameras from the recorded sequence.

After recording the LiDAR data and the vehicle's localization estimation from the GNSS+IMU fusion system, the LiDAR point clouds were processed using the trained model for Cylinder3D. Then, the extracted semantic information was exploited to filter the point clouds and feed LOAM with the filtered data, as it was done with the data from semanticKITTI. Finally, the obtained odometry was evaluated by using the vehicle's localization system as a reference. Although this localization estimate is not a perfect ground truth, the system's accuracy is high enough to be used as a reference in this experiment.

Following a similar format to the one used when showing the results from the semanticKITTI experiments, the results obtained with each of the localization error

Fig. 3.10: Driving environment from the vehicle's perspective.

metrics are presented in Table 3.4. Being the main difference from the previous tables that this experiment consists of a single sequence; this table shows the mean value of each metric together with the standard deviation of the data between parenthesis. Additionally, the best-performing filtering configuration for each one of the metrics is highlighted with bold text.

Tab. 3.4: Results for each error metric on the ATLAS experiment.

Configuration	Metric			
	APE Trans. (m)	APE Rot. (º)	RPE Trans. (m)	RPE Rot. (º)
A (RAW)	**1.21 (0.60)**	1.79 (0.75)	0.22 (0.12)	0.74 (0.63)
B (Dynamic)	1.29 (0.61)	**1.75 (0.76)**	0.25 (0.13)	0.70 (0.61)
C (Dynamic vehicles)	1.29 (0.61)	**1.75 (0.76)**	0.25 (0.13)	**0.69 (0.61)**
D (Far)	2.09 (1.21)	2.46 (0.84)	**0.21 (0.11)**	0.78 (0.79)
E (Ground)	1.46 (0.81)	1.96 (0.78)	0.22 (0.12)	0.80 (0.72)
F (Structures)	1.47 (0.77)	1.91 (0.91)	0.28 (0.15)	1.11 (0.79)

The first thing that brings our attention is that configurations B (dynamic) and C (dynamic vehicles) obtained the same results. This can be explained because this sequence has a very limited number of pedestrians. Therefore, the configuration where only the dynamic vehicles are removed has the same filter effect as the configuration when all dynamic objects are removed. Similarly, the results obtained from configurations E (ground) and F (structures) are very similar because the

environment is very rich in "structure" elements such as buildings, traffic signs, and lamp posts.

The distribution of each error metric on the different filtering configurations is presented in Figure 3.11 by boxplot graphs. The four plots are categorized as follows: Top and bottom graphs are for APE and RPE respectively, while left and right are for translation and rotation error components, respectively.

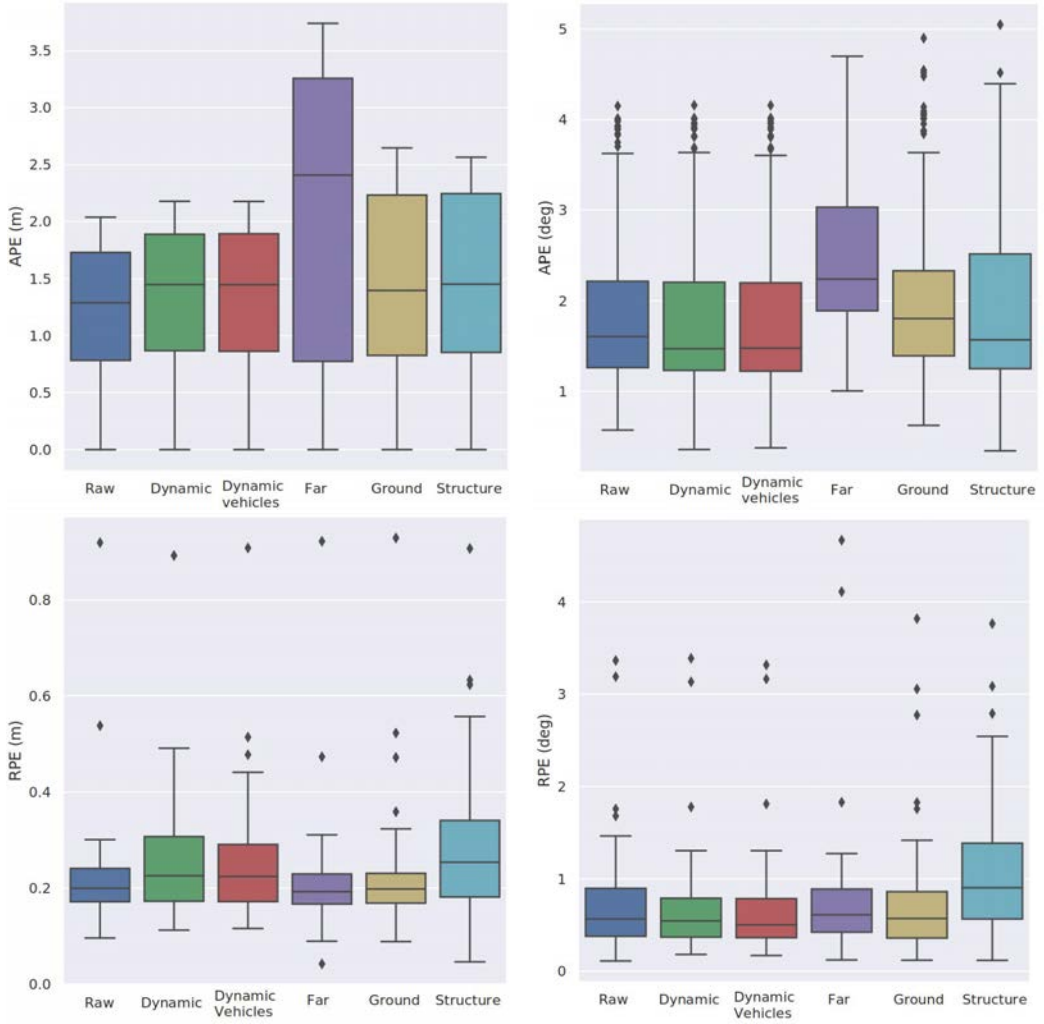


Fig. 3.11: Results for each localization metric on the ATLAS experiment.

The obtained results with ATLAS show some resemblance with the previous results from the experiments using the semanticKITTI dataset. First of all, the difference between configurations B and C is almost non-existent, because the amount of points corresponding to pedestrians is minimal and, therefore, filtering all moving objects or only moving vehicles produces almost the same results. Moreover, the latest filtering configuration, where only structures are kept, shows a much higher

standard deviation and increased error mean, as could be also observed in previous results. Configurations D (far) and E (Ground) produce different results with APE and glsrpe metrics, the removing far points is generally a bad idea, confirming the insights extracted from the analysis with the semanticKITTI data.

Finally, it is worth mentioning that in this experiment the filtering configurations that remove dynamic objects (B and C), are not able to generate an odometry trajectory as good as the baseline configuration (A). This fact was possible to explain after reviewing the 3D semantic segmentation obtained from Cylinder3D. Even though the semantic segmentation results are acceptable, the algorithm tends to flicker, and sometimes the labels of some objects, such as cars, are changed to a different class. This effect is increased at further distances, being this a common problem in most 3D semantic segmentation methods due to the lack of density in the point cloud at further distances. Additionally, most of the 3D semantic segmentation methods are single-shot and do not have a final tracking process to add some temporal consistency to the output. Consequently, these inconsistencies in the semantic labels result in the filtered point clouds having some objects appearing and disappearing over time. Therefore, the odometry algorithm shows a slightly worse performance with those filtering configurations because the point clouds are not consistent. Since these state-of-the-art 3D semantic segmentation techniques are still a relative novelty, it is expected that these problems will be much less occurrent with future, and better, methods, as was demonstrated when evaluating the odometry with the semantic data from SemanticKITTI.

3.5 Conclusions

In this chapter, an in-depth study of the performance of LiDAR odometry when filtering the input point clouds using 3D semantic knowledge is performed. The results obtained from the multiple experiments carried out using the SemanticKITTI dataset corroborate the effectiveness of filtering the raw input data before computing the LiDAR odometry. Particularly, this initial hypothesis regarding the removal of dynamic objects is confirmed, showing promising results. In addition, it has been proven that far-off points provide a beneficial contribution to the matching process and that removing the ground points improves the translation component of the estimated odometry significantly. Furthermore, the suitability of some filtering configurations is bound to the type of environment where the vehicle is driving. Therefore, the results obtained in some particular scenarios, such as highways, reveal that the input data can be drastically downsampled while still achieving better

performance. This performance boost is caused by removing the points from objects that would be more likely to generate spurious correspondences in the matching step.

Although the study was carried out using a representative LiDAR-based odometry method, future work should focus on confirming the generality of the extracted conclusions, including additional odometry approaches. Furthermore, even though the KITTI (and, by extension, the semanticKITTI) dataset includes different types of environments presenting multiple driving situations, this study would also benefit from other types of datasets featuring environments with more dynamic objects. This would ratify the conclusions obtained regarding the importance of removing dynamic objects from the point clouds before computing the odometry.

Finally, a special remark should be included reminding us that this study has been performed using ground-truth semantic information. The reason is that, in order to analyze the actual contribution of each filtering configuration, the semantic-based filter must be perfect. Nevertheless, as it was revealed with the experiment using real-world data, the performance of the proposed approach is affected by the quality of the 3D semantic segmentation. How to effectively exploit the insights extracted from this study to improve the performance of LiDAR-based odometry methods is out of the scope of this chapter and remains open for future work.

4 Validation of Localization Systems

> *Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.*
>
> **— Antoine de Saint-Exupéry**

Autonomous vehicles require a localization system with high accuracy in order to navigate safely. Most works related to localization systems are mainly focused on accuracy. This thesis, however, revolves around a different concept: Reliability. The safety levels and certifications that are tied to road vehicles, in general, and autonomous vehicles, in particular, are strict, demanding, and necessary. When human lives are at stake, an unexpected localization error that may cause an accident must not be tolerated.

As introduced in Section 2.5, obtaining the localization ground truth for an outdoor vehicle is not straightforward. RTK-GNSS solutions are the most common when a reference system is required. However, it is not possible to control their precision, and they depend on factors such as the weather or satellite visibility. Furthermore, existing localization reference systems for outdoor vehicles are unable to guarantee an upper bound in the localization error. Being able to guarantee a maximum error would provide the required robustness and reliability that validation systems for autonomous vehicles demand.

In order to verify that a localization algorithm is functioning properly and to validate its performance for its use in an autonomous vehicle, a robust, accurate, and reliable validation system is required. When studying the trends and typical localization methods used in mobile robotics and indoor positioning systems, it is common to see perception-based solutions that rely on marker or landmark detections to obtain a robust localization estimate. An example of such technologies can be found in Mo-Cap systems, which can provide a high-rate positioning estimate with astonishing

This chapter includes content from [3].

accuracy and robustness. These systems are usually based on infrared cameras and specific markers that are installed on the robot. A different alternative is to invert the Mo-Cap approach by detecting specific static landmarks from the onboard robot sensors.

In this chapter, we propose a novel method to validate localization systems for autonomous vehicles or other outdoor vehicles in general. This method is based on detecting static landmarks (artificial or natural) from the onboard vehicle sensors and estimating the vehicle's pose with respect to the set of landmarks. Additionally, we exploit the a priori knowledge about the landmark measurement model to estimate the final accuracy of the localization estimation, providing a strong a posteriori estimation of the localization error.

Furthermore, a landmark placement optimization algorithm is proposed in this chapter, suitable for generating a localization reference system. This method is focused on controlled environments where new artificial landmarks can be placed. The landmark measurement model is exploited to simulate the testing scenario and find the optimal set of landmarks, such as the expected error of the localization system is guaranteed to stay under a user-defined limit. As a final note, the work presented in this chapter was carried out during an international research stay in IAV GmbH, under the supervision of Dr.-Ing. Ahmed Hussein and has been previously published in [3]. In addition, the software implementation of the designed algorithms has been open-sourced [1].

4.1 Problem statement

A localization system based on landmark detections has some similarities with GNSS-based solutions.

In the case of GNSS systems, the vehicle must be able to receive measurements from at least three satellites in order to estimate a 3D position. With a higher number of visible satellites, the system is able to improve the precision of the position estimate. However, even if the vehicle is receiving measurements from multiple satellites, not all measurements are equally beneficial to the final estimation. If the visible satellites are not properly distributed geometrically, the system might suffer from poor GDOP, degrading the quality of the final position estimation.

[1] Software available at `https://github.com/butakus/landmark_placement_optimization`

Similarly, the precision of landmark-based localization systems also depends on the number of visible landmarks and their geometrical distribution. The difference, however, is that the position of landmarks can sometimes be controlled, as opposed to GNSS satellites.

The idea of modifying the environment to place landmarks in the places so that a landmark-based localization system would perform better has been exploited before.

In [267], different geometrical configurations for landmarks are studied, analyzing their impact on the final localization estimation. The proposed method is based on a multilateration algorithm with signals from WiFi and Zigbee beacons.

The landmark placement is generally considered an optimization problem. Authors in [268] proposed a simple greedy algorithm to reach a feasible solution, which was then validated by an MCL simulation. Additionally, the Hidden Markov Model (HMM) was considered to model the robot movement and landmark measurements. Nevertheless, the reference presented multiple systematic errors, which affected the performance. Afterward, they presented an incremental landmark placement algorithm that considered the robot trajectory to reduce the complexity of the optimization problem [269]. Obviously, the proposed solution is only valid for the previously planned robot trajectory and will fail if the robot takes a different path. In addition, the nonlinear least-squares problem is solved by a linearization of the system that assumes that all uncertainties are Gaussian.

Following the constraints in [269], authors in [270] presented a similar landmark placement algorithm with previous knowledge of the robot trajectory. Instead of mobile robots, the localization approach is applied to small aerial vehicles, using an onboard camera to detect the landmarks. The camera field of view is considered when solving the landmark placement problem. Nevertheless, their proposed solution is still based on a greedy algorithm, leading to a high number of placed landmarks.

A better solution can be found in [271], where a genetic algorithm is designed to solve the landmark placement problem using different trilateration metrics as fitness functions. Instead of only considering the robot trajectory, the optimal placement of landmarks is found by computing an error heat map for the application area. The accuracy is obtained from the determinant of the Fisher Information Matrix (FIM).

The landmark placement problem has been studied previously, mainly in mobile robotics, and several different approaches have been proposed. In order to address the shortcomings of the works that can be found in the literature, we propose a

novel landmark placement method; with the objective of covering the disadvantages of outdoor localization reference systems.

The proposed reference system is composed of two parts. On the one hand is a landmark-based localization system, which is able to provide an estimation of the vehicle's pose with respect to the set of landmarks. On the other hand, there is an offline optimization algorithm; that exploits a simulation of the environment to find the optimal number and position of landmarks. Instead of only considering a predefined vehicle trajectory, we assume that the vehicle can move freely inside the confined testing area. For this reason, we propose a method that exploits the knowledge of the a priori landmark measurement model in order to estimate the localization error at every possible point in the testing area. Consequently, the proposed landmark placement algorithm exploits this error information to find the optimal number and position of landmarks, such as the maximum error in the localization estimate is guaranteed to remain under a user-defined bound parameter, σ_{max} in the whole area.

While the proposed method covers multiple limitations from existing similar approaches, it is not yet perfect and still relies on two assumptions:

1. The vehicle can only move freely inside of a limited environment because only the testing area has been optimized to guarantee the upper error bound in the localization reference. Additionally, the testing environment must have sub-areas where new artificial landmarks can be physically placed.

2. The system requires prior knowledge about the landmark measurement model, i.e., the a priori error distribution of the landmark detection must be known. This information is used to estimate the final a posteriori localization error for a given set of landmarks in the offline environment simulation. In addition to the proposed localization and optimization algorithms, we also present a possible method to calibrate and extract the measurement model for any generic detection algorithm in Section 4.2.3.

Regarding the landmark detection method, the proposed approach is sensor agnostic and does not depend on any specific detection algorithm, as long as the error model is provided. For simplicity, the presented examples in this chapter are based on a LiDAR sensor, although other sensors such as cameras could also be used.

After all, considering the required assumptions, the proposed method can be applied in multiple environments where extra landmarks could be placed, such as proving grounds, parking lots, warehouses, or parks.

4.2 Proposed approach

The whole approach is based on a simulation of the environment where everything can be computed offline, removing real-time limitations. In order to carry out the simulation, it is required a map of the environment; that outlines which are the drivable areas and which areas can be used to place new landmarks. In addition, the environment map may also include neutral spaces, which represent those areas that are both non-drivable and that cannot contain new landmarks. The required map is later on discretized into an occupancy grid map representation. This occupancy grid map includes a set of free (drivable) cells, $\mathcal{F}$, where the vehicle may freely move, and a set of land cells, $\mathcal{L}$, where landmarks can be placed. The resolution of the grid map (or cell size), δ, is parameterizable and must be manually selected according to the computational power and time available. Normally, this selection should allow an acceptable computation time without a significant sacrifice in spatial resolution.

Additionally, the landmark measurement model provides the system with knowledge about the expected error when the vehicle detects a landmark. This error model depends on the type of landmark used, the sensor and detection algorithm used, and the relative position between the sensor and the landmark (distance and angle).

With this information (the map of the environment and the landmark measurement model), the proposed approach presents two major contributions. On the one hand, the accuracy model system is able to generate an accuracy heat map, given a set of landmarks $\mathcal{V}$. This accuracy heat map represents the maximum expected localization error at each drivable cell in $\mathcal{F}$. This system can also be used independently of the landmark placement optimization algorithm; to only estimate the expected localization error in an environment where landmarks are already placed. On the other hand, an optimization algorithm exploits the heat map generation to find a set of landmarks such as the heat map is minimized until an accuracy condition is met. Additionally, the algorithm also tries to maintain the number of placed landmarks as low as possible. The optimization problem is categorized as $\mathcal{NP}$-hard, and finding the global optimum is usually not feasible. For this reason, we introduce a user-defined parameter, σ_{max}, which determines the desired target accuracy. Consequently, the optimization algorithm will stop after a suitable landmark configuration $\mathcal{V}$ which satisfies inequation 4.1 is found.

$$\mathcal{H}^{\mathcal{V}}_{i,j} < \sigma_{max}, \quad \forall\{i,j\} \in \mathcal{F} \tag{4.1}$$

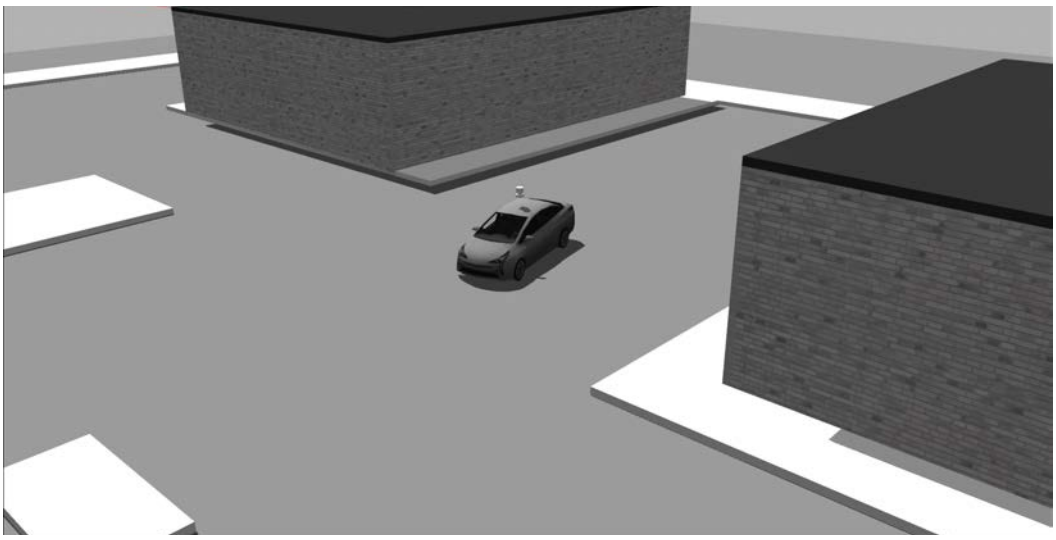


Fig. 4.1: Gazebo simulation environment.

In this inequation, $\mathcal{H}^{\mathcal{V}}$ represents the accuracy heat map, which contains the standard deviation of the localization error at each free cell with coordinates $\{i, j\} \in \mathcal{F}$. With this constraint, it is possible to guarantee that the localization error will always remain below the desired limit across the whole testing area.

4.2.1 Simulation environment

In order to validate the proposed approach, an absolute ground truth of localization and landmark detections is mandatory. For this reason, all the experiments have been carried out in a simulation environment implemented with Gazebo simulator [272]. There, different testing scenarios have been designed with the purpose of calibrating the landmark detection algorithm and validating the proposed landmark placement algorithm.

An example of one of the testing environments implemented in Gazebo is presented in Figure 4.1, where the white areas represent the areas where landmarks can be placed.

In these simulation scenarios, we have included a vehicle model of a Toyota Prius [2], which can be either teleoperated or automatically controlled. On top of the vehicle, we have integrated a model of a 64-layer LiDAR as the main perception sensor, which is a common type of LiDAR used in autonomous driving platforms. Both the vehicle and the LiDAR are fully connected with a Robotic Operating System

[2] Prius Gazebo model available in github.com/osrf/car_demo

(ROS) ecosystem [273]. ROS is used to communicate the algorithms and tools implemented with the Gazebo simulation.

Finally, the types of landmarks used in the experiments are lamp posts, like the one shown in Figure 4.2. These pole-like objects are easy to detect with a LiDAR sensor and are abundant in most driving scenarios. The type of lamp post object that was used has a diameter of 17cm and a total height of 9m.

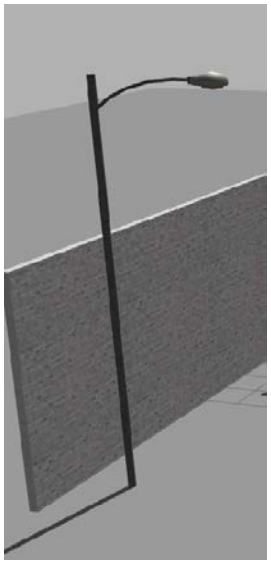


Fig. 4.2: Lamp post model used as a pole-like landmark.

The decision to use the Gazebo simulator instead of other more realistic simulators, such as Carla [274], is based on two facts. On the one hand, the sensor used is a LiDAR, so the quality of the visuals is irrelevant for our simulation. If we were to simulate a camera sensor for the detection of landmarks, a more realistic simulator would be preferable in order to have images with features closer to the real world. On the other hand, the simplicity of Gazebo allows us to modify the environment in runtime without having to recompile the whole map. This is useful when adding new landmarks and updating their position.

4.2.2 Landmark detection algorithm

Before being able to estimate the localization error from the landmark-based localization method, a landmark detection algorithm must be provided. This topic has been widely studied in the literature, and many different approaches have been proposed, as introduced in Section 2.2. This type of detection algorithm relies heavily on the sensor used and the type of landmark that must be detected (e.g.,

poles, traffic signs, or specific structures). Instead of integrating and calibrating one of the state-of-the-art landmark detection methods, a naive LiDAR-based pole detector has been specifically implemented to detect the lamp posts displayed in Figure 4.2. This implementation is oversimplified to work on the provided test cases and shall not be compared with other algorithms presented in the literature. However, a custom implementation allows us to have more control over the whole process and simplify the integration of all the systems. Furthermore, the design of a landmark detection algorithm is out of the scope of this work, even though one is needed in the process.

This customization allows us to simplify the implementation; because it is only required to work with the selected sensor and specific landmarks. The input point clouds have a horizontal resolution of $0.3\,\mathrm{deg}$, and the pole diameter is $17cm$. Therefore, it is guaranteed that a layer from the LiDAR will have at least one point hitting the pole if the distance between them is $32.4m$ at most. At further distances, we might obtain only one point per layer, and the pole could even be missed. For this reason, the maximum detection range is set at $30m$ in order to ensure that the method is reliable and that no landmarks are missed.

The pole detection process, which is illustrated in Figure 4.3, is divided into three steps.

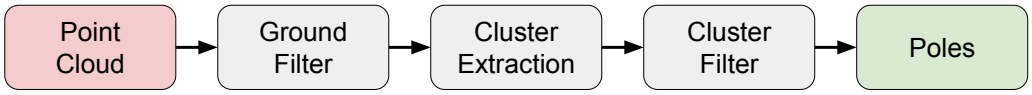


Fig. 4.3: Pole-like landmark detection process.

1. **Ground filtering**: The first step is to remove the points from the ground in order to better segment the objects in the scene. If the environment is flat, as is the case in the designed testing areas, it is possible to remove all the ground by filtering out all points with a z coordinate less than a fixed threshold. In other environments where the ground is not flat, it is possible to implement a different approach based on height maps as done in [11].

2. **Cluster extraction**: Once the ground points are removed from the cloud, it becomes easier to segment the different objects in the environment. This process is carried out by a clustering algorithm based on the euclidean distance between neighbor points. We include additional constraints for the minimum and the maximum number of allowed points in each cluster, according to the size of the lamp post.

3. **Cluster filtering**: The clustering process yields a list of objects that must be processed further, removing those clusters that are not poles, even if the size is similar to the lamp post. This post-filtering step analyzes the height of each of the extracted objects and compares it with the expected height of the pole object at that distance.

This process generates multiple sets of points corresponding to each detected pole. Each of these sets must undergo a final process in order to estimate the center of the pole and generate the final detection output. In this process, clusters are first cleaned, removing the highest points from the object to eliminate the lamp part and keep only the pole. Afterward, the center of mass $\mathbf{x}_m$ of each set of points is computed using equation 4.2, where $\mathbf{x}_i$ are the coordinates of each filtered cluster.

$$\mathbf{x}_m = \frac{1}{N} \sum \mathbf{x}_i \tag{4.2}$$

This center of mass, however, does not correspond to the center of the pole because the LiDAR points hitting the pole are only on the side of the cylinder that is facing the sensor. Therefore, the is an offset distance, d_{offset}, between the detected center of mass $\mathbf{x}_m$ and the actual pole center, $\mathbf{x}_c$, as depicted in Figure 4.4.

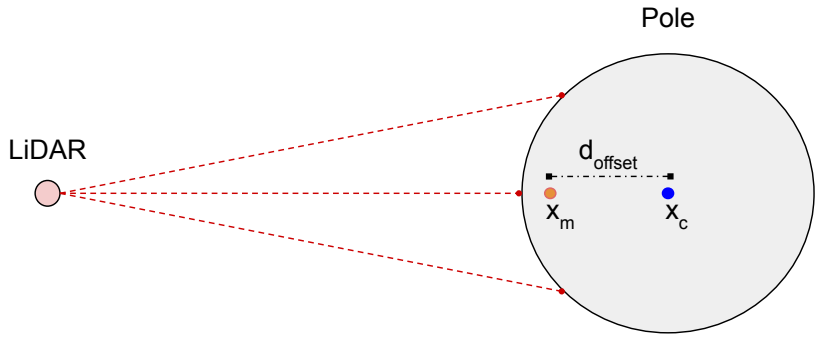
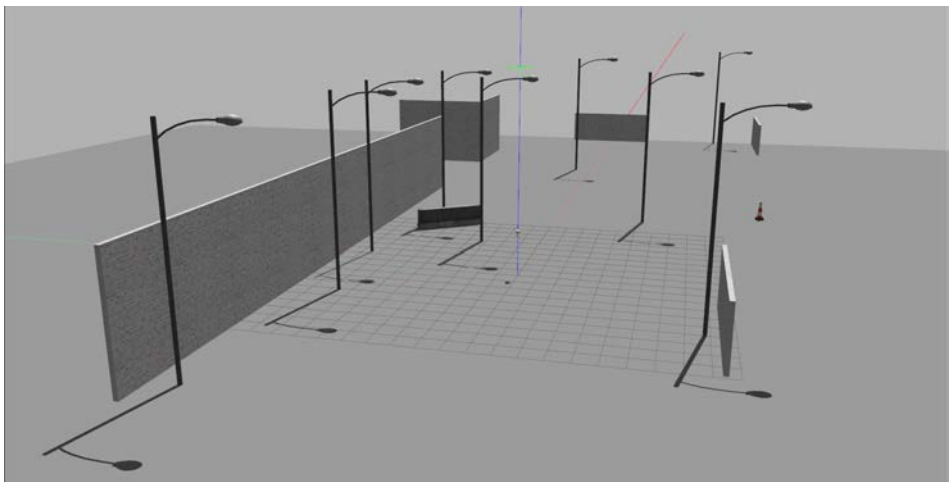
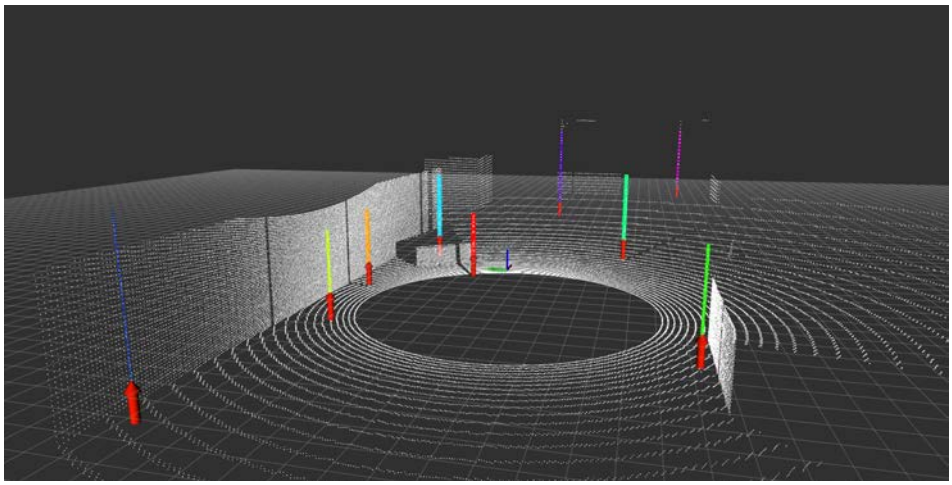


Fig. 4.4: Offset between center of mass and actual pole center.

Since this offset is not fixed because it depends on how the LiDAR rays hit the pole, we have performed a statistical study to find the average deviation between the pole center and the detected center of mass. This study consists in taking multiple measurements at different distances and comparing the detected center of mass with the actual pole center obtained from the simulation ground truth. Finally, the offset is manually added to the detected center of mass, obtaining an estimation of the pole's center.



(a) Simulation environment to validate the landmark detection algorithm.



(b) LiDAR point cloud with segmented poles by different colors. Red arrows mark the detected pole centers.

Fig. 4.5: Simulation environment to validate the landmark detection algorithm.

The implemented algorithm has been validated in a custom testing scene designed in Gazebo, presented in Figure 4.5.

4.2.3 Landmark measurement model

Regardless of the sensor and the landmark detection algorithm used, the most important thing that makes the proposed approach work is a trustworthy model of the landmark measurement process. This model provides the system with knowledge of the probability distribution of the measurement error.

In the heat map generation process, the simulation ground truth is used to generate virtual landmark measurements. The landmark measurement model is used in that step; in order to recreate the simulated measurement by adding an error component as close to reality as possible. The way the proposed system is designed makes the type of noise irrelevant, as long as it can be replicated in the simulation environment. Therefore, the proposed approach is able to deal with non-gaussian error distributions, thus overcoming one of the limitations of most state-of-the-art techniques.

Generally, a landmark measurement is defined as the transformation between the vehicle and landmark poses. If the localization system is working in 3D, then the vehicle pose is represented in a $\mathbf{SE}(3)$ space. The landmark, however, can typically be expressed as a 3D point if its orientation is not relevant. Accordingly, a 2D localization system would use a $\mathbf{SE}(2)$ vehicle pose and a 2D landmark representation. The transformation between the vehicle pose and the landmark point is defined in equation 4.3.

$$\mathbf{z} = \mathbf{l} \ominus \mathbf{x} \tag{4.3}$$

This transformation, representing the landmark measurement, is altered by an error that depends on different factors such as the noise in the sensor measurements or the performance of the landmark detection algorithm. The measurement error model includes knowledge about the statistical distribution of the measurement error, providing an estimation of the expected error based on several variables.

In this Section, we also propose a data-driven calibration method that was designed to find the measurement error model for the landmark detection algorithm presented in Section 4.2.2. Please note that, although the presented results are focused on a LiDAR sensor and pole-like landmarks, the proposed calibration method is can be generalized and applied to other sensor types. Furthermore, this calibration approach should be repeated if the type of sensor is changed, or if the landmarks to be detected are different.

In order to find the distribution of the measurement error, a simulation environment is created with a LiDAR in a fixed position and one lamp post that is placed at different points to take multiple measurements. The recorded data consists of multiple sequences of 10 seconds each, with around 100 point clouds per sequence, since the LiDAR runs at 10Hz. Furthermore, the sequences can be generated by combining different variables according to the following criteria:

- **Distance**: The distance between the sensor and the landmark. It is absolutely necessary to consider this variable in the measurement model.
- **Angle**: The angle between the sensor coordinate system and the landmark. The selected sensor has a FOV of $360\,\mathrm{deg}$, and the horizontal resolution is constant, so this angle should not affect the error. However, it is a good practice to test different angles, so the LiDAR rays do not always hit the object the same way. This variable would have a more meaningful contribution if we were using a different type of sensor, such as a camera, to detect the landmarks.
- **Landmark orientation**: The relative orientation of the landmark with respect to the sensor. This variable is not important for our test case because the landmarks are cylindrical and, therefore, are rotationally invariant. Nevertheless, the landmark's orientation must definitely be considered with other types of landmarks, such as traffic signs, for example. In those cases, the orientation of the landmark with respect to the sensor can influence the detection process.

In order to validate the proposed landmark detection algorithm, we generated multiple sequences by combining a range of distances with a number of random angles. At each distance, which ranged from 1m to 30m with increments of 25cm, 100 random angles were generated, thus producing a total of 11600 landmark poses. Figure 4.6 shows all the landmark positions that were used in the calibration process.

Each one of the positions is recorded for 10 seconds, yielding around 100 measurements per sequence. The sheer amount of data has been manually analyzed through data visualization techniques; in order to understand how the distance and angle variables affect the measurement error.

Due to the nature of the LiDAR measurements, it makes sense to perform the analysis in polar coordinates (i.e., distance and angle) instead of using a cartesian system. For this reason, the identification study will be focused on those two variables.

The first step in this model identification process was to confirm if the measurement error is really invariant to the angle of the measurement. To that end, the mean and standard deviation of the measurement error, grouped by distance and angle, is analyzed. As it can be seen in Figure 4.7, the error in the detected angle does not depend on the landmark angle. The angle error values represented in that graph are also colored by the distance to the landmark, which at first glance seems to be irrelevant as well. Next, the distance error with respect to the landmark angle is analyzed. These results, which are presented in Figure 4.8, are also colored

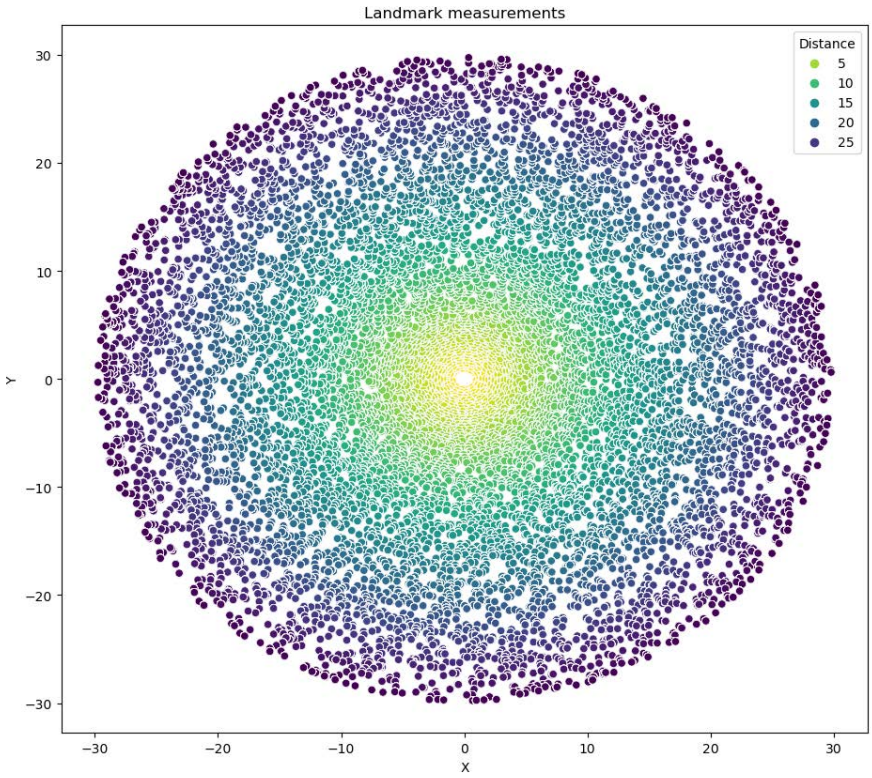


Fig. 4.6: Every landmark position used in the measurement model calibration process.

by landmark distance. Although there seems to be a trend in the error when the distance increases, the distance error is invariant to the angle of the landmark, as can be clearly seen in both the mean and the standard deviation graphs.

With these insights, we can validate the initial hypothesis stating that the angle of the landmark does not affect the measurement error in any of its components. Consequently, the next step consists in studying the measurement error with respect to the distance.

Regarding the angular component of the measurement error, the results obtained are presented in Figure 4.9. In the first subplot, we can see that the error does not seem to be biased and that the dispersion remains more or less constant with respect to the distance, with some oscillations. These oscillations are even more noticeable when visualizing the standard deviation of the error in the second subplot. Although there seems to be a pattern, the standard deviation of the angle error does not increase nor decrease with the distance to the landmark. Therefore, we find it safe to model the angle component from the measurement error with a fixed Gaussian distribution with zero mean and a standard deviation resulting from the average

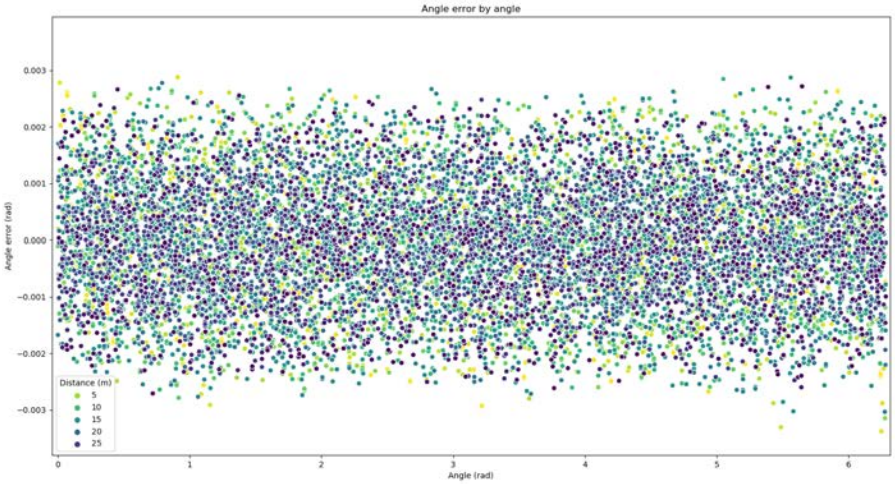


Fig. 4.7: Angle error with respect to the landmark angle. Each point corresponds with the average error of each landmark position. Points are colored depending on the distance to the landmark.

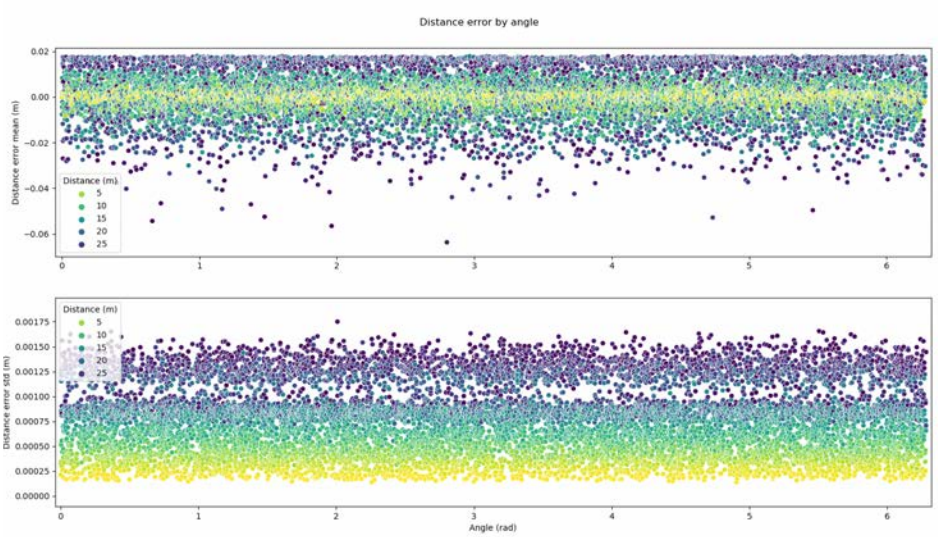


Fig. 4.8: Distance error mean and standard deviation with respect to the landmark angle. Each point corresponds with the mean (top) and standard deviation (bottom) of all measurements from each landmark position. Points are colored depending on the distance to the landmark.

of the standard deviation of the results data for each distance. More precisely, the estimated standard deviation for the angle error is $\sigma_{angle} = 0.001069$ rad.

Finally, the focus is shifted to the distance error with respect to the distance variable. While the angle error remains almost constant on all measurements, we started to perceive in Figure 4.8 that the distance error seems to increase when the distance to the landmark increases. The question to address is how this error increases and

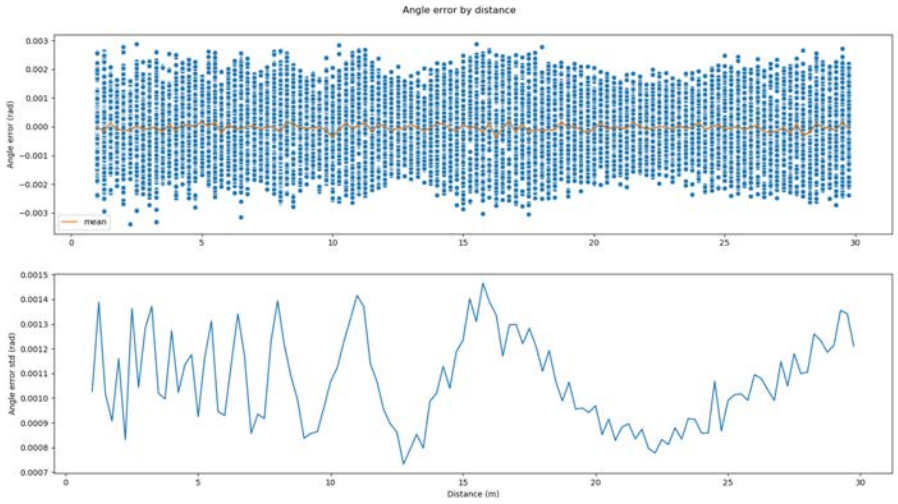


Fig. 4.9: Angle error and standard deviation with respect to the distance to the landmark. Each point in the top subplot corresponds with the mean of all measurements from each landmark position. The subplot at the bottom shows the standard deviation of all the average error points displayed above.

what its distribution is. First, the error of each sequence is presented in Figure 4.10, showing the mean error measurements on the top and their standard deviation at the bottom. The dispersion of the points increases evidently with the distance, as is reflected in the standard deviation. By the looks of the second graph, we could take one more step and suggest that the standard deviation of the distance error increases linearly. Therefore, a linear regression process was carried out, fitting a line with coefficients $a = 0.00054798$ and $b = 0.00070023$. Nevertheless, even though we have determined the relation between the distance error and the distance, the distribution of the error cannot be easily identifiable from Figure 4.10.

In order to further analyze the distribution of the distance error, the data used in Figure 4.10 is used to extract multiple histograms at different distances. These histograms, presented in Figure 4.11, provide more visual information about the distribution of the error data. Unfortunately, the distance error seems to follow a distribution that is not consistent at different distances, thus difficulting the model identification. Consequently, a compromise is made by fitting the data with a Gaussian distribution with zero mean and variable standard deviation that depends on the distance. Although the Gaussian distribution is not the perfect fit for most of the distances, it is certainly good enough for the task.

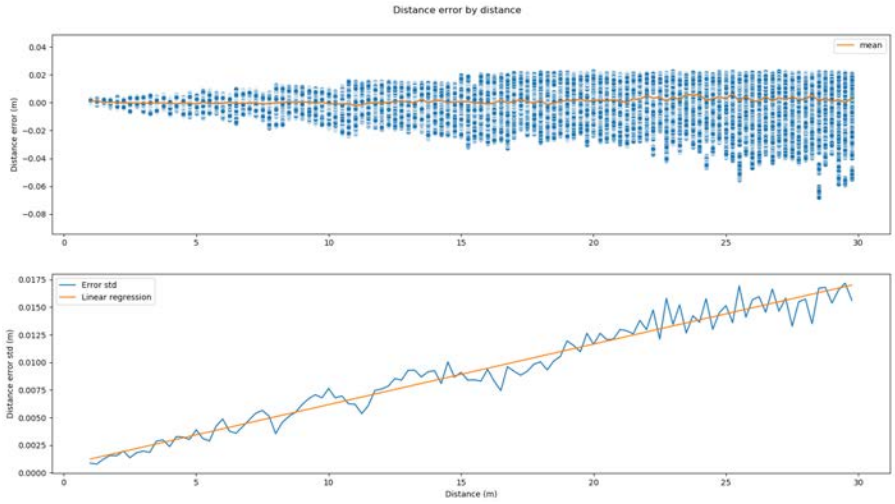


Fig. 4.10: Distance error and standard deviation with respect to the distance to the landmark. Each point in the top subplot corresponds with the mean of all measurements from each landmark position. The subplot at the bottom shows the standard deviation of all the average error points displayed above and the linear model of the standard deviation, estimated by linear regression.

To conclude, the final error parameters estimated for the landmark measurement model are presented in equation 4.4. The standard deviation of the distance error is defined in equation 4.5 with the coefficients obtained by the linear regression.

$$\begin{aligned} E_d &\sim \mathcal{N}(0.0, \sigma_d^2) \\ E_\theta &\sim \mathcal{N}(0.0, 0.001069) \end{aligned} \tag{4.4}$$

$$\sigma_d = 0.00054798 + 0.00070023 \cdot d \tag{4.5}$$

Furthermore, the measurement error can also be modeled as a multivariate Gaussian with covariance Σ_p, defined in equation 4.6, whith $\sigma_\theta = 0.001069$. Note that the correlation terms are zero because no correlation between variables was found when analyzing the data.

$$\Sigma_p = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\theta^2 \end{pmatrix} \tag{4.6}$$

Last but not least, the proposed covariance matrix addresses the measurement error in its polar form. However, the generated landmark detections are expressed in cartesian coordinates, which is the coordinate system used by the rest of the system. Therefore, if this model is to be used to generate virtual measurements from the

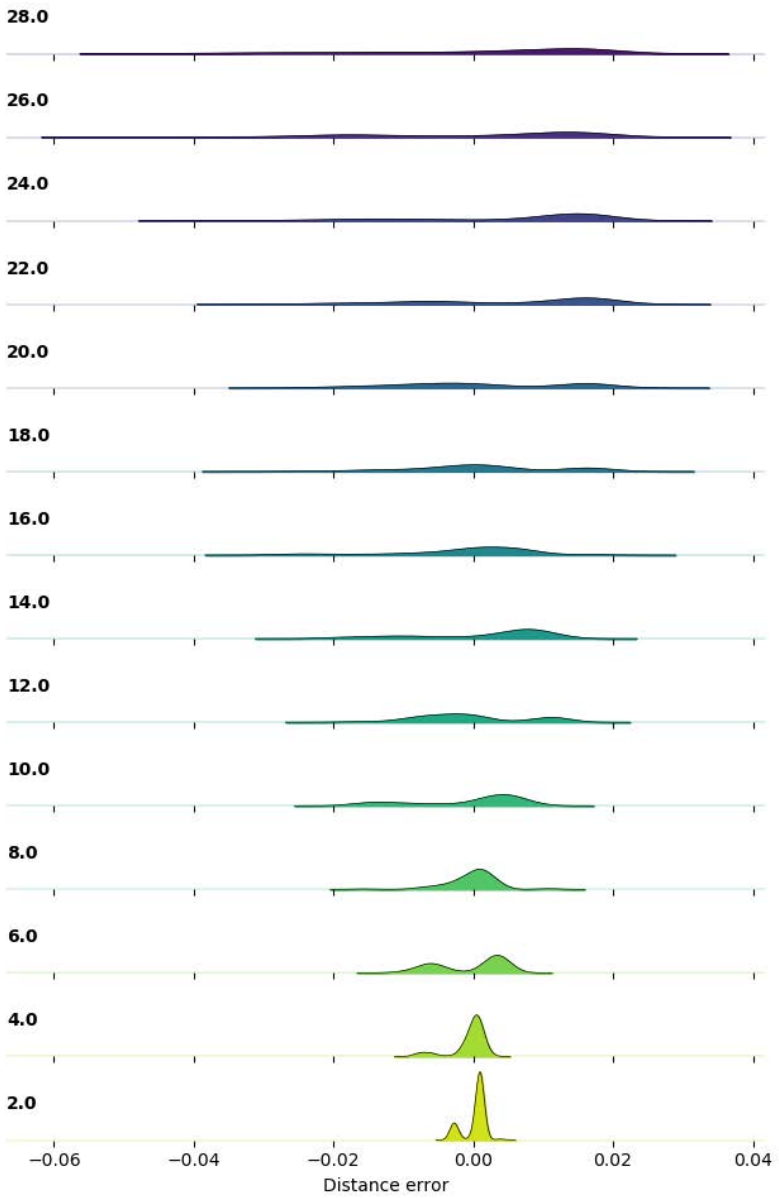


Fig. 4.11: Histogram of distance error at different distances.

simulated ground-truth data, it is required to transform the covariance matrix Σ_p to its cartesian form, Σ_c. Such transformation is defined as follows:

$$\begin{aligned} \sigma_{\theta,d} &= d \cdot \tan \sigma_\theta \\ \Sigma'_p &= \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_{\theta,d}^2 \end{pmatrix} \\ R &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \\ \Sigma_c &= R\Sigma'_p R^{-1} \end{aligned} \tag{4.7}$$

First, the standard deviation of the angle error is scaled with the distance and replaced in the covariance matrix Σ. Then, this covariance matrix is transformed by a rotation matrix, resulting in a covariance matrix representing the same error in cartesian coordinates.

4.2.4 Visibility model

In addition to the landmark measurement model, a visibility model is introduced in the system; in order to only generate virtual measurements for the landmarks that could actually be detected. Given a vehicle pose $\mathbf{x}$ and a set of landmarks $\mathcal{V}$, the visibility model is defined as a function $g(\mathbf{x}, \mathcal{V})$ that returns the subset of landmarks $\mathcal{V}^* \in \mathcal{V}$ that would be observable in the real world from the vehicle pose $\mathbf{x}$. A realistic and reliable visibility model is important to the system, so the simulation does not consider measurements that would not be possible in the real world. Therefore, the following details should be considered when building the visibility model:

- **Sensor characteristics**: The visibility model must consider the sensor limitations, such as the maximum operating range or the field of view.
- **Detection algorithm limitations**: In addition to the sensor's physical limitations, some landmarks might not be detected by the detection algorithm, even if the sensor has information about them. For example, a LiDAR sensor might receive some point measurements from a distant landmark, but if the point cloud density is very low in that area, the detection algorithm might miss the detection.
- **Static elements**: Since the system has access to a map of the environment, the static elements can be considered to determine if a landmark will be occluded from the vehicle pose $\mathbf{x}$.
- **Other landmarks** The visibility model function $g(\mathbf{x}, \mathcal{V})$ takes the whole set of landmarks as an input. Therefore, the occlusions caused by other landmarks can also be taken into account when generating the visibility output.

4.2.5 Landmark-based localization

Localization systems based on landmarks have been widely used in robotics since autonomous mobile systems began to exist. Assuming that the position of the

landmarks is known within a common reference system (usually a map), it is possible to estimate the vehicle's pose from a set of measurements to those landmarks. This nonlinear problem, which is depicted in equation 4.8, consists in finding an estimation for the vehicle pose $\mathbf{x}^*$ such as the residual error function $\mathbf{F}(\mathbf{x})$ is minimized. The most common method that is used in the literature to solve this problem is the NLLS approach.

$$\mathbf{x}^* = \underset{\mathbf{x}}{argmin}\{\mathbf{F}(\mathbf{x})\} \tag{4.8}$$

Next, the process to define the residual function $\mathbf{F}(\mathbf{x})$ is described. Let $\mathbf{Z}$ be a set of landmark measurements from the real vehicle pose $\mathbf{x}$ (which is actually unknown) to each of the visible landmarks. Each landmark measurement, $\mathbf{z}_i$ is defined as a 3D vector, as presented in equation 4.3, and suffers an error with covariance matrix $\mathbf{\Sigma}$. Furthermore, let $\hat{\mathbf{x}}$ be an estimator of the vehicle's true pose, representing the belief that the system has regarding the vehicle's pose. Considering the estimation $\hat{\mathbf{x}}$ and the set of landmarks $\mathcal{V}$, we can generate the set of the expected measurements, $\hat{\mathbf{Z}}$, representing the landmark measurements that would be expected if the vehicle's pose was $\hat{\mathbf{x}}$. This concept can be seen in Figure 4.12, presenting the actual and expected measurements from the true vehicle pose and the estimated pose, respectively.

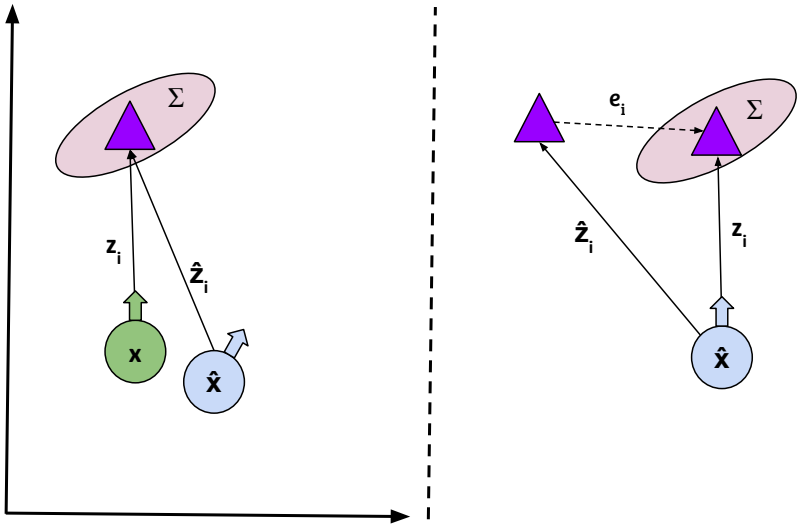


Fig. 4.12: Left: Received measurement from $\mathbf{x}$ and expected measurements from $\hat{\mathbf{x}}$. Right: Difference (error) $\mathbf{e}_i$ between received and expected measurements, with respect to $\hat{\mathbf{x}}$.

As it is also depicted in Figure 4.12, the difference between the received measurements $\mathbf{z}_i$ and the expected measurements $\hat{\mathbf{z}}_i$ is defined as the measurement error.

The measurement error, defined in equation 4.9, is then applied in equation 4.10 to build the resiual function for the measurement system.

$$\mathbf{e}_i = \mathbf{z}_i - \hat{\mathbf{z}}_i \tag{4.9}$$

$$\mathbf{F}(\mathbf{x}) = \sum_i \mathbf{e}_i^T \mathbf{\Sigma}^{-1} \mathbf{e}_i \tag{4.10}$$

Finally, a solution to the nonlinear system can be estimated by solving equation 4.8 with the proposed residual function. The estimation of the optimal $\hat{\mathbf{x}}$ can be obtained using a popular NLLS solver like the Levenberg-Marquardt algorithm [275]. However, algorithms such as the Levenberg-Marquardt method may find problems when the variables to be optimized are not defined in a Euclidean space. This is the case for the rotation component of the vehicle's pose, which is non-Euclidean. In order to overcome this problem, the optimization can be performed on a manifold, as introduced in [178]. In the manifold, the changes in the rotation appear to be locally Euclidean. Therefore, since the Levenberg-Marquardt algorithm performs the optimization taking relatively small steps, the optimization on a manifold approach can be exploited to overcome this problem.

4.2.6 Accuracy metric

One of the two main contributions presented in this chapter is the ability to generate an accuracy heatmap representing the expected localization error at each point in the testing area. In order to determine what is the expected error in every position of the map, having a localization system based on landmarks is not enough. Even if it is possible to obtain an estimation of the vehicle's pose using the proposed localization method, the a posteriori distribution of the localization error must also be estimated.

The a posteriori distribution of the error depends mainly on the a priori error from the landmark measurements. This estimation problem is not straightforward, and simplifications cannot be applied because of the possibility of losing accuracy and because the system should not be limited to a single type of error distribution (e.g., Gaussian error). The problem of estimating the a posteriori error distribution can be tackled by two different approaches. On the one hand, the most basic approach consists in directly sampling from the posterior. This way, it is possible to take multiple measurements and estimate the distribution from all the collected

samples. However, this approach is usually not available in most situations, so it is not common in the literature. On the other hand, when only one measurement is available, the a posteriori distribution can still be estimated using sampling methods such as the Markov Chain Monte Carlo (MCMC) method [276]. This sampling approach builds a Markov Chain and takes multiple steps in that chain to sample and estimate an approximation of the a posteriori distribution. One of the advantages of the MCMC method is that it can deal with non-Gaussian distributions and only requires one single measurement, making it a suitable method for a wide range of applications.

Although the MCMC is a robust method and is suitable for the proposed system, the posterior is just an approximation that consists of a single measurement. On the other hand, directly sampling from the posterior is the most natural way to estimate the distribution of the localization error, and the accuracy of this estimation can be controlled by the number of samples taken. One of the reasons to implement the proposed approach as an offline simulation is having the advantage of being able to repeat multiple virtual measurements with known ground truth. Consequently, it is possible to exploit this advantage and sample the posterior directly, using these samples to determine the distribution of the localization error. This process consists in obtaining multiple sets of landmark measurements from the same vehicle pose and then using each of the measurement sets to estimate the vehicle's pose using the proposed NLLS method. Afterward, all the localization error samples are analyzed to extract the standard deviation of the error, thus obtaining an assessment of the localization accuracy at the testing vehicle pose for the given set of landmarks.

In order to determine the number of samples N required to extract a robust estimation of the posterior, a small simulation experiment was carried out, analyzing the localization error after combining N samples. Figure 4.13 shows the results obtained for the translation part, where the error (top graph) is defined as the absolute difference between the mean of all the samples taken and the ground truth pose of the vehicle. Additionally, the graph at the bottom represents the standard deviation of the samples, providing an estimation of the expected error. It is not difficult to notice how incrementing the number of samples makes the error tend to zero, whilst the standard deviation converges to the a posteriori error std.

Finally, this process can be repeated for each point in the testing area, generating an accuracy heatmap like the one shown in Figure 4.14. To summarize, the combination of a proper landmark measurement model and the offline simulation of the environment allows us to estimate the expected localization error at each cell in the map for a given set of landmarks, $\mathcal{V}$.

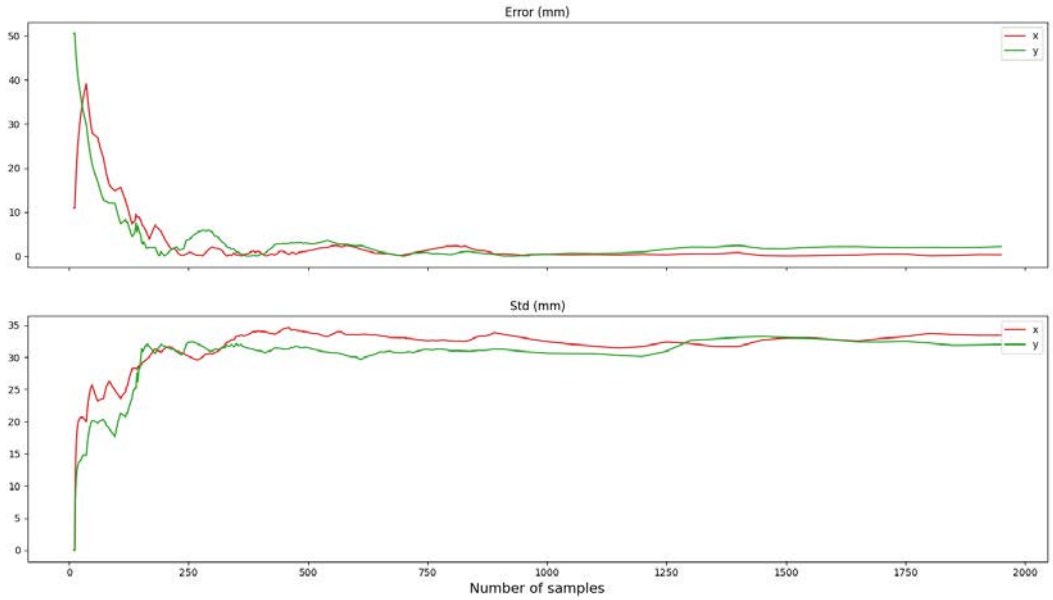


Fig. 4.13: Sampling the posterior of the NLLS pose estimation.

4.2.7 Landmark placement optimization

Being able to estimate the expected localization accuracy in the whole testing area with a given set of landmarks is a major contribution on its own, albeit being only half of the proposed approach. Additionally, we propose an optimization method based on genetic algorithms to solve a landmark placement problem in the testing area. This algorithm has four inputs:

1. Environment map, $\mathcal{M}$, represented as an occupancy grid map containing free cells (where the vehicle can drive) and land cells (where landmarks can be placed).

2. Landmark measurement model, as defined in Section 4.2.3. Usually defined as a covariance matrix if the error is assumed Gaussian.

3. Visibility model, $g(\mathbf{x}, \mathcal{V})$, as defined in Section 4.2.4.

4. Accuracy target, σ_{max}, determining the maximum standard deviation allowed for the localization error. This is a user-level parameter to adjust the algorithm depending on the accuracy required. Note that some accuracy levels might be unreachable if the landmark detection error is too high.

The output of the optimization algorithm is a set of landmarks $\mathcal{V}$ that satisfies inequation 4.1 while trying to use the minimum number of landmarks. The genetic

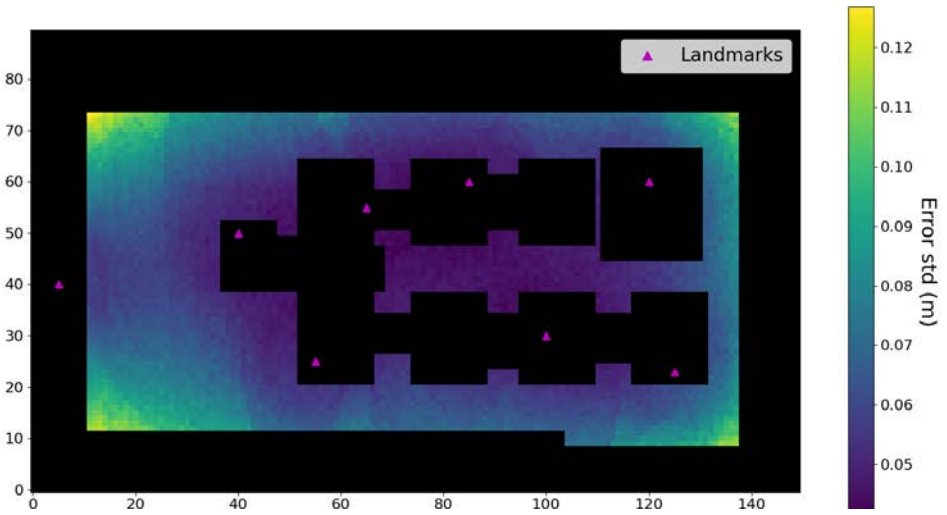


Fig. 4.14: Accuracy heatmap for one of the testing environments. The color of the cells represent the standard deviation of the localization error.

algorithm will exploit the accuracy metric previously proposed to determine if a set of landmarks is able to generate a heatmap (like the one presented in Figure 4.14) where inequation 4.1 is satisfied. After any solution is found, the genetic algorithm will stop iterating.

Nonetheless, this approach requires computing the accuracy heatmaps at every iteration for every set of landmarks that must be evaluated. Evidently, this process requires a gigantic computational effort since the system must simulate the measurements and solve the NLLS problem numerous times for each cell in each map. In consequence, we propose performing the actual optimization using a different metric function based on coverage maps. Although the optimization is performed on the coverage maps, the heatmaps are still periodically computed after a defined number of iterations; in order to estimate the solutions' accuracy and check for the stop condition.

Previous works have used similar metric functions based on landmark coverage to solve the landmark placement problem with successful results [277]. A coverage map is based on the landmark visibility model defined in Section 4.2.4. Consequently, the coverage value of each cell in the map is calculated by counting the number of landmarks that can be detected from that cell. Figure 4.15 presents an example of a coverage map with the same set of landmarks as the one used in Figure 4.14. A close analysis of these two figures can reveal that there is a clear correlation between coverage and localization accuracy.

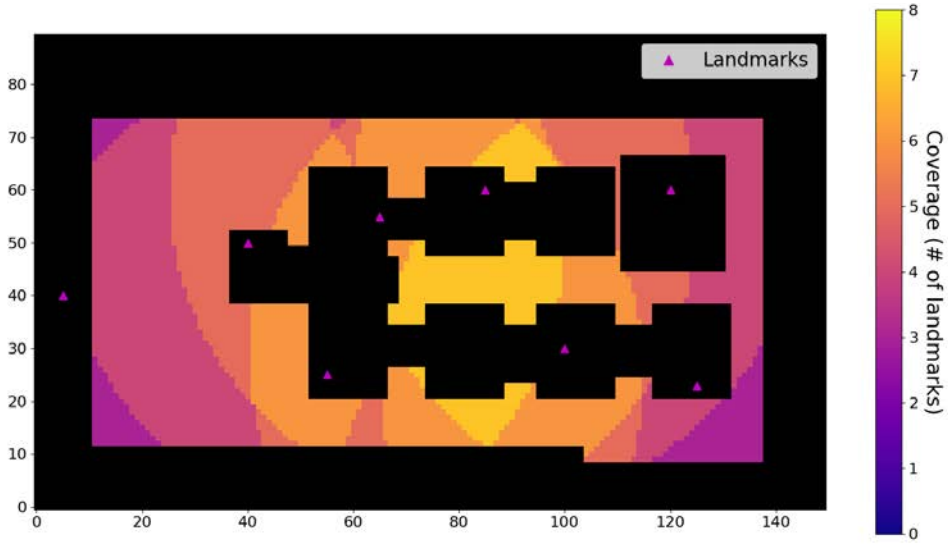


Fig. 4.15: Coverage map for one of the testing environments. The value of each cell represents the number of visible landmarks from that point.

Finally, the genetic algorithm can exploit the proposed coverage maps to optimize the set of landmarks, where the optimization problem now consists in the maximization of a fitness function, defined in equation 4.11, based on the coverage map.

$$Fitness(\mathcal{V}) = \sum_{i,j \in \mathcal{F}} \tanh \frac{\mathcal{C}^{\mathcal{V}}_{i,j}}{k} \tag{4.11}$$

In equation 4.11, $\mathcal{C}^{\mathcal{V}}$ is the coverage map computed with the set of landmarks $\mathcal{V}$, $\mathcal{F}$ is the set of free (drivable) cells, and k is the minimum number of landmark measurements required to obtain a solution for the NLLS problem. The purpose of the hyperbolic tangent process is to grant a higher fitness reward when a low-coverage cell improves its fitness while providing a much smaller benefit if a cell with an already high coverage gets even more. Furthermore, the inflection point in the proposed hyperbolic tangent function equals the parameter k, which is defined as the minimum number of landmarks needed to obtain a localization estimate. This feature allows a fast initialization, granting better fitness rewards to actions that cover the minimum coverage over actions that improve the coverage of areas that are already covered by the minimum number of landmarks.

The presented optimization problem is solved by a genetic algorithm that has been purposely designed for this specific task. The population of the proposed genetic algorithm is a set of elements, $\mathcal{P}$, where each element represents a set of landmarks. Accordingly, each element is composed of multiple genes, where each

gene corresponds to a landmark position. Therefore, the number of genes in each element corresponds to the number of landmarks in that possible solution.

The proposed genetic algorithm is described in Algorithm 1. The main process is composed of an outer and an inner loop. First, the population is initialized, considering the minimum theoretical number of landmarks needed to cover the map area. All elements in the population are initialized following a greedy method that will be presented below. Then, the inner loop is responsible for the coverage optimization while the outer loop manages the number of landmarks, periodically incrementing it. The inner loop process consists in modifying the set of landmarks by applying the genetic process, always maintaining a fixed number of landmarks, N. If the algorithm is not able to find a valid solution with N landmarks after the inner loop has been executed for I_{max} iterations, the inner loop ends, and N is increased. Concurrently, the accuracy heatmaps are computed every 100 iterations in order to check if the target accuracy has been reached.

The genetic process that is executed inside the inner loop is mainly composed of the three common genetic operators (i.e., selection, crossover, and mutation). First, the best elements in the population are selected via a tournament process. This tournament-based selection process takes small subsets of the population and selects the element within the subset with the highest fitness. The set of selected elements is often called the set of parents because the remainder of the population will be generated from their genetic information. After selecting the set of parent elements, a special crossover process is applied, using their genetic information to generate the offspring and complete the new population. This crossover process combines all the parents' genes into a gene pool, meaning that all landmark positions that exist in the set of parents are separated from their original elements and mixed all together in the gene pool. Afterward, each new element from the offspring will be created by combining random genes from the pool, so each one of the new elements can have genes from any of the selected parents. The motivation that leads to this crossover design is that each element is a possible solution to the problem, and elements with high fitness are those that contain good landmark positions. Instead of combining the information from two parents, all the potentially good landmark positions are mixed, and the offspring set is generated from the gene pool instead, allowing a larger genetic variety. Finally, the new elements that form the offspring might be randomly affected by a mutation process, which can affect each of the genes individually. When a gene is mutated, the corresponding landmark position will not be drawn from the pool, but instead, a new landmark position will be generated using the auxiliary greedy algorithm.

Input: Map, $\mathcal{M}$
Output: Optimal landmark configuration, $\mathcal{V}^*$
`/* Maximum number of landmarks */`
Parameter: N_{max}
`/* Number of inner iterations */`
Parameter: I_{max}
`/* Compute the theoretical minimum number of landmarks to have full coverage */`
$N = MinNumberOfLandmarks(\mathcal{M})$
`/* Initialize population */`
$\mathcal{P} = InitializePopulation(N, \mathcal{M})$
while $N < N_{max}$ **do**
  $i := 0$
  while $i < I_{max}$ **do**
    `/* Update population fitness */`
    $Fitness = CoverageFitness(\mathcal{P}, \mathcal{M})$
    `/* Tournament Selection */`
    $Parents = TournamentSelection(Fitness)$
    `/* Offspring generation and mutation */`
    $Offspring = GenOffspring(Parents)$
    $\mathcal{P} = Parents \cup Offspring$
    `/* Local optimization */`
    $\mathcal{P} = LocalOptimization(\mathcal{P})$
    `/* Check accuracy every 100 iterations */`
    if $i \mod 100 = 0$ **then**
      `/* Compute accuracy heat maps */`
      forall $k \in \mathcal{P}$ **do**
        $\mathcal{H}_k := ComputeHeatmap(\mathcal{V}_k)$
      end
      `/* Extract agent with the minimum maximum heat map error */`
      $k_{best} := \underset{\mathbf{k}}{argmin}\{\max\{\mathcal{H}_k\}\}$
      if $\max\{\mathcal{H}_{k_{best}}\} < \sigma_{max}$ **then**
        return $\mathcal{V}_{k_{best}}$
      end
    end
    $i = i + 1$
  end
  `/* Add new landmark to each agent based on greedy algorithm */`
  forall $k \in \mathcal{P}$ **do**
    $\mathcal{V}_k := GreedyAddLandmark(\mathcal{V}_k, \mathcal{M})$
  end
  $N = N + 1$
end
`/* If no solution is found after adding` N_{max} `landmarks, return an empty set */`
return $\varnothing$

Algorithm 1: Genetic algorithm for the landmark placement problem

After going through the three typical genetic operators, the new population of solutions is finally constituted. Nonetheless, before computing the population fitness and finishing the current iteration of the inner loop, the new population is subject to a local optimization procedure. The purpose of this last process is to allow every element in the population to improve its own fitness, emulating a training process before the next selection tournament. The local optimization procedure, illustrated in Figure 4.16, consists of a simple process where each of the landmarks is allowed to move one cell from its original position. If the new landmark position results in a better fitness value, the landmark is moved to that position. Otherwise, the landmark is kept in its initial position.

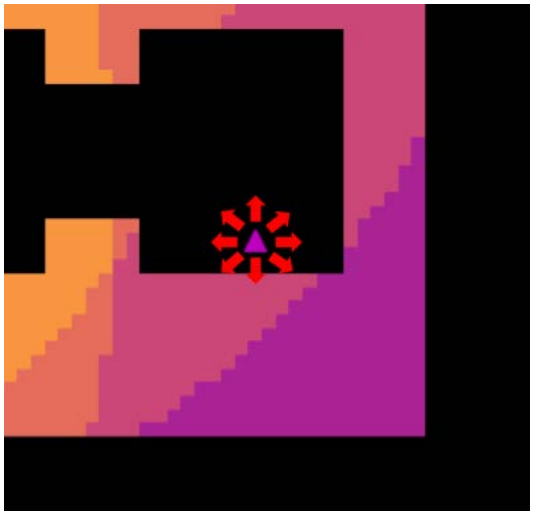


Fig. 4.16: Local optimization process.

At some points, the proposed system needs to add a new landmark to one of the sets. The steps where such a feature is required are the initialization process, the element mutation operation, and when a new iteration of the outer loop occurs. For this purpose, a greedy algorithm to incrementally add new landmarks based on the coverage fitness has been designed. For a given set of landmarks, this algorithm will find the new landmark position that would maximize the fitness of the coverage map. In order to find the cells with higher coverage gains, a score map is built, where each land cell has a score based on the number of free cells that are covered if a landmark is placed there. This map, presented in Figure 4.17, allows finding the position with the best coverage in a straightforward manner. However, if the best cell is always directly selected, the obtained solutions are more likely to converge to a local maximum because a fully-greedy approach will produce configurations with a low variety. Consequently, instead of selecting the cell with the best score, we select one of the cells with the highest scores; in order to make this auxiliary algorithm a bit more exploratory. This selection process is based on a score threshold, generating

a set of landmarks, $\mathcal{V}_{best}$, which contains all cells with a score higher than a given percentage of the maximum score. For example, if the maximum score is 1000, and the randomness threshold is set at 10%, then $\mathcal{V}_{best}$ will be composed by all cells with a score higher than 900. This process is described in equation 4.12, where $\mathcal{G}$ is the computed score map, and r is the randomness threshold that controls the amount of randomness allowed in the selection process.

$$\mathcal{V}_{best} = \{i, j\}, \quad \forall\{i, j\} \mid \mathcal{G}_{i,j} \geq (1 - r) \max \mathcal{G} \tag{4.12}$$

Finally, the final location for the new landmark is selected randomly from the $\mathcal{V}_{best}$ set. This selection mechanism also allows different levels of randomness or greediness by adjusting the parameter r, which determines the amount of exploration. Consequently, the different methods that use this auxiliary algorithm have different configurations for this parameter. For example, the initialization algorithm is configured to allow a high amount of randomness in order to generate a variety of initial solutions with more differences between them.

Last but not least, once the genetic algorithm is finally able to find a set of landmarks that satisfies the accuracy requirements, the obtained solution is simplified by a postprocessing method. This method consists of an ablation process that tries to remove unnecessary landmarks. Therefore, for each landmark in the proposed solution, this method will recompute the accuracy heatmap without that specific landmark and verify if the accuracy requirement is still satisfied. If the accuracy map is still good enough, the landmark is removed, and the cleaning process continues until there is nothing left to take away.

4.3 Experiments and results

The proposed approach (both the accuracy heatmap generation and the landmark placement optimization) has been validated using the ground truth from the simulation environment. To that end, two different scenarios have been designed, testing the implemented algorithms with different configurations. The validation approach is to execute the landmark placement optimization algorithm for each of the experimental scenarios and use the generated sets of landmarks to localize a moving vehicle. Then we verify that the obtained localization has an accuracy within the required limits. In this Section, the experimental setup and the configuration of

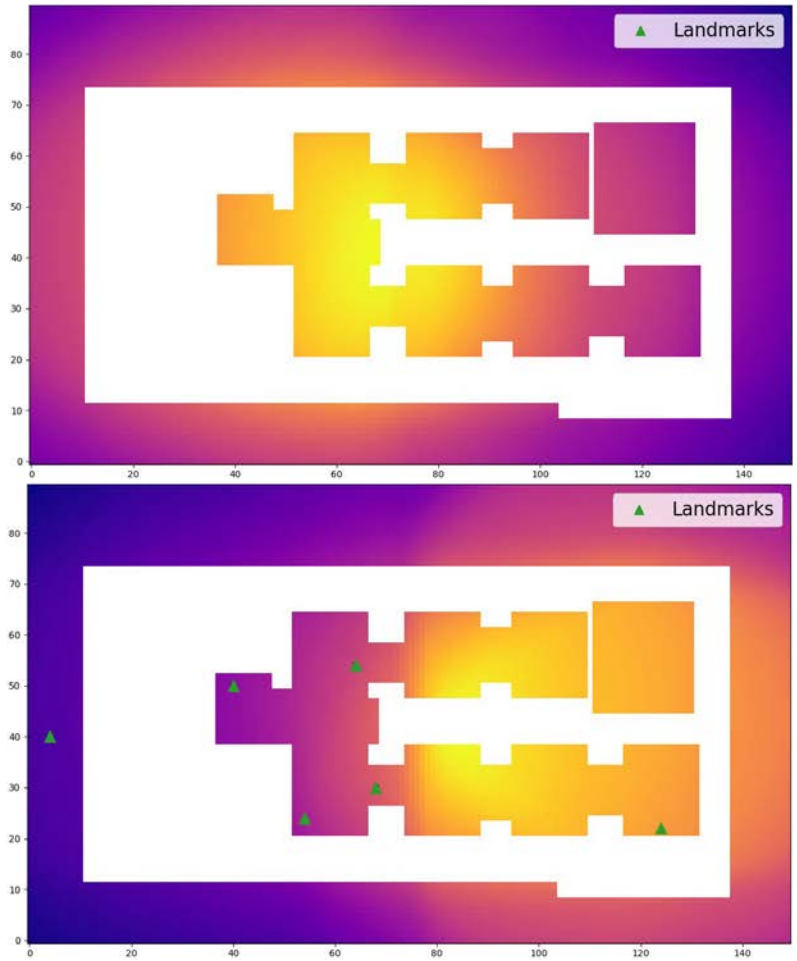


Fig. 4.17: Score maps for the auxiliary greedy algorithm. Brighter cells (more orange) have a higher score. The top map is empty; hence the score is higher towards the center with cells with higher coverage. The bottom map already has some landmarks, so the score is higher in the right area, which is less populated.

both scenarios are first described. Then, the experimental results from the simulated environments are presented and discussed.

4.3.1 Experimental setup

The experimental setup is divided into two validation stages that feature two simulation scenarios with different test environments and configuration parameters.

The first scenario has been designed to be simple. The testing area, depicted in Figure 4.18, does not have any type of elements that could create occlusions. The map is formed by a drivable area with asphalt texture and a semi-elevated area (in white) where landmarks can be placed, with a total area of $150 \times 90m$.

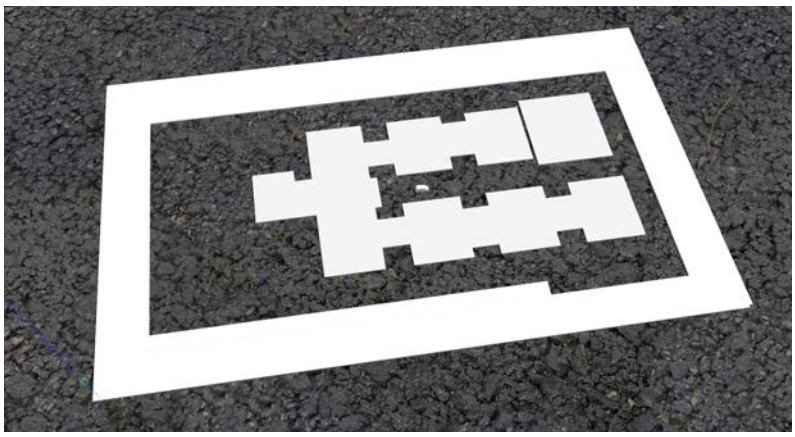


Fig. 4.18: Gazebo environment for scenario 1.

The second scenario, which is the environment designed for the second validation stage, has a more realistic setup and includes occluding elements such as buildings. This map has a total size of $80 \times 50m$ and makes use of the three types of space, as seen in Figure 4.19. The gray inner area is the drivable space, the white area is where landmarks can be placed, and the buildings are neutral spaces, where the vehicle cannot drive and also landmarks cannot be placed.

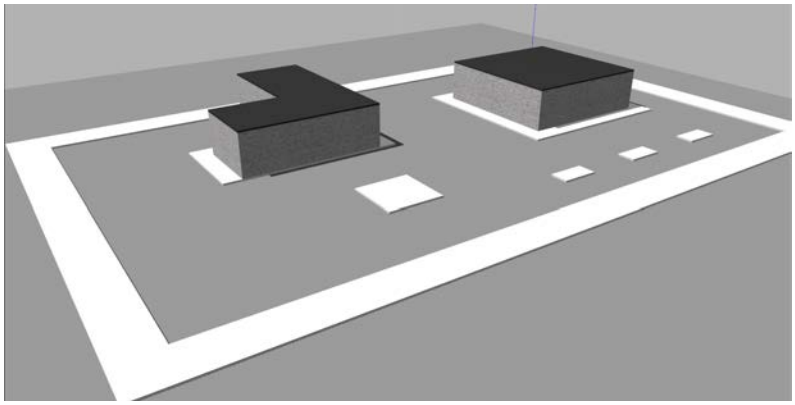


Fig. 4.19: Gazebo environment for scenario 2.

Afterward, the gazebo world model is transformed into an occupancy grid map representing the three types of areas. As an example, the processed occupancy grid map for the second scenario is presented in Figure 4.20, with free cells as white, land cells as gray and neutral cells as black. This occupancy grid map is the one used as the input for the optimization algorithm to estimate accuracy and coverage and to find the landmark positions.

Regarding the landmark measurement and visibility models, these setups are also different in both validation stages. On the one hand, the first validation stage

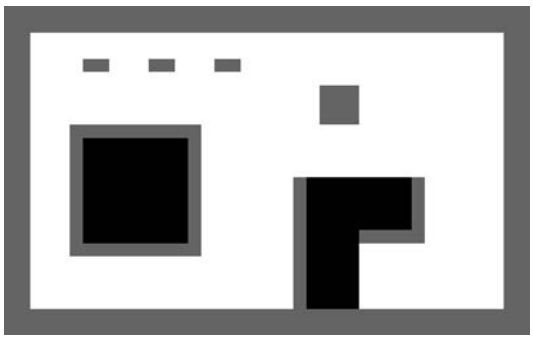


Fig. 4.20: Occupancy grid map for scenario 2.

was carried out before performing the measurement calibration analysis presented in Section 4.2.3, so a handcrafted mode was used instead. In order to cover for the inaccurate model, the standard deviation of the noise was inflated, resulting in a very pessimistic model. Consequently, the virtual landmark detections were assigned an error with a standard deviation between $5cm$ and $20cm$, increasing with the distance to the landmark. Additionally, because of the simplicity of this first validation stage, the sensor visibility model only accounted for the maximum range ($30m$), but occlusions from other landmarks were not considered. On the other hand, the second validation stage is more focused on making the simulation as close to reality as possible. Accordingly, the complete study presented in Section 4.2.3 is used as the landmark measurement model, with a more accurate standard deviation that ranges from $2mm$ to $2cm$, as shown in Figure 4.10. Furthermore, the visibility model is properly implemented in this stage, taking into account occlusions created by buildings and by other landmarks, as it is visible in Figure 4.22.

Once the grid maps are generated, the landmark placement algorithm is executed for both scenarios, resulting in two landmark configurations that are (presumably) guaranteed to satisfy inequation 4.1. In the first scenario, the parameter σ_{max}, which limits the maximum standard deviation of the translation error across the whole map, is set to $5cm$. With this parameter and the measurement model defined before, the landmark placement optimization algorithm is able to find a suitable set of landmarks with 27 landmark locations. Similarly, the second scenario is configured with a maximum error of $\sigma_{max} = 1.5cm$. In this second stage, it is possible to reach a lower localization error because the measurement model has a lower standard deviation compared to the pessimistic approach used in the first scenario. Furthermore, the landmark configuration generated by the placement optimization algorithm is composed of 14 landmarks.

Finally, to complete the simulation environment for the validation experiments, the calculated landmark positions are used to spawn new lamp posts in the gazebo

scenario. Figure 4.21 provides a global picture of the validation environment, including the testing area, the generated landmarks, and the vehicle. The final accuracy heatmap from the second scenario is also presented in Figure 4.22, next to the corresponding coverage map.

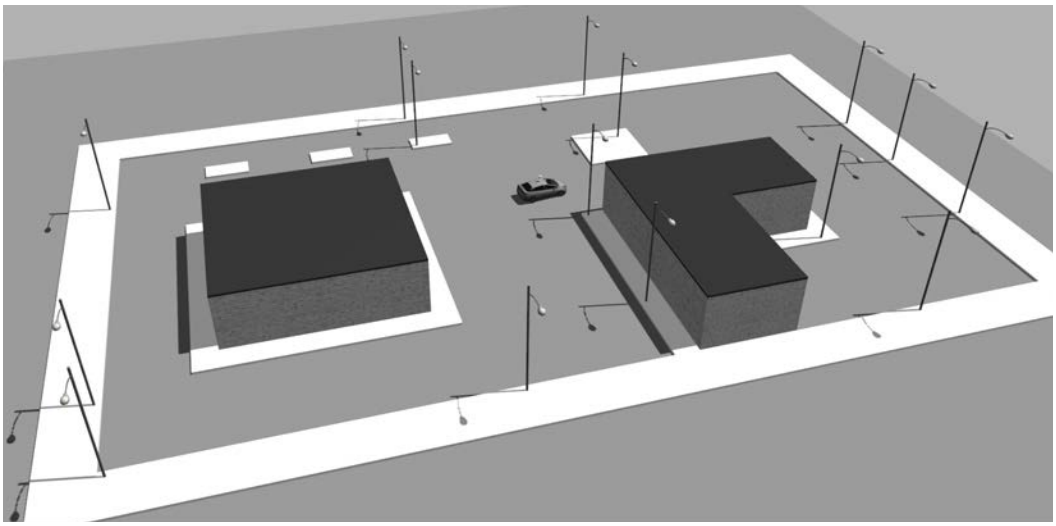
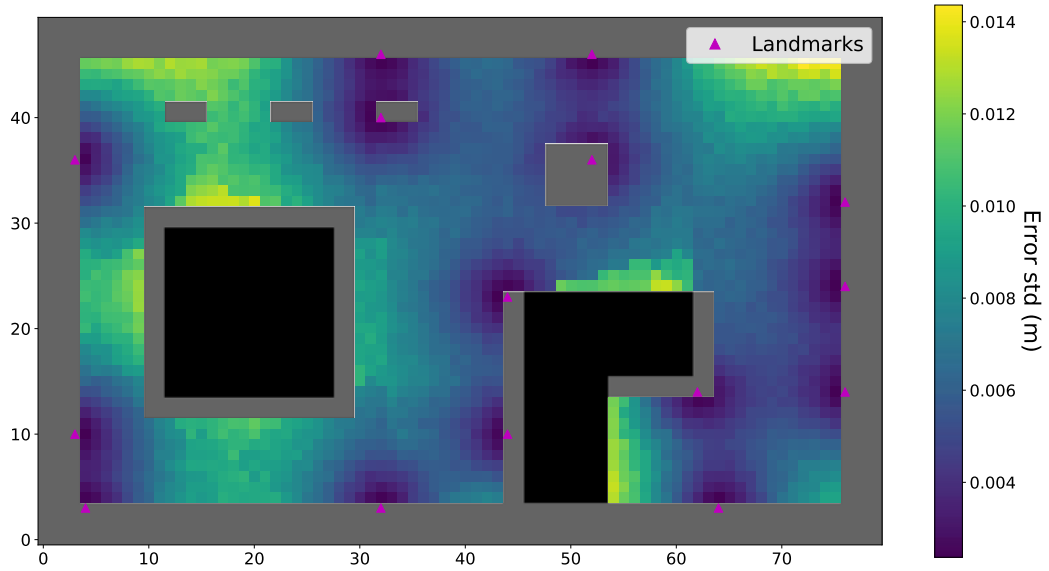


Fig. 4.21: Gazebo environment used in the validation experiments of scenario 2.

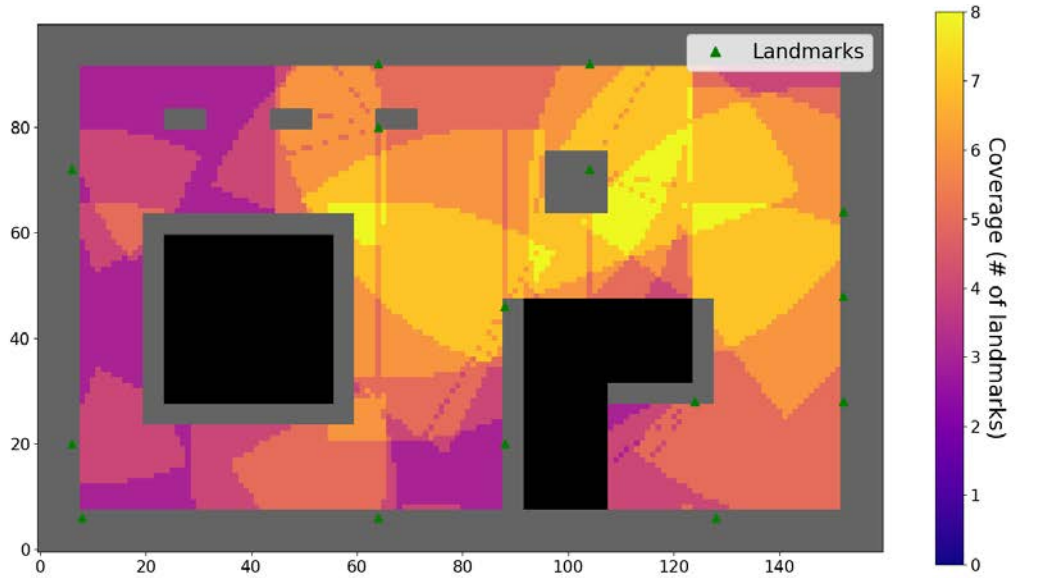
After completing the setup of the validation environment, a vehicle is integrated into the simulation environment, including a LiDAR sensor. The simulation ground truth is used to reference the true pose of the vehicle at every moment, and the proposed approach is validated by comparing the localization estimate obtained from the landmark-based localization method proposed in 4 with the simulator reference. The simulated vehicle is then teleoperated through ROS using an external joystick and driven around the testing area.

4.3.2 Results

The validation runs carried out for the first stage are presented in Figure 4.23, showing the three different trajectories followed by the vehicle and the landmark set provided by the placement optimization algorithm. Accordingly, the landmark positions and validation trajectories used in the second validation stage are presented in Figure 4.24. The driving of the vehicle was performed manually by teleoperation, and the performed trajectories were focused on covering as much of the testing area as possible. Furthermore, the validation trajectories were also conducted in a way that the most critical points were covered. These points, which have a higher localization error in the heatmap, generally include corners and areas that are far from the landmarks placed, such as map borders.



(a) Accuracy map.



(b) Coverage map.

Fig. 4.22: Accuracy and coverage maps obtained with the landmark configuration calculated for scenario 2.

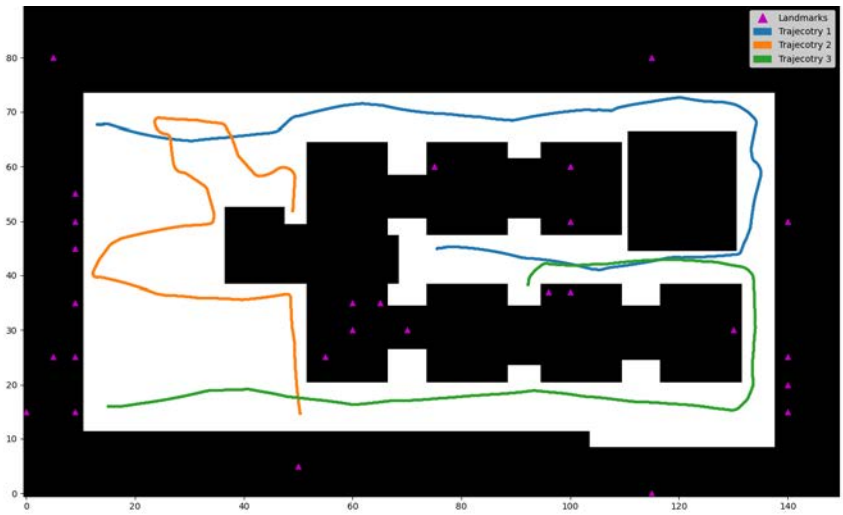


Fig. 4.23: Validation trajectories followed by the vehicle in scenario 1. Landmarks are displayed as magenta triangles.

Regarding the error metrics used to evaluate the proposed approach, the most suitable metric for this task is the APE metric. The reason is that the proposed localization method is designed to be used as a global reference localization system, so the error should be evaluated individually at each pose. Therefore, the absolute error between each pair of poses is computed using equation 4.13, where $P_{ref,t}$ is the reference pose from the simulation ground truth at timestamp t and $P_{est,t}$ is the pose estimated by the proposed landmark-based localization system.

$$E_t = P_{ref,t} \ominus P_{est,t} \tag{4.13}$$

Then, the translation and rotation components are separated to be analyzed individually, similarly to the approach followed in Section 3.2.3.

$$\begin{aligned} E_{trans} &= \|E\|_2 \\ E_{rot} &= \angle\,[E] \end{aligned} \tag{4.14}$$

These metrics are then applied to the estimated localization data, computing the mean and the standard deviation of the localization error. Additionally, the maximum translation error is also extracted for each one of the validation sequences. The maximum translation error is of special interest because it constitutes the main validation indicator for the proposed approach. If the proposed approach is valid, then the maximum translation error should not exceed σ_{max}. These values are presented in table 4.1 and table 4.2 for the translation and rotation error, respectively.

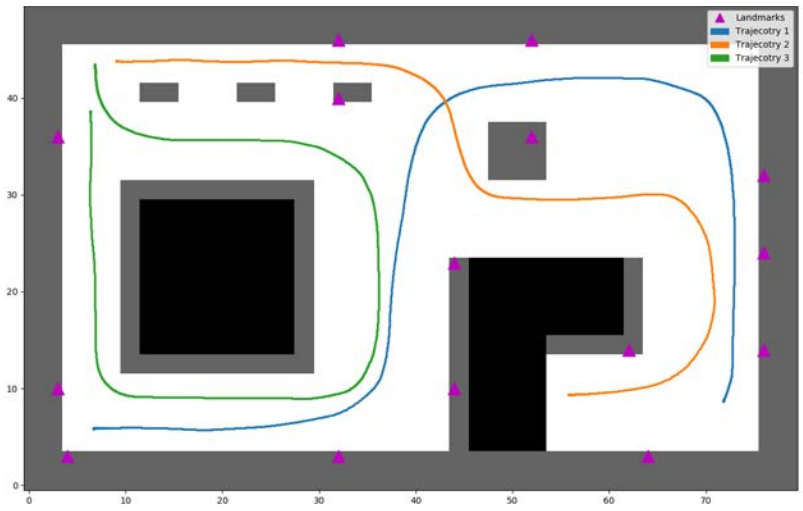


Fig. 4.24: Validation trajectories followed by the vehicle in scenario 2. Landmarks are displayed as magenta triangles.

		Mean (cm)	Std (cm)	Max (cm)	$\sigma_{\max}$ (cm)
Stage 1	Trajectory 1	1.404	0.779	4.922	5.0
	Trajectory 2	1.252	0.697	4.787	5.0
	Trajectory 3	1.227	0.649	4.309	5.0
Stage 2	Trajectory 1	0.230	0.140	1.354	1.5
	Trajectory 2	0.249	0.175	1.455	1.5
	Trajectory 3	0.289	0.185	1.499	1.5

Tab. 4.1: Translation error of each validation sequence.

As it can be appreciated in table 4.1, the translation error is kept under the upper bound σ_{max} at all times. This fact proves that the proposed landmark placement optimization approach is able to solve the presented problem, generating a set of landmarks that is suitable to keep the error under the desired bound. Furthermore, it is important to remark that although the maximum error is close to the parameter σ_{max}, the average error is several times lower. In fact, the translation error only gets closer to σ_{max} in some critical points in the testing area, resulting in a localization system that can provide a high accuracy consistently while guaranteeing an upper error bound. The quality of the localization solution is also ratified by the low values of the standard deviation, which indicate a stable estimation.

Regarding the rotation component of the localization error, while it is true that the optimization target is solely focused on the translation component to simplify the system, the results obtained for the rotation error are exceptionally good. Although the optimization method is not paying attention to the rotation error, this one is being reduced indirectly due to the nature of the NLLS estimator. The reason is that

		Mean (deg)	Std (deg)	Max (deg)
Stage 1	Trajectory 1	0.041	0.031	0.205
	Trajectory 2	0.040	0.030	0.169
	Trajectory 3	0.042	0.031	0.181
Stage 2	Trajectory 1	0.0009	0.0008	0.0035
	Trajectory 2	0.0012	0.0009	0.0043
	Trajectory 3	0.0058	0.0063	0.0411

Tab. 4.2: Rotation error of each validation sequence.

in the residual function, defined in equation 4.10, even a small error in the rotation component will result in a higher residual cost, as is also visible in Figure 4.12. Nevertheless, the accuracy target σ_{max} can also be easily adapted to include the rotation error or any other cost function by modifying the accuracy metric presented in Section 4.2.6 used in the heatmap generation.

4.4 Concluding remarks

In this chapter, a method aimed at controlling the localization error in controlled environments is proposed. On the one hand, it is possible to estimate the localization error produced by a landmark-based localization system. To that end, we propose a system that can use the landmark measurement model to compute a heatmap that represents the expected error at each point in the testing area. On the other hand, this accuracy heatmap can be exploited to implement a landmark placement optimization method that is able to find a set of landmarks that guarantees an upper bound in the localization error.

In order to have access to actual ground-truth data, the proposed approach has been validated using a simulation environment. Multiple experiments have been carried out, featuring different environments, sensor models, and configurations. The obtained results confirmed that the proposed approach is able to generate a localization reference with a constrained error while actually achieving a very high accuracy level.

The proposed method has been designed to build accurate and reliable localization systems in a controlled environment, such as proving grounds, where the environment can be modified in order to add new landmarks in the testing area. The obtained results show the feasibility of having a localization reference system based on landmarks using the proposed approach, with the reliability that provides the guaranteed upper error bound of the localization error.

However, it has only been possible to validate this approach in a simulation environment, as outdoor reference systems do not provide actual ground truth. This task remains open for future work, being an interesting topic that is still open to study.

Cooperative Localization

5

> *The more complex the world becomes, the more difficult it is to complete something without the cooperation with others.*
>
> **— Alexander Fleming**

The future of transportation systems is rapidly evolving towards autonomous driving technologies. Likewise, urban development techniques are beginning to integrate IoT technologies, evolving toward the so-called smart cities. With the development of new communication technologies, such as 5G, high-bandwidth, low-latency communication between the multiple agents involved in the driving scene becomes now a possibility.

This chapter revolves around the idea of exploiting the advances in communication and ITS technologies to improve the localization systems of vehicles. Localization systems usually rely on GNSS systems, which are known to behave badly in urban areas due to the reduced satellite visibility and the structured environment causing multipathing problems. Instead of proposing the use of specific sensors and technologies for an accurate localization system where GNSS systems fail, the work presented in this chapter advocates for the reuse of the perception systems already operating in autonomous vehicles.

5.1 Introduction

In this chapter, a cooperative localization system for autonomous vehicles is presented. This system requires a low-latency communication system that allows sharing of different data between the vehicles, such as the localization estimate or the detected obstacles. Therefore, the recent advances in technologies developed for smart cities are perfect to finally bring a powerful communication system for autonomous vehicles, enabling new forms of cooperative algorithms.

Urban environments are characterized by their complexity when performing autonomous driving tasks. Cities are very dynamic environments, and vehicles must be

able to detect multiple elements, such as other vehicles, infrastructure elements, and VRUs like pedestrians or cyclists. Consequently, great efforts have been carried on in recent research to improve perception systems in autonomous vehicles; in order to guarantee a safe operation.

Additionally, GNSS systems have been established as the defacto standard localization system for outdoor, autonomous systems due to their high availability and their relatively low cost. One of the problems of GNSS systems, however, is that despite the multiple low-cost solutions available in the market, accurate systems are generally very expensive. Furthermore, GNSS systems perform better in open spaces, where proper satellite coverage is available, and the connection with the satellites is not interrupted. This ideal is the opposite of what is found in urban environments, where tall buildings form urban canyons that limit satellite coverage, and the vision of the sky changes at each new intersection. The lack of signal coverage is even worse in closed spaces such as tunnels or covered parking lots, where the GNSS position estimation is usually completely lost.

While it is possible to design better localization systems for urban environments, they usually rely on additional sensors and the use of digital maps. High-definition digital maps, nonetheless, have limited availability and are costly to produce due to the precision level required. Moreover, such localization systems require an additional computational load in the vehicle processing system to extract features from the environment and match them with the map data.

Nevertheless, it should be possible to reuse the processed data from the perception systems, avoiding an unnecessary overprocessing of the sensors' data. Particularly, a system where vehicles are able to share their position estimation (from GNSS) and the list of detected objects (from the perception system) could combine that information to improve the initial localization estimation of each vehicle.

Several works have been presented in recent years proposing different approaches to deal with the problem of cooperative localization in autonomous vehicles. The objective in outdoor cooperative localization problems is always to enhance the performance of the GNSS-based system using the additional positioning information that is inferred from the relative transformation between vehicles. The relative transformation between vehicles can be acquired from onboard sensors and provides information about the distance or the full 3D transformation between one vehicle and another. Although these measurements also carry some uncertainty, it is usually much lower than the error from the GNSS estimation, thus being able to improve its accuracy.

The most common way to obtain this information is by measuring the distance between vehicles. In most cases, such as the one presented in [278], the same communication system used to share data between vehicles is used to estimate the distance between them by exploiting the signal's RSSI. This approach, however, has the inconvenience of being only available when the communication is performed directly between the vehicles, as it is done in cooperative systems based on Vehicular Ad-Hoc Network (VANET) technologies. Moreover, relying on distance measurements to enhance the GNSS estimation does not provide as much information as the detection of a vehicle from a perception system, which includes 3D coordinates and even the orientation of the target.

Regarding the estimation output, most cooperative localization methods present an egoistic approach by focusing only on estimating the pose of the ego vehicle. This decision simplifies the algorithm implementation because the number of free variables to optimize is greatly reduced. However, an estimation algorithm focused only on the ego vehicle has more restrictions and is more sensitive to the bias of the GNSS estimation obtained from the other vehicles. Some works, such as [260], cope with this problem by implementing a mechanism to select the best vehicle poses as anchors; in order to obtain a better estimation of the ego vehicle's pose. Still, this approach relies on having enough vehicles nearby with high-quality GNSS systems, which in practice is not a realistic scenario.

Accordingly, an enhanced localization system through sensor fusion and data sharing for autonomous vehicles is proposed in this chapter. This cooperative localization system relies on vehicles sharing their initial localization estimation and their lists of detected vehicles. The objective is to design a system that is able to improve the already existing localization systems (based on GNSS) with the help of vehicle detections and the combination of the data shared by multiple vehicles. Furthermore, instead of centering the focus on the pose of the ego vehicle, the proposed system is designed to optimize the pose of all vehicles, improving the localization performance globally. Such a localization system requires a low-latency communication system that vehicles can use to share the processed data. Fortunately, the raw sensor data is not required; hence the size of the shared information is relatively small, making efficient use of the communication channel and avoiding the necessity of high bandwidth.

5.2 Proposed approach

In this section, the proposed solution to the cooperative localization problem is described.

5.2.1 Problem statement

The presented problem is formulated as it is usually done in the literature, but with the difference that instead of estimating the ego vehicle's pose, the system must estimate all poses simultaneously. Despite increasing the complexity of the method because of the extra degrees of freedom, we believe this approach can result in better results than the egoistic one. The reason is that common methods generally optimize the pose of the ego vehicle while assuming that the rest of the poses are fixed. Therefore, localization bias has a strong contribution to the estimation and can lead to converging to wrong estimates. On the other hand, when all poses are optimized concurrently, the localization system has more flexibility, creating new opportunities to mitigate the bias in the optimization process. The proposed approach is exemplified in Figure 5.1, including all the information sources used in the data fusion mechanism.

Due to the numerous variables to optimize, and in order to deal with the nonlinearities associated with the orientation of the vehicles, we propose a Genetic Particle Filter (GPF) to solve the cooperative localization problem. This type of Particle Filter shares most of its process with a generic Bootstrap Particle Filter, with the particular exception that the resample process is performed by a Genetic Algorithm. The proposed estimation algorithm has been implemented in a cooperative localization framework based on ROS2 named *LoCo*, which was specifically designed for this task.

The inputs and outputs of the proposed cooperative localization method are summarized in Figure 5.2. This method has several inputs that are required from each of the vehicles in the system:

- **GNSS prior estimation**: The initial (noisy) pose estimation provided by the GNSS system.
- **Vehicle detections**: The list of vehicle detections extracted from the perception system.

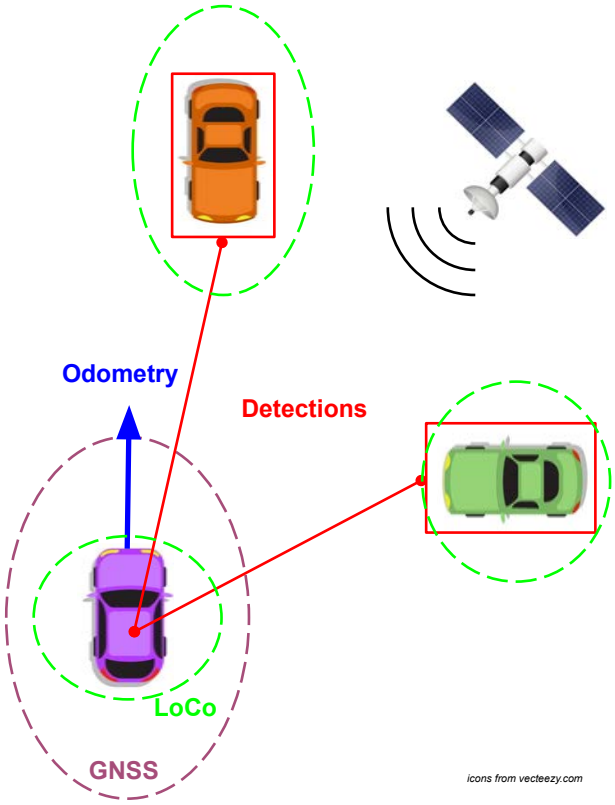


Fig. 5.1: Proposed cooperative localization approach.

- **Local odometry**: Odometry source obtained from the vehicle's wheel odometer or a different odometry algorithm. This input is optional but can be used to improve the prediction step.

5.2.2 LoCo framework

The proposed cooperative localization method is based on a Genetic Particle Filter, but the whole approach requires a bigger framework to integrate the data from the multiple vehicles that form the cooperating system. For this reason, we have designed and developed the *LoCo* framework [1], which is a software suite implemented in ROS2 that handles the received data and prepares it for the estimation process.

This framework is based on ROS2 for two reasons. On the one hand, this second version of the Robot Operating System has a communication system based on Data Distribution Service (DDS) protocol to share messages between the different

[1] LoCo framework is open source software available at `https://github.com/butakus/loco`

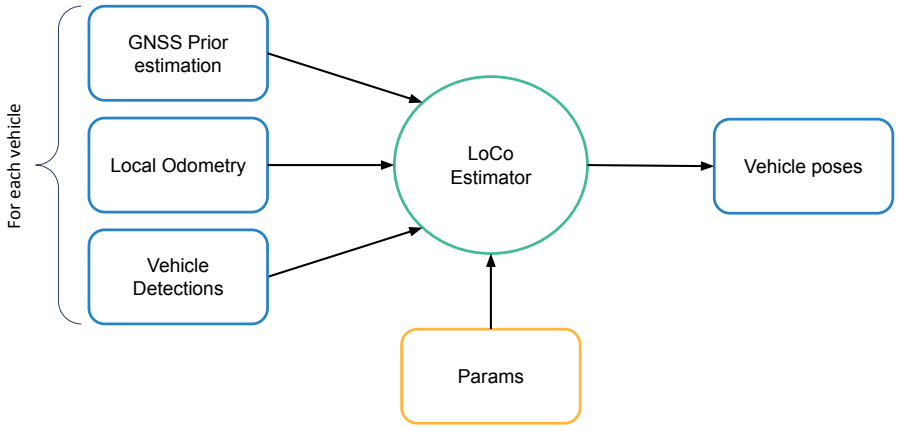


Fig. 5.2: Inputs and outputs of the proposed cooperative localization method.

process nodes. The use of a standard protocol such as DDS as the communication middleware provides multiple advantages to cooperating systems compared with the previous ROS version, especially for systems based on wireless communications. On the other hand, ROS2 provides multiple features that simplify software integration in both simulation environments and real-world platforms.

One of the purposes of the *LoCo* framework is to handle the input data. This step consists in synchronizing the received messages, associating the received measurements with each of the vehicle agents, and providing the appropriate data to the estimation algorithm. Furthermore, the core estimation process is implemented as an independent module, so the *LoCo* framework can be used to compare different cooperative localization algorithms by exchanging the core estimation module (in our case, the Genetic Particle Filter). Finally, the implemented framework is able to handle 2D and 3D data in their $\mathbf{SE}(2)$ and $\mathbf{SE}(3)$ forms, respectively. Nevertheless, the work presented in this chapter has been represented only in 2D for the sake of simplification.

5.2.2.1 Data synchronization

The cooperative localization process is executed in a centralized way, where every vehicle receives all the required data and runs the estimation algorithm by itself. In a later stage, the vehicle poses estimated by each of the vehicles could be shared to keep the best estimation, but that kind of post-processing is not considered in the proposed approach.

Before processing the shared data, it is important that the information shared between the vehicles is time stamped and that all agents share the same clock. This clock synchronization process can be carried out at the system level by a Network Time Protocol (NTP) process, which would provide enough precision for the problem at hand. If necessary, better clock synchronization approaches are possible within 5G communication schemes, as presented in [279].

Even if the system clock of each vehicle is synchronized, the data shared from each vehicle is not acquired at the same exact moment, so an additional message synchronization process is carried out after receiving the data from all vehicles. The *LoCo* framework is implemented in multiple threads, isolating data acquisition, estimation, and output generation in concurrent processes. This way, the data acquisition is not blocked by the estimation process, which takes most of the computation time.

When the estimation process starts, the system takes a snapshot of the latest data received from each vehicle. If the time difference between a message and the current time is too high, the data is interpolated using the current odometry data for the given vehicle. In normal operating conditions, the data is received at 10Hz, and the delay is only about a few milliseconds, so this interpolation does not provide a significant benefit.

5.2.2.2 Data association

In addition to synchronizing the input data, the received vehicle measurements must also be processed before being ready for the estimation process. The reason is that the vehicle detections from the perception system do not provide an identification of which vehicle was detected at each position. Moreover, not all vehicles can be detected from every vehicle, so the list of detections provided by each vehicle will likely not contain the whole set of vehicles participating in the cooperating system.

The vehicle measurement data is sent to the estimation algorithm as a detection matrix $\mathbf{M}_{V\times V}$, where V is the number of vehicles in the system and $\mathbf{M}_{i,j}$ represents the vehicle detection from vehicle i to vehicle j. An additional matrix $\mathbf{M}^*_{V\times V}$ is also provided, where $\mathbf{M}^*_{i,j}$ indicates if the detection from i to j exists and is valid. Each detection is represented as a $\mathbf{SE}(2)$ pose $\mathbf{z}_{i,j}$ with an associated 3×3 covariance matrix $\mathbf{\Sigma}_{i,j}$. Finally, the list of detections received from each vehicle is defined as $\mathbf{Z}_{1\times V'}$, with $V' \leq V$.

In order to determine which vehicle corresponds to each one of the measurements contained in $\mathbf{Z}_i$, an unbalanced assignment problem must be solved. Given a set of

vehicles V, which constitutes all the vehicles in the system, and a list of measurements $\mathbf{Z}_i$, the problem consists in finding which measurement $\mathbf{z}_{i,j}$ corresponds to which vehicle in V. The optimal assignment is obtained by the Hungarian method [280], which makes use of a cost matrix $\mathcal{C}_{j,v}$ to determine the best assignment of measurements. Each element of the cost matrix represents the cost of assigning the measurement $\mathbf{z}_{i,j}$ to the vehicle $v \in V$. This cost is based on the difference between the previously estimated pose of vehicle v, $\mathbf{x}_v$ and its virtual pose $\hat{\mathbf{x}}_v^j$, which is calculated according to equation 5.1 and represents the pose that vehicle v would have if the measurement $\mathbf{z}_{i,j}$ was assigned to it.

$$\hat{\mathbf{x}}_v^j = \mathbf{x}_i \oplus \mathbf{z}_{i,j} \tag{5.1}$$

Afterward, the assignment error is defined as the difference between the pose $\mathbf{x}_j$ and the virtual pose $\hat{\mathbf{x}}_j$:

$$\mathbf{E}_{j,v} = \mathbf{x}_j \ominus \hat{\mathbf{x}}_v^j \tag{5.2}$$

Using this assignment error in $\mathbf{SE}(2)$, different alternatives are available when calculating the assignment cost. In the *LoCo* framework, the following three assignment cost metrics are implemented:

- Translation error:

$$\mathcal{C}_{j,v} = \|\mathbf{E}_{j,v}\|_2$$

- Weighted translation plus rotation errors:

$$\mathcal{C}_{j,v} = \omega_t \|\mathbf{E}_{j,v}\|_2 + \omega_r \angle\left[\mathbf{E}_{j,v}\right]$$

- Frobenius norm of the $\mathbf{SE}(2)$ error matrix:

$$\mathcal{C}_{j,v} = \|\mathbf{E}_{j,v}\|_F$$

5.2.3 Particle Filter

With all the inputs synchronized and properly prepared, the pose estimation is performed by a Particle Filter, which is a localization method that has shown good performance in other similar state-of-the-art approaches such as [261]. The proposed Particle filter, however, has gone through several modifications from the generic Bootstrap Particle Filter that is used in Monte Carlo localization methods. On the one

hand, the filter has been adapted to handle a state composed of multiple vehicles. On the other hand, the designed Particle Filter implements genetic operators in the resampling step, making this estimation algorithm a hybrid between a Particle Filter and a Genetic Algorithm.

The benefits of this kind of hybrid Genetic Particle Filter have been studied before for robot localization [281] and target tracking [282], leading to the conclusion that a genetic resample is able to deal with the problem of sample impoverishment better than other typical resample approaches. In addition, these two algorithms are easy to combine because they have multiple similarities, sharing different elements:

- The set of particles in a Particle Filter is equivalent to the population of a Genetic Algorithm. Therefore, a common data encoding can be used.
- The particle weights in a Particle Filter can be seen as a representation of the population fitness in a Genetic Algorithm.
- The likelihood function to compute the weight of each particle has its counterpart in the Genetic Algorithm in the fitness function.
- The resample step in a Particle Filter generates a new set of particles from the old set based on their weights. Similarly, each iteration in a Genetic Algorithm also generates a new population, based on the previous elements and their fitness.

In this work, the localization problem involves a state composed of the poses from multiple vehicles. Consequently, the state encoding into the filter population (or particles) $\mathcal{P}$, which is illustrated in Figure 5.3 is done as follows: The whole population set is a list of N elements, where each element constitutes a potential solution to the localization problem. Furthermore, each element is composed of the pose of each of the vehicles in the cooperating system, forming a list of V elements. The vehicle poses are represented as transformation matrices in a $\mathbf{SE}(2)/\mathbf{SE}(3)$ form, where each vehicle pose might be considered as a chromosome from that element.

With this state configuration, the Particle Filter process is summarized in Algorithm 2. Nevertheless, the resampling step is just defined as a black box because, in addition to the proposed genetic algorithm, it can also be implemented as one of the typical resampling methods, such as the Sequential Importance Resampling (SIR). This will be used in Section 5.3 to compare the proposed genetic resample method with other resampling alternatives. The proposed genetic resample method, nonetheless, is described in Section 5.2.4.

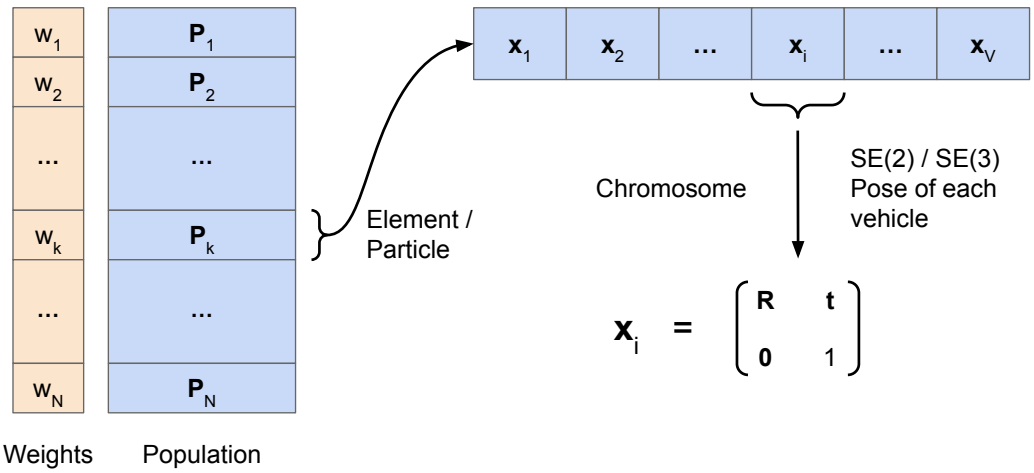


Fig. 5.3: Population state encoding for the Genetic Particle Filter.

This process can be divided into five parts, which are marked as comments in Algorithm 2. The first step in the filter is to initialize the population. This initialization process is only executed when the algorithm starts and makes use of the initial measurements from the GNSS system. The GNSS measurements form the prior estimation $\mathbf{x}_p$, where the GNSS measurement from each vehicle, $\mathbf{x}_{p,i}$, is used to initialize the corresponding chromosome on each element. In order to add dispersion to the initial population, an extra noise is added to each of the initial poses according to the covariance of the prior estimation.

Afterward, the remaining steps are executed in cyclic iterations, which are executed whenever new data is ready or at a fixed frequency. The second step is the prediction step, also known as the motion update step. In this process, the particles are propagated according to the vehicle motion model, which is based on the local odometry measurements encoded in $\mathbf{u}_i$.

Then, the particle weights are updated in step number three. This process is often called the sensor update because the filter state is updated with the information received from the sensors. In the proposed filter, the weights are calculated according to the detections received from each vehicle, which are combined into a detection matrix $\mathbf{Z}_{V\times V}$, as defined previously.

Parameter: $V \equiv$ Number of vehicles
Parameter: $P \equiv$ Number of particles
`/* 1. Initialize population */`
$\mathcal{P} = InitializePopulation(\mathbf{x}_p)$
while *True* **do**
  `/* 2. Prediction */`
  for $k = 1$ *to* P **do**
    for $i = 1$ *to* V **do**
      $\mathbf{x}_{k,i} = MotionUpdate(\mathbf{x}_{k,i}, \mathbf{u}_i)$
    end
  end
  `/* 3. Compute likelihood and update particle weights */`
  for $k = 1$ *to* P **do**
    $\omega_k = L(\mathbf{x}_k, \mathbf{Z})$
  end
  `/* Normalize weights */`
  for $k = 1$ *to* P **do**
    $\omega_k = \frac{\omega_k}{\sum_{k=1}^{P} \omega_k}$
  end
  `/* 4. Update estimation (mean and covariance of each vehicle pose) */`
  for $i = 1$ *to* V **do**
    $\mathbf{x}_i^* = \sum_{k=1}^{P} \omega_k \mathbf{x}_{k,i}$
    $\mathbf{\Sigma}_i = \frac{\sum_{k=1}^{P} \omega_k (\mathbf{x}_{k,i} - \mathbf{x}_i^*)^T (\mathbf{x}_{k,i} - \mathbf{x}_i^*)}{1 - \sum_{k=1}^{P} \omega_k^2}$
  end
  `/* 5. Resample particles */`
  $\mathcal{P} = Resample(\mathcal{P})$
end

Algorithm 2: Particle Filter process

The proposed likelihood function $L(\mathbf{x}_k, \mathbf{Z})$ serves as well as the fitness function for the current population, and it is defined in equation 5.3.

$$\begin{aligned} \omega_k = L(\mathbf{x}_k, \mathbf{Z}) &= \sum_{i=1}^{V} \sum_{j=1}^{V} l(\mathbf{x}_{k,i}, \mathbf{x}_{k,j}, \mathbf{Z}_{i,j}), \quad \text{if } \mathbf{Z}_{i,j} \neq \emptyset \\ l(\mathbf{x}_{k,i}, \mathbf{x}_{k,j}, \mathbf{Z}_{i,j}) &= \frac{\exp\left(-\frac{\mathbf{z}_e^T \Sigma_s \mathbf{z}_e}{2}\right)}{2\pi\sqrt{\det \Sigma_s}} \end{aligned} \tag{5.3}$$

Where $\mathbf{z}_e$ is the measurement residual error reduced to its translation component, defined as $\mathbf{z}_e = \|\mathbf{z}_{i,j} \ominus (\mathbf{x}_j \ominus \mathbf{x}_i)\|_2$. Moreover, the covariance matrix Σ_s is a scaled version of the measurement covariance provided by the perception system. This scaling is required when the perception system is very accurate and provides a very

small covariance. In these cases, most of the resulting weights will be very close to zero. Consequently, a positive scaling of such small covariances results in keeping the weights meaningful and avoiding particle depletion. Finally, the weights are normalized, so their sum is equal to 1.

Once the weights are calculated, it is possible to extract the pose of each vehicle from the set of particles. This process, presented in point number four, estimates the mean and the covariance of each one of the vehicle poses. First, the sample points from each particle are separated and grouped by vehicle. Then, the set of sample poses for each vehicle, $\mathbf{x}_{k,i},\ \forall k$, is used to extract the mean and the covariance of the data using the given equations. Note that the samples are weighted according to the particle weights when the mean and covariance are computed.

The next and final step is to resample the current set of particles and generate a new one. In the presented literature, the resampling step is implemented following a Sample Importance Approach, where the particles are randomly sampled according to their weights, and the normalized weights represent the probability of being drawn. In the proposed approach, however, this resampling process is carried out by a genetic algorithm, which is presented in the next section.

5.2.4 Genetic resampling

The genetic resampling process makes use of the particle weights as the population fitness and exploits the typical genetic operators to generate a new generation of particles after each iteration. Every iteration in the Particle Filter is followed by one iteration in the Genetic Algorithm, making the population grow and converge to the actual vehicle poses.

The genetic process is composed of three steps, applying three genetic operators: selection, crossover, and mutation.

5.2.4.1 Selection

The selection process extracts a subset of the particle set based on their weights. The set of parent particles, or the selection winners, has a fixed size based on a percentage of the total size, defined by a parameter w_p. Consequently, the number of selected particles is defined as $W = w_p P$.

Adjusting this parameter affects the behavior of the estimation algorithm because, with bigger values of w_p, the new set of particles will be very similar to the previous one, limiting the exploration provided by the crossover operator. On the other hand, a small value of w_p will result in having a new generation based on new particles, while most of the previous particles will be lost.

This selection process is controlled by a tournament process, where each one of the W particles is selected from a small subset of the population. The tournament size is defined by the parameter t_p, which is a percentage of the total population. Accordingly, the total number of particles that go to each tournament round is $T = t_p P$.

Smaller values of t_p produce tournament rounds with a smaller number of particle contestants, so there is a higher chance of getting a parent whose fitness is not among the best. This, however, promotes exploration by not always sticking to the W best particles, so this parameter must be adjusted in a balanced way.

5.2.4.2 Crossover

After selecting the subset of parent particles, the remainder of the population is filled with the offspring, which is created by combining chromosomes from the parents in a crossover process. This crossover process, which is exemplified in Figure 5.4, is performed by pairs of parents. For each new offspring particle, two random parent particles are selected. Then, each one of the chromosomes that form the offspring particle is chosen randomly from one of the two selected parents, with a chromosome representing a vehicle pose.

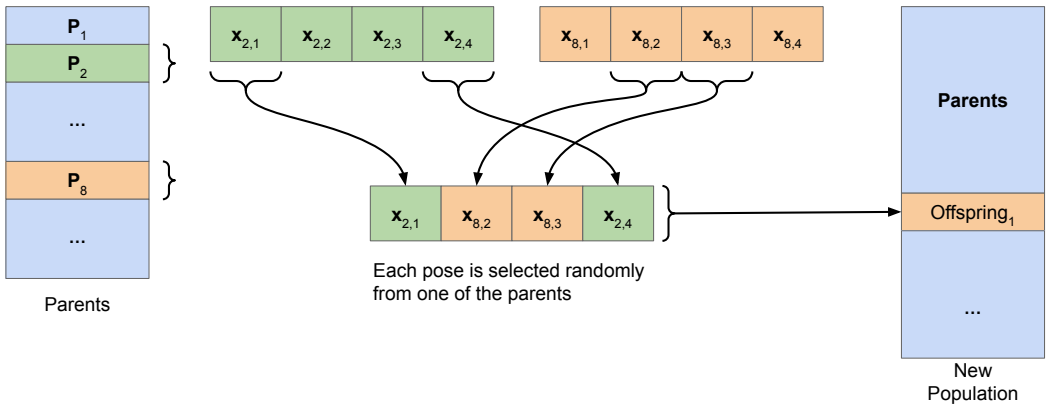


Fig. 5.4: Genetic crossover example.

This mating crossover process combines the vehicle poses from different parents with good fitness. The motivation behind this type of crossover is that one particle from the subset of parents might have good fitness because most of the vehicle poses are estimated correctly, but other poses might be wrong. By combining the poses from different parents, it is possible to obtain a new particle with better vehicle poses overall.

5.2.4.3 Mutation

The last of the genetic operators applied is a mutation process that only affects the new offspring elements when they are being created. As opposed to the previous operators, this mutation process is based on a novel approach that integrates the prior estimation from the GNSS measurements into the resampling process.

The mutation might affect individual chromosomes when generating a new offspring particle, as illustrated in Figure 5.5. Furthermore, the probability of a chromosome being affected by a mutation is defined by the parameter m_p.

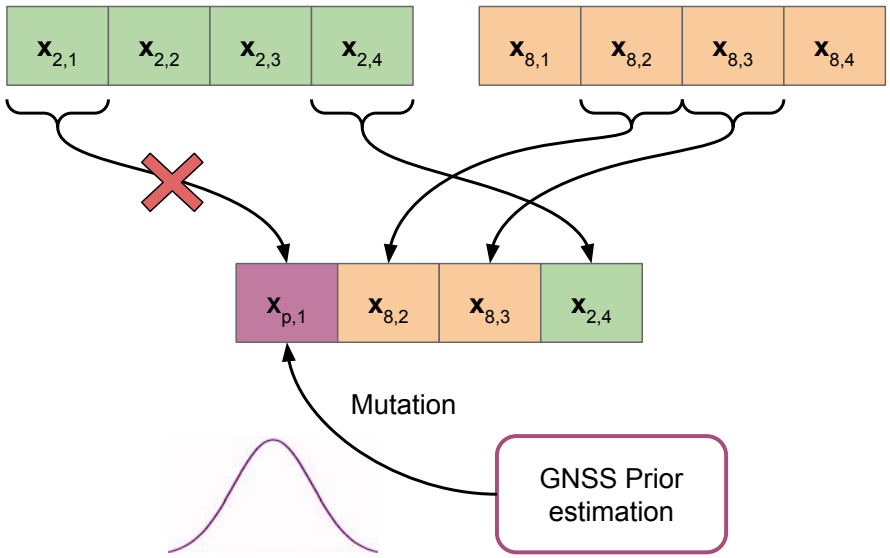


Fig. 5.5: Genetic mutation example.

When a chromosome is mutated, the corresponding vehicle pose is not copied from either of the parents. Instead, a new pose is drawn from the prior estimation obtained from the latest GNSS measurement, similarly to how it is done in the initialization process.

This type of mutation has the ability to destabilize the filter if it converges to a biased estimation that might happen if all vehicle poses are transformed by the same offset. In that case, the vehicle measurements would result in a high likelihood, even though the poses are wrong. If the filter was to converge into such a state, then this mutation mechanism can make some particles jump closer to the actual pose, providing some global consistency. This global consistency cannot be guaranteed otherwise because the GNSS measurements are only used in the initialization step, and the rest of the time, the filter only uses detection and odometry measurements, which are local to the vehicles. Furthermore, this mutation mechanism also prevents the kidnapped robot problem, which is a common problem in Monte Carlo Localization methods.

Consequently, this process makes mutation one important tool in the proposed estimation algorithm. However, the obtained effect is very dependent on the selected value for the mutation rate m_p. If the mutation rate is very high, the vehicle poses created for the offspring particles will be reset to the GNSS distribution, considerably reducing the effect of the filter. On the other hand, a low mutation rate might not have enough power to change the population in a way that the previously mentioned problems can be avoided.

Furthermore, the contribution of the mutation rate m_p is highly tied to the parameter w_p, which controls the number of parents and the size of the offspring. The reason is that the mutation process only affects particles from the offspring subset, so the effect of each one of these parameters is also affected by the other. Consequently, the parameter-tunning process must consider this, thus jointly adjusting both parameters.

5.3 Experimental results

The performed experiments are directed to three different objectives. The first series of tests are destined to find a proper configuration of the filter parameters that require a tuning process, as previously described. Then, the genetic resampling approach is validated by comparing its performance against another resampling method used typically used with Particle Filters. Finally, the overall performance of the proposed approach is evaluated.

In this section, the experimental setup is first described. Then, the different tests and scenarios used are detailed, followed by the presentation and discussion of their corresponding results.

5.3.1 Simulation environments

One of the most common problems that arise when developing cooperative algorithms is the lack of resources to perform proper experiments with real-world platforms. This case is especially true for cooperative autonomous vehicles due to their high cost. For this reason, the majority of the works presented in the literature are based on simulation environments. Likewise, all the experiments carried out in this work have been implemented in simulation environments, making use of the simulation ground truth and overcoming the lack of enough physical resources.

In order to test and validate the proposed approach, two different simulation environments are introduced. On the one hand, we introduce a simplistic agent emulator that has been designed for the testing of cooperative algorithms in simple and controlled environments without any complex physics simulation. On the other hand, we use Carla simulator [274] to carry out a proper simulation of the environment, obtaining a simulation of the vehicles closer to the real world and thus obtaining more realistic results.

5.3.1.1 Agent emulator

The *agent emulator* is a software tool that has been specifically designed to perform simple tests on cooperative algorithms implemented in ROS or ROS2. This tool provides multiple mechanisms to emulate moving agents, which are represented as poses without volume. In order to keep it as simple as possible, only the movement is simulated without accounting for collisions or friction physics.

Despite being based on a simple simulation, this tool is very useful in the early stages of the development of an algorithm, allowing one to quickly perform simple tests with multiple vehicles without requiring a powerful computer. Additionally, one of the main advantages of this simulation approach is that the software modules are fully integrated with ROS and ROS2 ecosystems. Therefore, the implemented cooperative software can be easily integrated later on into a more powerful simulation environment or in actual real-world platforms.

The *agent emulator* suite does not provide a graphical interface to visualize and control the simulation. Instead, the state of the agents can be monitored by using ROS tools, such as *Rviz*. Figure 5.6 shows a snapshot of one of the tests with three agents, displaying the ground truth and localization data of one of the agents.

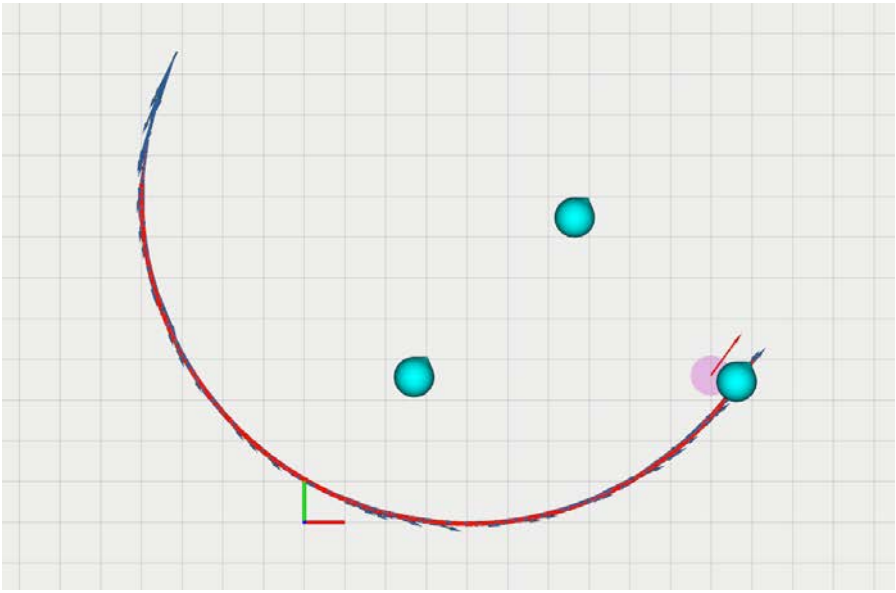


Fig. 5.6: *Agent emulator* environment.

In these simulations, the agents' movement can be controlled either by position setpoints or by sending velocity and/or acceleration commands. The kinematic model, however, is not based on the Ackermann model, which is the appropriate model for car-like vehicles. Since the agents are just considered points in space with an orientation, their movement is calculated by modeling their linear and angular accelerations independently. To ease the setup of testing scenarios, several velocity and acceleration controllers are included in the software suite.

Regarding the sensors implemented in the *agent emulator*, the ROS node that controls each agent provides the following data:

- **Global localization**: The global localization output imitates the behavior of a GNSS system, providing an estimation of the global pose of an agent with an added Gaussian noise of adjustable covariance. The covariance can be modified in runtime, to simulate the effect of urban canyons or tunnels. In Figure 5.6 this measurement is displayed as a red arrow with a filled ellipse around that represents the uncertainty.

- **Local odometry**: This output simulates the measurements of a local odometry system, which are displayed in Figure 5.6 as blue arrows. To compute this odometry, the emulator integrates the agent's velocity. In each iteration, the agent's velocity is modified by adding Gaussian noise before it is integrated. Therefore, these measurements suffer from drift, as any other local odometry system would do.

- **Agent detections**: The agents are also able to provide detections of other agents relative to their pose, simulating a perception system. Detections are also affected by a parameterizable Gaussian noise, and have a maximum range, albeit occlusions are not considered.

Finally, the ground truth of the complete state of each agent is provided to be used as the reference in the performed experiments.

5.3.1.2 Carla simulator

Carla is a simulator built on top of *Unreal Engine*, which is an engine that was initially developed for game development applications. For this reason, the environments simulated with this simulator are very close to reality, and the data obtained from the virtual sensors have a high degree of fidelity.

The Carla simulator was specifically designed for the development of autonomous driving applications, so the vehicle and sensor models have been implemented with special attention to detail. Therefore, the experiments performed in Carla-based simulation environments will use simulated data that will be as close to reality as possible. Furthermore, the simulator has integration with ROS and ROS2, so all the software implemented for this work can be easily integrated with Carla.

The only drawback of the Carla simulator is that these simulations require a serious amount of computational resources, requiring a powerful machine and more time than other simulation suites.

In order to perform the validation experiments for the proposed approach, three different environments have been modeled in Carla. Considering that the proposed approach is only concerned about the movement of the vehicles and the vehicle detections, the designed environments only include the roads, thus simplifying the simulation model and allowing us to include more vehicles in the simulation with the same amount of computational power.

The first two simulation environments have been designed from scratch to represent simple navigation settings such as a straight road, displayed in Figure 5.7, and a curved road, shown in Figure 5.8. These two environments present one-way roads with a single lane, providing the vehicle agents with an environment where it is easy to drive.

The last of the simulation environments introduced has been extracted from a real-world location using OpenStreetMap. Particularly, this is a location next to

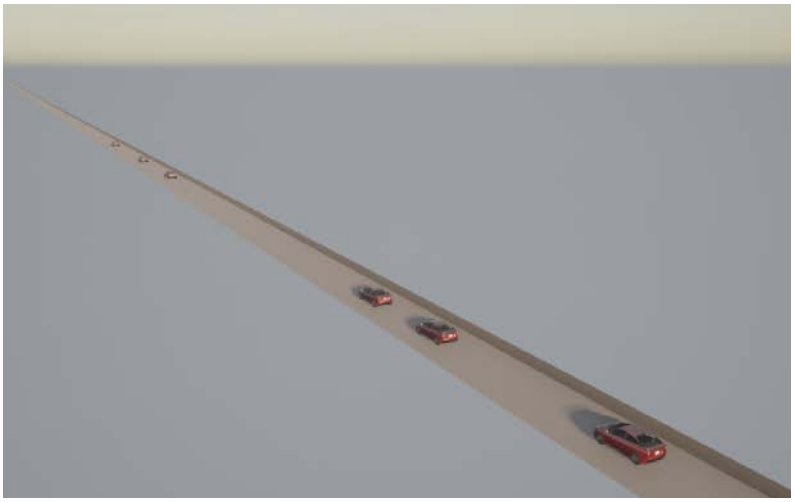


Fig. 5.7: Carla simulation scenario 1: Straight line.

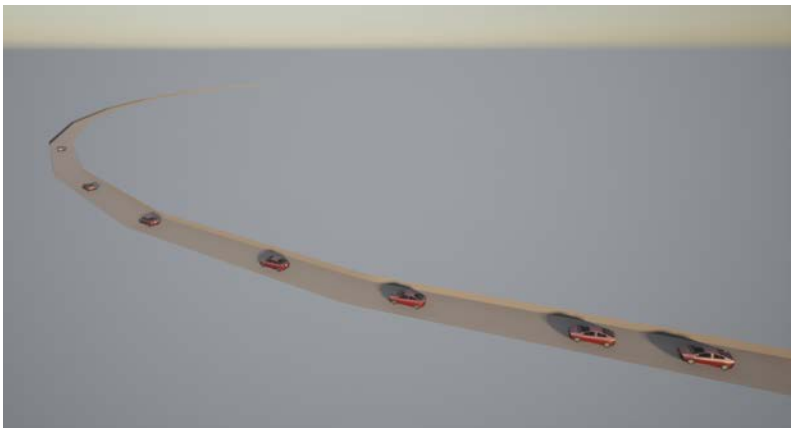
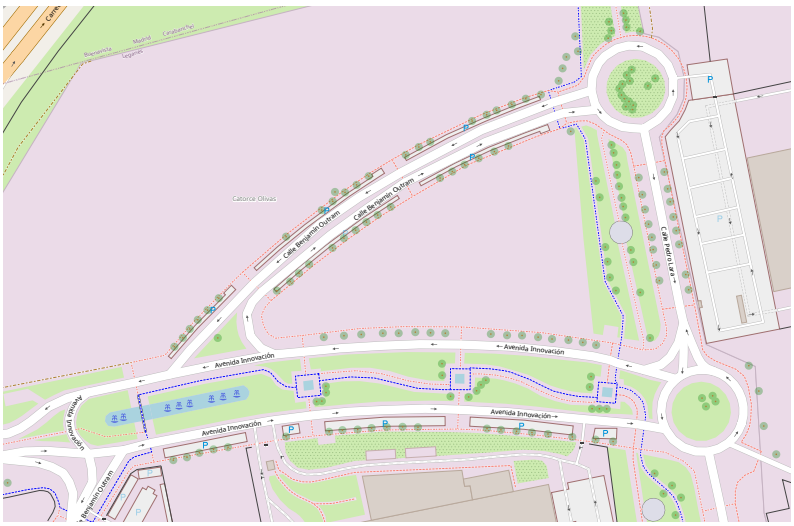


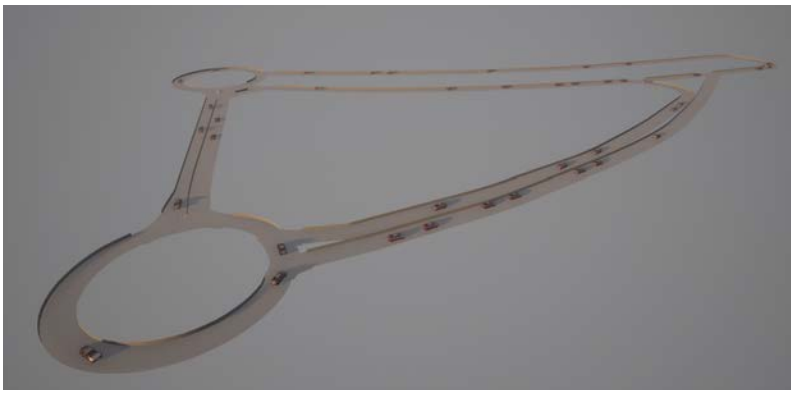
Fig. 5.8: Carla simulation scenario 2: Curved line.

the *University Carlos III de Madrid* building in the technological park in Leganés, which is also close to the main campus, where the laboratory is located. The map data extracted from OpenStreetMap has been processed using a tool from the Carla simulator to generate the simulation environment with the same roads as the map. This scenario is presented in Figure 5.9, where the top image shows the map area in OpenStreetMap and the bottom subfigure provides an aerial view of the environment simulated in Carla with some cars.

The selected area has the benefit of forming a closed circuit, forming a triangle composed of two roundabouts and a simple intersection. Being a closed circuit, the simulation can be run for an unlimited time, while always having the same amount of vehicles inside the circuit.



(a) Aerial view of OpenStreetMap data.
Map URL: `openstreetmap.org/#map=18/40.35904/-3.74442`.



(b) Simulated environment in Carla.

Fig. 5.9: Carla simulation scenario 3: Real-world location.

Regarding the vehicles used in the simulation environments based in Carla, these are based on a model of the Toyota Prius. The simulated cars are equipped with virtual sensors that provide localization data, such as a GNSS localization system, an IMU, and an odometer.

Additionally, the vehicles were initially equipped with a 3D LiDAR of 64 layers, which was going to be used as a perception sensor to obtain the vehicle detections using the method presented in [9]. Unfortunately, simulating a LiDAR sensor for each one of the vehicles requires a huge amount of computational resources, which made this approach unfeasible. Alternatively, the vehicle detection method used in the *agent emulator* was adapted to simulate vehicle detections from Carla's ground truth.

Finally, the vehicles implemented in Carla come with an autopilot system that is able to drive inside its current lane while maintaining a safe distance from the vehicle in front, if any. When the vehicle reaches an intersection, the autopilot is programmed to take a random option among the ones available. This autopilot has been used in all the experiments that were carried out, leaving the vehicle control to Carla. The implemented autopilot is perfect for the proposed simulation environments because the first two maps are single lanes and the last one is a closed circuit with well-defined lanes and intersections.

5.3.2 Processing time

One of the most important things to consider when implementing such an algorithm is how much time is required to run each iteration, and how well it escalates when the number of vehicles increases. Due to the high complexity of the method, the estimation process not only becomes more costly when the number of vehicles V increases, but also when the number of particles P selected for the Particle Filter is increased.

For this reason, the first study that was performed on the proposed algorithm was to analyze the computational time used with different values of V and P. This study aims to determine the limits and the feasible configurations of V and P that allow running the implemented algorithm in real-time. While the number of vehicles is not an actual parameter of the algorithm, it is decisive in the computational time, and it is useful to know how many vehicles can be handled in the estimation while providing a real-time output. On the other hand, the number of particles is a crucial parameter that must be adjusted. Generally, the number of particles should be increased as much as possible because with more particles it is possible to obtain a better estimation (more pose samples). The main drawback of increasing the number of particles is that the estimation requires more time. Therefore, this study has also the objective of finding a proper configuration value for the number of particles, P.

In order to perform this analysis, a simple scenario has been implemented using the *agent emulator* simulation environment. Because these tests are only focused on computational time, the performance of the estimation algorithm is not a concern. Therefore, these scenarios have all agents moving in circles at a constant velocity.

Then, the processing time is evaluated individually for each of the different sub-processes that constitute the algorithm:

- **Prediction**: The motion update step requires updating the pose of all vehicles in all particles, resulting in a complexity of $\mathcal{O}(VP)$.

- **Likelihood**: The likelihood process where the weights are computed is the most complex because the measurements between each pair of vehicles must be considered. Therefore, the complexity of this step is $\mathcal{O}(V^2P)$ if the vehicle detection matrices are complete, meaning that all vehicles were able to detect all vehicles in the system.

- **Update**: The update step, where the pose of each vehicle is estimated requires estimating the mean and covariance of each pose. Consequently, this process has a complexity of $\mathcal{O}(VP)$.

- **Selection**: The selection process is based on a tournament selection, which only considers the weight (or fitness) of each particle, regardless of the number of vehicles. For this reason, this is one of the most simple processes, since the complexity is lower than $\mathcal{O}(P)$ because only the set of parents is processed and $W \leq P$.

- **Crossover**: Finally, the crossover operation also has a complexity of $\mathcal{O}(VP)$, because each chromosome (vehicle) must be processed for each element in the offspring set.

Using the *agent emulator* environment, multiple test sequences are executed with different combinations of V and P values. Each sequence has a duration of 1 minute, providing a sufficient number of iterations to obtain an estimation of the computation time.

The average total time used in each iteration is presented in Figure 5.10, for up to 20 vehicles and 2000 particles. These measurements have been obtained using an AMD Ryzen 5 3600 CPU running at 3.6GHz, which can be considered an average CPU in terms of performance. Moreover, a special remark should be done to mention that the tested implementation of the proposed algorithm runs on a single CPU core. The nature of the proposed algorithm has a high potential for parallelization because most of the computations are independent for each particle. Therefore, the values presented in Figure 5.10 could be highly reduced with a better implementation that would favour parallelization.

As mentioned before, the most complex sub-process in the estimation process is the likelihood estimation, with a complexity of $\mathcal{O}(V^2P)$. The processing times of this single sub-process are displayed in Figure 5.11. Comparing these two figures

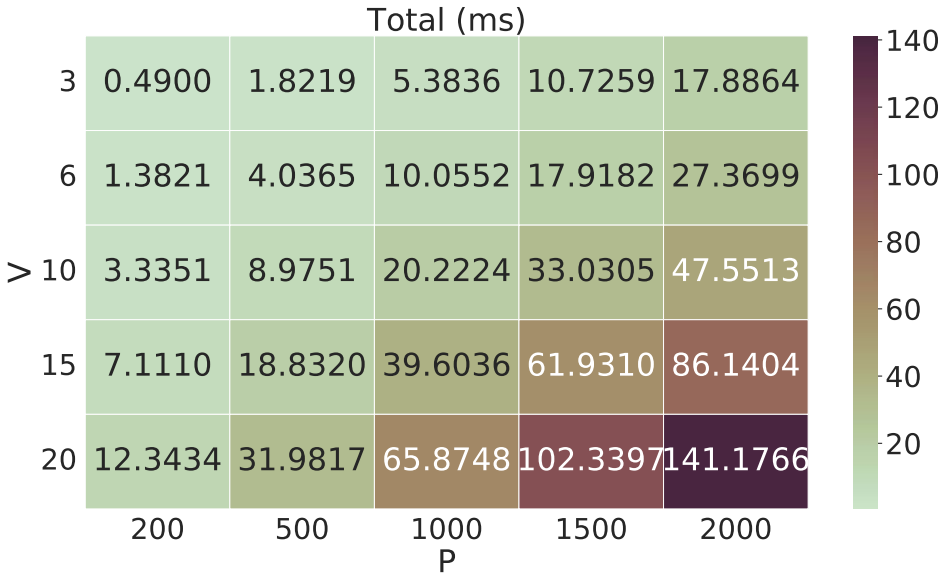


Fig. 5.10: Total processing time of each iteration, averaged for 1 minute-long sequences.

clearly shows that most of the total time in the process is consumed by the likelihood estimation, as was initially hypothesized.

Again, it should be noted that these measurements assume full detection matrices, implying that all V vehicles are able to detect all V vehicles, which is a pessimistic scenario. A more likely reality would have some pairs of vehicles that are not able to detect each other, thus leaving empty spaces in the detection matrix. These elements are not processed, so the total likelihood time will always be lower in a realistic scenario.

Considering all this, we believe that the proposed approach is suitable for real-time operation in environments with up to 20 vehicles without any problem. Actually, finding a scenario with more than 20 vehicles where the detection matrix is complete is not easy, because more vehicles lead to more occlusions and a higher separation between vehicles, keeping them out of range. Therefore, we find 20 to be a reasonable upper bound to evaluate the processing time since it is not likely to have detection matrices with more elements.

Regarding the number of particles, 10Hz is generally assumed to be the lower bound frequency for a global localization system to operate in real-time conditions. Therefore, if the system needs more than 100ms to obtain an estimation, it will be considered too slow.

With the obtained results, we could raise P up to 1500 particles, and this constraint would be satisfied. Nevertheless, the remainder of the experiments will set the

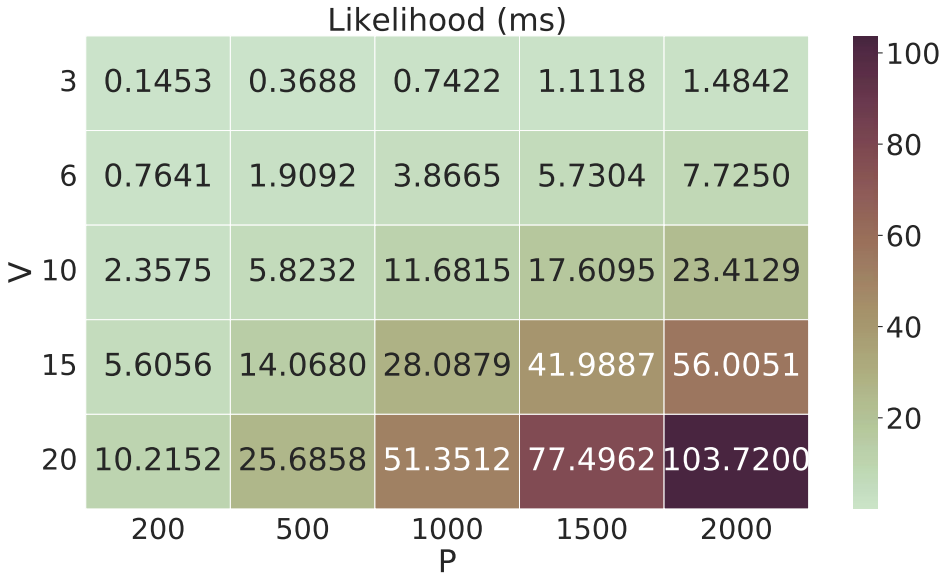


Fig. 5.11: Processing time of the likelihood process, averaged for 1 minute-long sequences.

number of particles P to 1000 for two reasons. On the one hand, 1000 is a well-round number that should be more than sufficient to reach a proper estimation of the vehicle poses. On the other hand, this setting would allow the algorithm to easily run at up to 20Hz, providing a higher-rate estimation and reducing the time between predictions.

5.3.3 Parameter tuning

The performance of the proposed approach is very sensitive to the parameters that control the behavior of the proposed genetic resampling method. For this reason, it is important to properly adjust these parameters.

In this section, the parameter tunning process that has been performed is described. First, the different simulation scenarios that were designed to carry out the tunning process are detailed. Then, the tunning process is presented, explaining the methodology used. Finally, the obtained results are discussed.

Since this process requires changing and testing multiple combinations of parameter values, a high number of simulations must be carried out. Therefore, these tests are performed using the *agent emulator* environment, allowing us to automatically change the parameter configuration and run the multiple tests in batch. The simulation scenarios used to obtain the data for this process included a fixed number of vehicles $V = 6$ and particles $P = 1000$. Additionally, the movement of the vehicles

follows a circular trajectory with a linear velocity of around $2m/s$. Regarding the detection system, these tests have no restriction in the maximum detection range, thus allowing all vehicles to detect each other. Finally, the error of the GNSS system that provides the prior estimations of the vehicle poses is considered a variable as well, because it is used in the initialization and mutation steps and should affect the quality of the final estimation.

In the proposed algorithm, the parameters that require a tuning process are the following:

- **Number of particles** (P): The number of particles has been fixed at $P = 1000$ according to the results obtained in the time processing analysis.
- **Number of selection winners** (W): This parameter determines how many potential solutions have a direct pass from one iteration to the next. The parameter that is actually controlled is w_p, which is a percentage of the total population.
- **Tournament size** (T): The tournament size is also defined as a percentage of the population size, by t_p. This parameter determines how many particles enter the tournament in the selection process.
- **Mutation rate** (M): The mutation rate is a percentage that determines the chances that each chromosome from an offspring particle has to be affected by the mutation process described previously.

The procedure that was followed to understand how these parameters affect the performance of the filter is based on testing the algorithm with different combinations of values. Fortunately enough, the three remainder parameters are defined as percentages, so their possible values are constrained.

Each combination of parameter values represents a testing scenario, and each scenario is repeated 10 times to have more samples and avoid relying on a single run. The sequences have a duration of 1 minute each, which provides enough measurements due to the fact that the estimation algorithm is set to run at 20Hz. For each sequence, the ground truth and the poses estimated by the *LoCo* framework are recorded and analyzed offline using EVO [264]. The metric used to evaluate the localization performance is the APE, as defined earlier in the previous chapter. Particularly, the average and standard deviation of the APE metric are first computed for each individual vehicle in the system. Then, the mean and the standard deviation of the APE are computed, by combining the APE of each vehicle in each one of the 10 sequences.

Finally, the results are analyzed using data visualization techniques to find proper values for the presented parameters. Note that this type of analysis will not reach the actual optimal configuration of parameters. Instead, the objective of this procedure is to understand the contribution of each parameter and to find a configuration that is good enough.

An initial test is performed, evaluating all parameters with variable values. Due to the high amount of time required to evaluate each scenario, this initial test is done with a reduced set of possible values for each variable, resulting in a coarse analysis. The values used in this initial test are $w_p = \{30, 50, 75\}$, $t_p = \{10, 20, 30, 50, 75\}$, and $M = \{10, 50, 75\}$. Additionally, the standard deviation of the GNSS error is set to 2m, which is approximately what low-cost solutions are able to offer. As it can be noted, this test has a higher number of possible values for the parameter governing the tournament size, intending to find a proper value for this parameter and leave it fixed in future tests.

The obtained results in this preliminary test are displayed in Figure 5.12. In this figure, the rows and columns are associated with the values of w_p and t_p, respectively. Then, inside each subgraph, the mean of the error is divided by the mutation rate M. The values for T, W, and M are percentages (%).

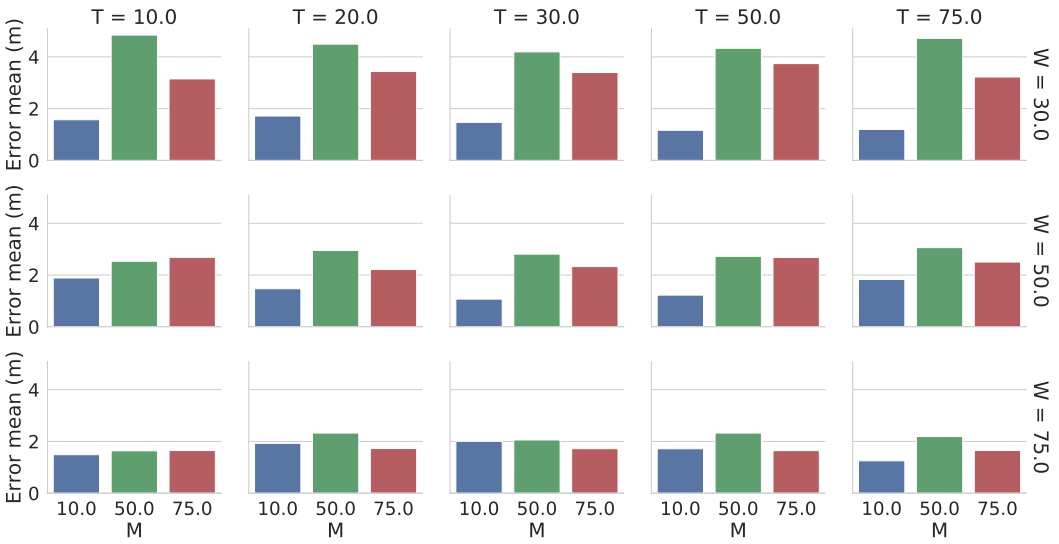


Fig. 5.12: Average error mean for the initial tuning test, averaged on 10 sequences.

While the amount of information might be too much, it is shown that the performance is mainly affected by the number of winners, W, and the mutation rate, M. Regarding the tournament size, even though it does not seem to have a contribution as big as the other two variables, it can be appreciated that the values around

30% seem to have better performance overall, especially when the values of W are small.

For this reason, the value of T is fixed to 30% in the next experiments. Fixing this parameter allows us to focus deeper on the effect of W and M, increasing the number of values that can be evaluated for each parameter. Moreover, the performance of the proposed algorithm is more sensitive to changes in W and M than it is to the changes in T, so it is a good idea to focus the testing efforts on studying the effect of the former two parameters.

Accordingly, the second stage of the tuning tests is aimed at exploring the configuration space formed by the parameter W and M. After fixing the tournament size at 30% of the total population, the possible parameter values that are explored in this stage are $w_p = \{10, 20, 30, 40, 50, 75\}$ and $M = \{0, 5, 10, 25, 50, 70, 90\}$. Furthermore, we acknowledge that the quality of the prior GNSS pose estimation is of great significance in the proposed method and directly affects the posterior estimation. For this reason, the error of the prior estimation is also considered a variable in this test, which follows the configurations represented in table 5.1.

GNSS Setting	Translation error std (m)	Rotation error std (rad)
XS	0.1	0.001
S	0.5	0.005
M	2.0	0.02
L	5.0	0.02

Tab. 5.1: GNSS noise settings.

The proposed GNSS settings emulate four different cases. The XS configuration would be the equivalent of a very accurate system with RTK corrections, reaching an accuracy level with a standard deviation of 10cm. The next configuration has a standard deviation of 50cm, similarly to the average output of a GNSS system with differential corrections. Finally, the last two settings correspond to a low-cost system, with good (M) and bad (L) signal coverage.

Similar to what was done in the first stage, the possible parameter values are combined and each configuration scenario is repeated 10 times, generating multiple sequences to afterward average the resulting data. The collected results are presented in Figures 5.13 and 5.14, which show the mean and standard deviation of the localization error, respectively.

After carefully studying the data resulting from the multiple experiments performed, it is possible to extract the following insights:

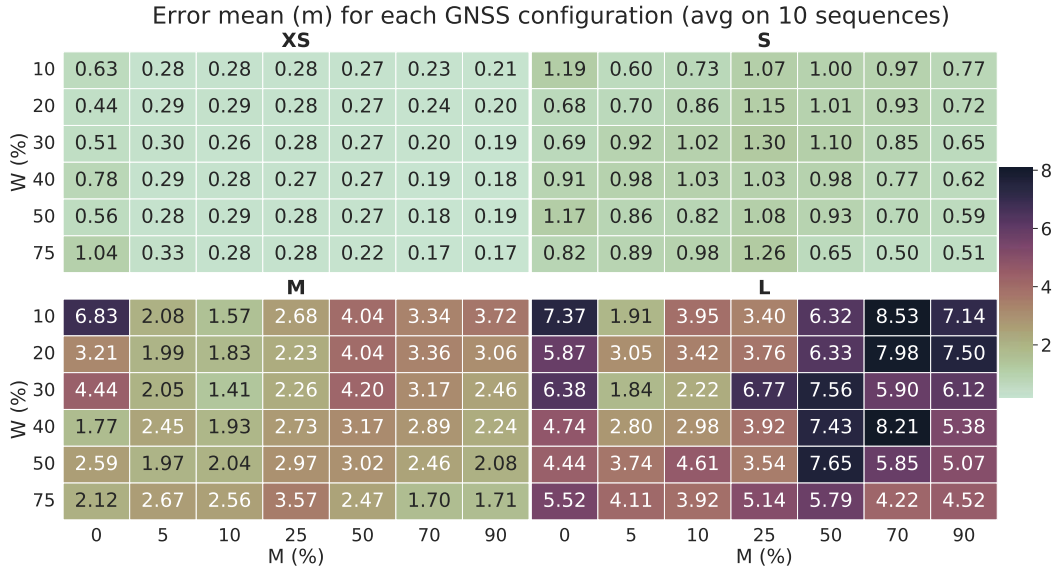


Fig. 5.13: Average error mean for the second tuning test, divided by GNSS noise and averaged on 10 sequences.

- Having high W values means that the initial solutions are more likely to be kept in the population. Since only the offspring introduces new solutions and mutations, a high W means having less variation in the estimation, with the possibility of getting stuck with the initial estimation for a longer time.

- If the GNSS is very accurate (RTK level), it is usually better to have a high W. If the system has a good GNSS, but a low W, it is more likely that the good initial position will be corrupted by the filter. Keeping more solutions from previous iterations helps maintain a stable estimation, assuming that the initial estimations are good.

- If the GNSS is bad, and M is low, the offspring will receive fewer mutations, so it is more likely to keep the initial solutions, which are probably inaccurate. For that reason, it is better to have a low W, so the population changes faster and the wrong initial solutions can be forgotten and improved.

- If the GNSS is bad, but M is high, the offspring will receive many mutations with inaccurate estimations from the GNSS system. In that case, it is better to have a high W, so the number of generated children is reduced and fewer mutations occur.

- If the GNSS is bad, M should be low (low but not zero, see next point). It is not clear if it is better to have high r low W, but with a bad GNSS, M must be lo (<30%).

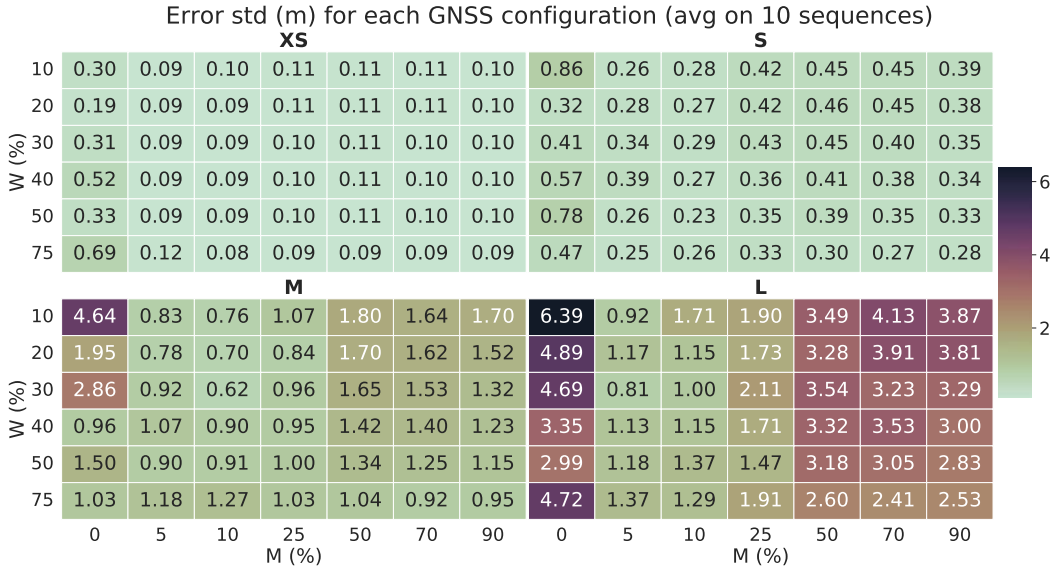


Fig. 5.14: Average error std for the second tuning test, divided by GNSS noise and averaged on 10 sequences.

- If the GNSS is bad, it could seem reasonable to leave $M = 0$, so the offspring never receives mutations from a bad GNSS estimation. This, however, always gives bad results. The reason is that if the GNSS is bad, it is more likely to get a very bad initialization. That means that the filter might converge to a bad estimation, or not converge at all because the measurements will never make sense. For that reason, it is good to have a small mutation rate (5-10%) even with a bad GNSS, so those mutations can help redirect the estimation to the actual position.
- If the GNSS is good, it is always better to have a high M, allowing the offspring to benefit from the accurate mutations.
- The proposed system is very sensitive to the GNSS error. Even though the final estimation error can outperform the GNSS system with the proper parameter settings, the estimation error is increased when the prior estimation is worse.

In order to analyze the contribution of GNSS errors at the initialization and mutation steps, a final experiment is conducted. This test has the same parameter combinations as the previous one, but this time the mutation rate is set to $M = 0$, completely removing mutations from the system. This way, the only contribution of the GNSS prior estimation is in the initialization step. Figure 5.15 presents the results obtained from this last experiment, where it can be seen how a bad GNSS system still affects the quality of the estimation, even if it is only used in the initialization of the population.

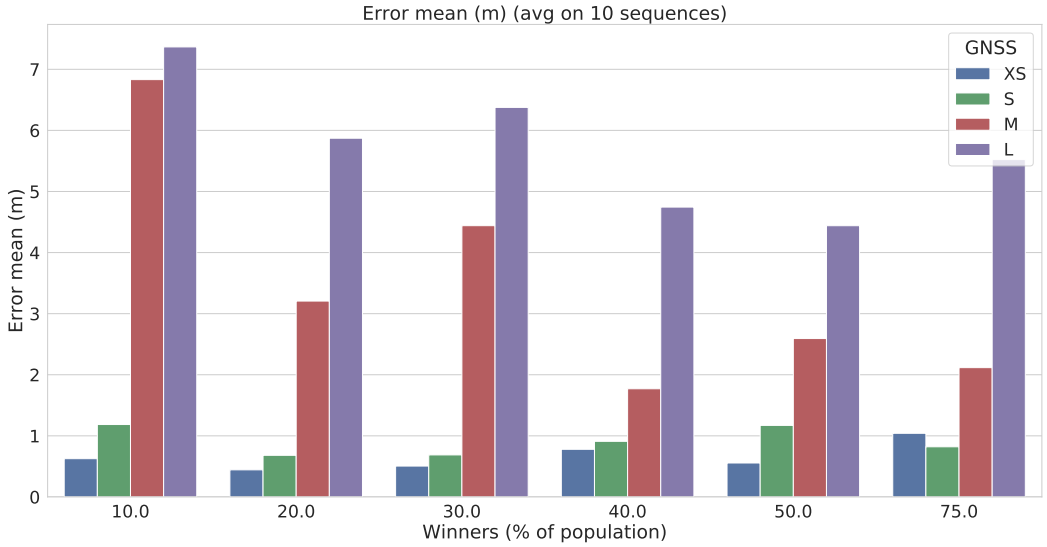


Fig. 5.15: Error mean analysis with $M = 0$, divided by GNSS noise and averaged on 10 sequences.

As one can observe in Figure 5.15, the systems with poor GNSS accuracy tend to perform bad regardless of the value of W. Since the mutation is eliminated in this test, the GNSS system is only used to initialize the particles. Therefore, it can be concluded that a bad initialization can lead to the system converging to the wrong estimation. While this could be seen as a bad thing, it only confirms our initial hypothesis and corroborates the importance of the proposed mutation approach. This problem, which arises when $M = 0$, is drastically reduced as soon as the mutation rate increases, as can be observed in Figure 5.13.

To summarize, a balance between W and M must be reached; in order to be able to handle different types of GNSS systems. In cases where the vehicles that form the cooperative system have a specific, previously known, type of GNSS system, the parameters can be better tuned for that specific system according to the performed study. In conclusion, the proposed values for the remainder of the experiments are $W = 30$ and $M = 10$, resulting in a balanced configuration that has been reached after carefully analizing Figures 5.13 and 5.14.

5.3.4 Genetic resample validation

One of the major contributions of the proposed approach is the genetic resampling in the Particle Filter to estimate the vehicle poses. This resampling approach should

theoretically perform better than traditional resampling methods in the given localization problem due to the high dimensionality of the state. Nevertheless, this hypothesis must also be validated experimentally.

To that end, a new batch of experiments is prepared, evaluating the performance of the proposed algorithm and a typical resampling method under different GNSS noise settings. These experiments are also based on the *agent emulator* environments and have a setup similar to the one used in the tuning experiments. In addition to the tunning tests, where the number of vehicles was fixed in $V = 6$, these experiments include tests with different numbers of vehicles from the set $V = \{4, 7, 9\}$.

In addition to the GNSS noise settings previously defined, a new scenario is added in these experiments, where the noise of the GNSS system is randomly selected among the defined configurations (XS, S, M, L) for each vehicle. This new setup resembles a more realistic scenario where each vehicle might have a different type of GNSS unit.

Regarding the resampling comparison, the baseline is a Sample Importance Resampling (SIR) approach as defined in [108]. With this resampling method, the particles for the new iteration are drawn with a probability according to their weight. Since the weights are normalized (i.e., $\sum_k \omega_k = 1$), each weight ω_k represents the probability that the particle k has of being resampled for the next iteration. The rest of the Particle Filter algorithm implementation is exactly the same, using the *LoCo* framework, to ensure that the comparison of the two resampling methods is as fair as possible.

Each scenario is also repeated 10 times, generating multiple sequences to average the error afterward. In the scenario where the vehicles have a random GNSS noise setup, the random combination of GNSS settings is computed once and maintained for the 10 sequences.

After gathering the localization data from each scenario and aggregating the results from all the sequences in each scenario, the extracted results are summarized in Figure 5.16. The graph presented in this figure shows the distribution of the localization error of each resampling method (simple vs genetic) in addition to the error distribution of the prior estimation obtained directly from the raw GNSS data. The last GNSS configuration (R) represents the scenarios where the GNSS unit of each vehicle has a random noise setup selected from the other four.

From the extracted results, it is clear that the proposed approach outperforms the SIR method and also improves the accuracy of the prior estimation, as was expected. However, the SIR approach is not even able to achieve a better estimation accuracy

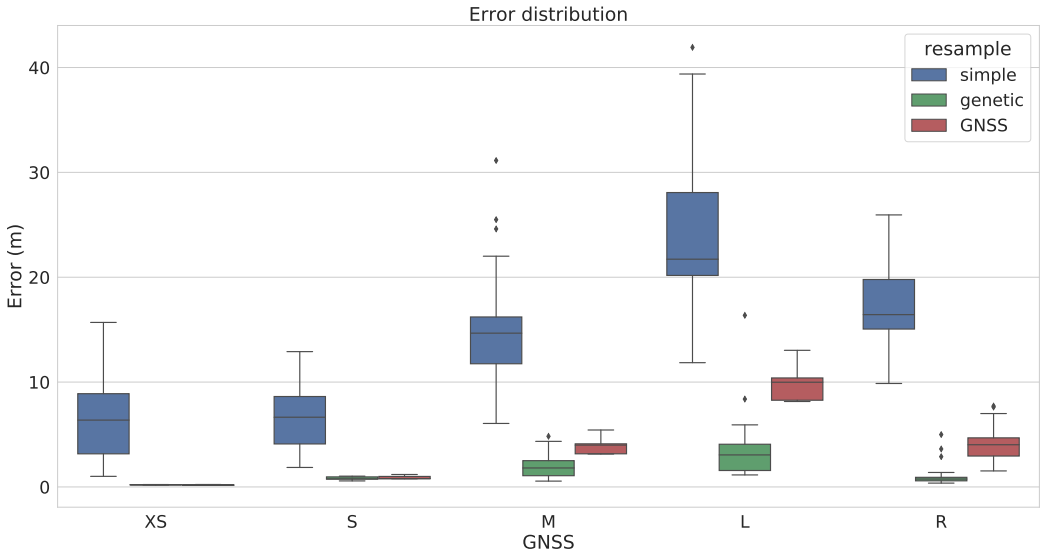


Fig. 5.16: Error distribution with each resampling approach.

than the prior estimation from the GNSS system. While this approach is suitable for cooperative localization problems that only estimate the pose of the ego vehicle, the problem to solve presents a much higher degree of complexity because all poses must be estimated jointly. For this reason, the simple resampling approaches that are typically used in Monte Carlo localization methods are not able to reach a good estimation of the presented problem. In addition to the problem of having a wrong initialization (as seen in Figure 5.15), these resampling approaches have limited exploration capabilities resulting in an impoverishment of the population after some iterations.

Another interesting observation is that the proposed approach performs very well in the scenario where the GNSS noise is randomized. This indicates that the proposed approach is able to benefit from the vehicles that have a good prior estimation in order to improve the estimation of all vehicles.

Finally, Figure 5.17 presents a zoom of the boxplots for the genetic resample results and the prior GNSS estimation for the XS and S configurations, which were not properly visible in the previous graph. In this zoomed section it can be perceived that the proposed approach is only outperformed by the raw GNSS estimation when it has very high accuracy (around 10cm of error).

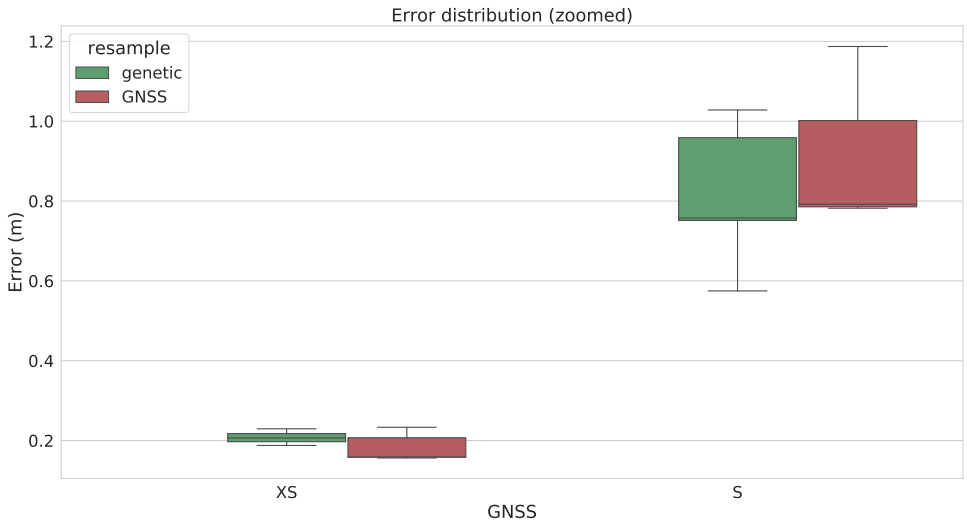


Fig. 5.17: Error distribution with each resampling approach (zoomed).

5.3.5 Performance analysis

Finally, the proposed approach is further validated using a more realistic simulation environment implemented in the Carla simulator. The main disadvantage of the *agent emulator* is that the vehicle trajectories are too artificial and do not follow an Ackermann model. For this reason, an extra set of experiments is performed on the simulation environments presented in Section 5.3.1.2, which are based on the Carla simulator. The main benefit of these experiments is being able to evaluate the performance of the proposed approach with realistic car-like trajectories.

Furthermore, the performance of the proposed approach is also compared with the pose estimation results obtained with a more traditional method based on the Kalman filter. More specifically, the proposed cooperative localization approach will be compared with a UKF estimator that estimates the pose of each vehicle independently without using any information about the rest of the vehicles.

5.3.5.1 Experimental setup

The simulation environments used in these experiments are those defined in Section 5.3.1.2. The first two environments present simple scenarios with a single-lane road that is either straight (first environment), or curved (second environment). Additionally, the third environment is based on a real-world location and includes a closed circuit with a triangular shape, composed of two roundabouts and an intersection.

Regarding the number of vehicles included in these simulations, the first two simulation environments have different scenarios with 4, 7, and 15 vehicles. The third environment, however, has scenarios with 20 and 30 vehicles driving at the same time. All these vehicles move using Carla's autopilot, with a cruise velocity of around $20Km/h$. In addition to increasing the number of vehicles, another difference from the previous experiments performed using the *agent emulator* is that now the maximum detection range is set at 80m, also following a more realistic approach.

Furthermore, the GNSS noise is configured to have a standard deviation around 2m, similar to the configuration S previously defined. This is a typical noise value for most common low-cost GNSS receivers, similar to the ones equipped in cars and phones, thus resembling a realistic real-world scenario.

Finally, the UKF estimator is configured to fuse the prior estimation from the GNSS system, the local odometry of the vehicle, and the IMU system. The UKF implementation is the one provided in the ROS *robot_localization* package [283].

5.3.5.2 Results

The localization data from each one of the experimental scenarios are recorded in sequences with a duration of approximately 1 minute. These sequences include the simulation ground truth, the prior estimation from the GNSS system, the pose estimation from the proposed cooperative localization method (obtained with the *LoCo* framework), and the UKF pose estimation (obtained with the *robot_localization* package).

Then, the estimated poses from the proposed approach and the UKF estimator are evaluated using EVO [264] to extract the APE metric for each individual vehicle in each recorded sequence. The proposed analysis of the localization data is the same as the one used in the resampling evaluation experiments, where the APE metric obtained from each vehicle in each sequence is combined.

The distribution of the error according to the APE metric is represented in Figure 5.18, by using violin plots. These plots show the distribution of the error, in addition to the mean (white point) and the quartiles (inner black line).

From these results, one can see that the proposed approach is able to improve the localization accuracy of the prior estimation provided by the GNSS system, in the same way as was observed in the *agent emulator* experiments. Likewise, the pose estimation obtained from the UKF data fusion is also able to outperform the

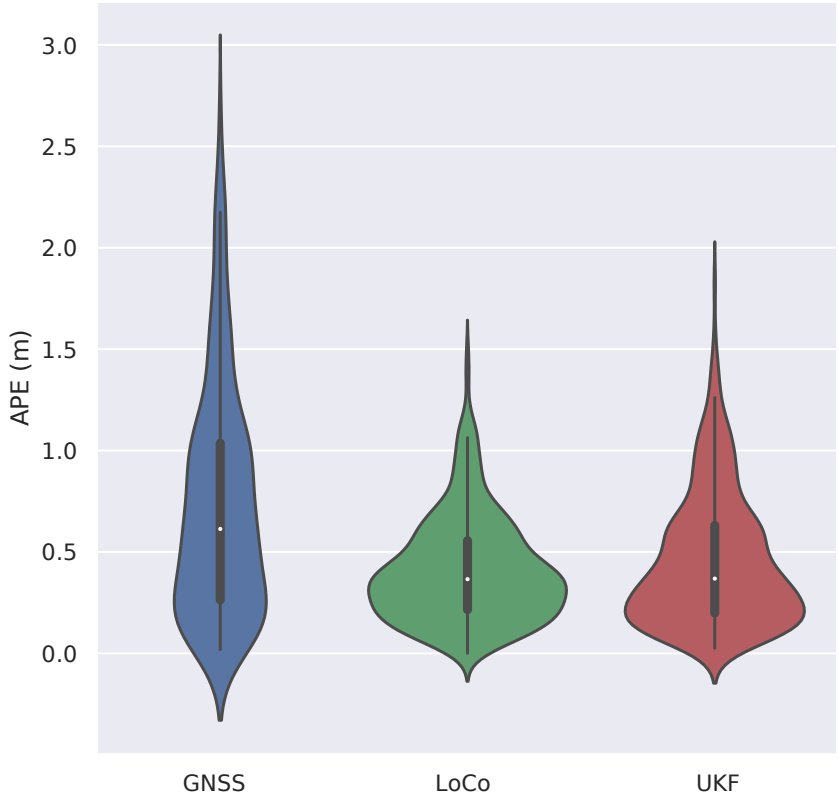


Fig. 5.18: Error distribution in Carla's simulation experiments.

standalone GNSS localization system, justifying why it is one of the most used filters in localization systems for autonomous vehicles.

When comparing the proposed cooperative localization approach with the traditional UKF estimator, it can be observed that both estimators seem to perform similarly, although the error distribution of the proposed approach is bigger towards the lower values. A summary of the numerical results is presented in Table 5.2, showing the mean, standard deviation, and maximum value of the APE metric. In this table, it is easier to see that the proposed approach is able to slightly outperform the UKF estimation, with the help of the cooperation between the vehicles.

	GNSS	**LoCo**	**UKF**
Error mean (m)	0.743	0.410	0.455
Error std (m)	0.573	0.261	0.326
Error max (m)	2.701	1.505	1.857

Tab. 5.2: Performance results on Carla's simulations.

5.4 Concluding remarks

The problem of cooperative localization in outdoor vehicles is a real challenge for the future of smart cities, especially in areas where the availability of GNSS systems is limited or inexistent. In this chapter, a cooperative localization algorithm based on a Genetic Particle Filter is proposed; in order to jointly estimate the pose of all the vehicles that constitute the cooperative system. While other works in the literature perform pose optimization just for the ego vehicle, the proposed method approaches the problem by jointly estimating all poses, trying to improve some of the deficiencies of the state-of-the-art methods.

The proposed Genetic Particle Filter has been proven to perform better than other state-of-the-art methods used in cooperative localization when concurrently estimating all poses. Particularly, the benefits of genetic resampling against traditional resampling approaches have been demonstrated through experimental data in a simulation environment. The fact that typically used resampling methods are not able to solve the presented problem is that finding multiple poses instead of just one drastically increases the complexity of the problem. In that scenario, it is required a filtering method capable of exploring and generating more potential localization solutions when processing the sensor data.

Additionally, a tuning mechanism for the proposed approach has been presented, manually adjusting the multiple parameters based on data visualization techniques. This mechanism has been exploited to study and understand the inside of the genetic resampling method, analyzing the contribution of each parameter to the performance of the estimator. In future work, an automatic tuning method should be considered, allowing the localization system to adapt to each scenario.

Finally, the performance of the proposed approach has been validated in a simulation environment based on Carla, which is one of the most realistic ways to perform such type of experiment without a fleet of 30 real autonomous vehicles. In these experiments, the proposed approach has been compared with a traditional localization approach based on a UKF estimator. While both estimators are able to provide a localization solution that outperforms the initial GNSS solution, their performance is similar. Nevertheless, the proposed cooperative localization approach has shown better results than the UKF, as displayed in Table 5.2. Even though the proposed approach has a higher complexity, it is still able to run in real-time (20Hz) on a modest machine, so the additional complexity can be justified since the cooperative solution outperforms the classical UKF approach.

Conclusion and Future Work

6

> *Never confuse education with intelligence, you can have a PhD and still be an idiot.*
>
> **— Richard P. Feynman**

Autonomous driving technologies have shown a high potential for improving road safety and providing new means of transportation without human drivers. Furthermore, the numerous and remarkable advances in recent years have demonstrated the suitability of these technologies, making autonomous driving a tangible reality. The perspectives for the future include even greater adoption of these technologies in our daily lives. Therefore, it is essential to continue research lines related to autonomous driving, especially the topics focused on improving the safety and reliability of vehicle systems.

Accordingly, the work presented in this thesis aims to push the current state of the art of localization technologies for autonomous vehicles. Particularly, the humble contributions of this thesis are focused on understanding and exploiting the perception of the environment to increase the reliability of localization systems.

This section presents a summary of the conclusions extracted from the work presented in this thesis and introduces a series of suggestions for future work to continue these research lines.

6.1 Conclusion

With the rapid advances in autonomous driving technologies, commercial vehicles are getting more intelligent. Among these advances, the research community has placed most of its focus on perception technologies, which have shown a huge boost in performance in the latest 10 years. Localization technologies, on the other hand, have not received as much attention, and GNSS systems are still widely used as the primary localization system in vehicles.

The objectives that were marked for this thesis were, overall, directed at improving the accuracy and reliability of localization systems by the means of exploiting such advances in perception systems. With that global goal in mind, a summary of the main contributions of this work is presented below:

- Globally, all the contributions presented in this thesis use the strengths of perception systems to improve localization. The perception technologies exploited include 3D semantic segmentation to pre-filter point clouds, infrastructure detection methods to use traffic lights or lamp posts as landmarks, and vehicle detection systems to find the position of other vehicles in a cooperative system.

- An exhaustive study of the performance of LiDAR-based odometry when the input point clouds are filtered based on semantic knowledge has been carried out. In this work, the input point clouds have been filtered using the information from a 3D semantic segmentation method.

  The experimental results obtained using the SemanticKITTI dataset show an overall improvement with some specific filtering configurations, while providing very useful insights regarding the influence in the odometry of each type of element in the environment. The proposed filtering approach has been tested in a real-world research platform, where most of the insights previously extracted have been corroborated.

- A novel technique to estimate the expected localization error using a landmark-based localization method in controlled environments has been presented. In this work, a sensor-agnostic calibration method is proposed; in order to determine the measurement error covariance of a landmark detection method. This calibration method is also independent of the type of perception algorithm used to detect the landmarks. Then, the landmark measurement model is exploited to build an accuracy heatmap that represents the expected localization error at each point in the testing area.

  This method can provide useful information about the accuracy of the localization estimation in an environment with a given set of landmarks, allowing a vehicle to use it to its advantage.

- A landmark placement optimization algorithm that exploits the previously presented accuracy heatmaps has been proposed. This method is suitable for finding a configuration of landmarks that guarantees an upper error bound in the localization estimation from a landmark-based localization algorithm. The applications of such an optimization method are relevant for the evaluation of other localization systems, as it can be used to generate a reference system with

a controlled error. Additionally, the proposed approach is of particular interest for controlled environments such as proving grounds, parks, or warehouses.

The proposed optimization method has been validated through multiple simulation experiments, showing that the localization error obtained with the generated reference never exceeds the desired limit.

- A novel Genetic Particle Filter has been proposed to solve the cooperative localization problem with multiple connected vehicles. The presented algorithm solves the pose estimation of all vehicles jointly, and it has been proven that the proposed genetic resampling approach outperforms traditional resampling methods. The proposed multimodal cooperative system makes use of vehicle detections to estimate the relative transformation between the vehicles in the system, instead of using a specific ranging system as other approaches in the literature. The vehicle detections are obtained from the vehicle's obstacle detection system, thus reusing the output from the perception system.

  Multiple experiments in different simulation environments have shown that the proposed approach is able to improve the localization estimation from GNSS systems. Additionally, a study on the contribution of each parameter in the filter has been presented for tuning purposes.

In general, all these contributions properly cover the objectives that were initially marked for this thesis, which could be considered successfully covered. Consequently, the works presented in this thesis were focused on enhancing localization systems through perception data in intelligent vehicles, by covering the disadvantages of GNSS systems and concentrating on the reliability of the systems.

Last, but not least, the work presented in this thesis, and other works derived from it, have contributed to the publication of 4 journal articles, 9 conference articles, one book chapter, and the submission of one patent. The details of the scientific publications produced with this work, in addition to other research merits are detailed in the preamble sections.

6.2 Future Work

There is still a long road ahead of us concerning fully autonomous vehicles. While the obtained results in this thesis are satisfactory, some points could be improved:

- In Chapter 3, it would have been beneficial, for the sake of completeness, to test other LiDAR-based odometry algorithms. Furthermore, the use of additional datasets would have increased the variety of driving scenarios in the study. Particularly, driving sequences with a higher number of dynamic agents would have helped in the analysis of the filtering configurations where dynamic objects were removed.
- The landmark placement optimization method proposed in Chapter 4 has been designed to work offline. For this reason, the initial implementation was not focused on the performance of the optimization algorithm, as long as a valid output was generated. Therefore, an additional study could be performed to analyze the convergence of the proposed optimization algorithm and adjust the parameters of the genetic algorithm.
- The mutation rate in the cooperative localization method presented in Chapter 5 is set as a fixed value, although the parameter tunning study showed that this parameter is linked to the GNSS accuracy. In order to increase the adaptability of the system, a dynamic mutation approach could be studied, where the mutation rate of each vehicle depends on the covariance of the prior GNSS estimation. This way, the system would be more likely to mutate a chromosome if the prior estimation is more accurate.
- The cooperative localization system was only validated in simulation experiments, mainly because gathering a fleet of intelligent vehicles big enough is usually out of reach. However, it would have been valuable to also test the proposed approach with real-world platforms using real sensor data.

Additionally, future work on the research lines followed by this work could expand the presented contributions in different ways. Firstly, the insights extracted from the semantic study could be exploited to build or enhance LiDAR-based odometry algorithms. Regarding the calibration method proposed to estimate the landmark measurement model, better approaches could be applied. Instead of performing the calibration manually based on data analysis and visualization, designing an automatic calibration method based on machine learning could provide significant value. Finally, part of the work presented in this thesis has been only validated in simulation environments, for justified reasons. Nonetheless, the integration of the proposed methods into real-world platforms would be the next natural step in this work.

Bibliography

[1] Francisco Miguel Moreno, Carlos Guindel, José María Armingol, and Fernando García. "Study of the Effect of Exploiting 3D Semantic Segmentation in LiDAR Odometry". In: *Applied Sciences* 10.16 (2020), p. 5657 (cit. on pp. v, 41).

[2] Miguel Ángel de Miguel, Francisco Miguel Moreno, Pablo Marín-Plaza, et al. "A Research Platform for Autonomous Vehicles Technologies Research in the Insurance Sector". In: *Applied Sciences* 10.16 (2020) (cit. on pp. v, 41, 55).

[3] Francisco Miguel Moreno, Ahmed Hussein, and Fernando Garcia. "Landmark Placement Optimization for Accurate Localization in Autonomous Vehicles". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 128–134 (cit. on pp. v, 63, 64).

[4] Fabian Pucks, Roman Blaschek, Stefan Rechenberger, Ahmed Hussein, and Francisco Miguel Moreno. "Verfahren, Steuergerät und Fahrzeug zur Validierung der Lokalisierung eines Fahrzeuges durch Referenzierung zu statischen Objekten im Umfeld". DE 10 2020 132 397.2. Dec. 2020 (cit. on p. vii).

[5] Abdulla Al-Kaff, María José Gómez-Silva, Francisco Miguel Moreno, Arturo De La Escalera, and José María Armingol. "An appearance-based tracking algorithm for aerial search and rescue purposes". In: *Sensors* 19.3 (2019), p. 652 (cit. on p. vii).

[6] Pablo Marin-Plaza, David Yagüe, Francisco Royo, et al. "Project ARES: Driverless transportation system. challenges and approaches in an unstructured road". In: *Electronics* 10.15 (2021), p. 1753 (cit. on p. vii).

[7] Abdulla Al-Kaff, Francisco Miguel Moreno, and Ahmed Hussein. "Ros-based approach for unmanned vehicles in civil applications". In: *Robot Operating System (ROS)*. Springer, 2019, pp. 155–183 (cit. on p. vii).

[8] Bahae Abidi, Francisco Miguel Moreno, Mohamed El Haziti, et al. "Hybrid V2X Communication Approach using WiFi and 4G Connections". In: *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2018, pp. 1–5 (cit. on p. viii).

[9] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, et al. "BirdNet: A 3D Object Detection Framework from LiDAR Information". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3517–3523 (cit. on pp. viii, 19, 120).

[10] Mostafa Osman, Ricardo Alonso, Ahmed Hammam, et al. "Multisensor Fusion Localization using Extended Hinf Filter using Pre-filtered Sensors Measurements". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1139–1144 (cit. on pp. viii, 32).

[11] Francisco Miguel Moreno, Omar El-Sobky, Fernando Garcia, and Jose Maria Armingol. "Hypergrid: A Hyper-Fast ROS-Based Framework for Local Map Generation". In: *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2019, pp. 1–6 (cit. on pp. viii, 70).

[12] Abdulla Al-Kaff, Angel Madridano, Ahmed Radwan, Francisco Miguel Moreno, and Ahmed Hussein. "Heterogeneous Multiple Vehicles Cooperation Approach for Smart Roads". In: *11th International Micro Air Vehicle Competition and Conference (IMAVS)*. 2019 (cit. on p. viii).

[13] Miguel Ángel de Miguel, Francisco Miguel Moreno, Fernando García, Jose María Armingol, and Rodrigo Encinar Martin. "Autonomous vehicle architecture for high automation". In: *International Conference on Computer Aided Systems Theory*. Springer. 2019, pp. 145–152 (cit. on p. viii).

[14] Carlos Justo de Frías, Abdulla Al-Kaff, Francisco Miguel Moreno, Ángel Madridano, and José María Armingol. "Intelligent Cooperative System for Traffic Monitoring in Smart Cities". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 33–38 (cit. on p. viii).

[15] Walter Morales Alvarez, Francisco Miguel Moreno, Oscar Sipele, Nikita Smirnov, and Cristina Olaverri-Monreal. "Autonomous Driving: Framework for Pedestrian Intention Estimation in a Real World Scenario". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 39–44 (cit. on p. viii).

[16] European Commission. *Fact-finding studies in support of the development of an EU strategy for freight transport logistics*. 2015. URL: https://transport.ec.europa.eu/system/files/2016-09/2015-01-freight-logistics-lot1-logistics-sector.pdf (visited on Feb. 28, 2022) (cit. on p. 1).

[17] European Commission and Eurostat. *Energy, transport and environment statistics : 2020 edition*. Publications Office, 2020 (cit. on p. 1).

[18] European Commission and Eurostat. *The EU in the world : 2020 edition*. Publications Office, 2020 (cit. on p. 1).

[19] International Organization of Motor Vehicle Manufacturers. *Motorization rate 2015 - WORLDWIDE*. 2015. URL: https://www.oica.net/category/vehicles-in-use (visited on Feb. 28, 2022) (cit. on pp. 1, 2).

[20] ODYSEE-MURE proyect. *Number of cars per capita*. 2019. URL: https://www.odyssee-mure.eu/publications/efficiency-by-sector/transport/number-cars-per-capita.html (visited on Feb. 28, 2022) (cit. on p. 1).

[21] ODYSEE-MURE proyect. *Change in distance travelled by car*. 2019. URL: https://www.odyssee-mure.eu/publications/efficiency-by-sector/transport/distance-travelled-by-car.html (visited on Feb. 28, 2022) (cit. on p. 1).

[22] World Health Organization. *Global status report on road safety 2018*. 2018 (cit. on pp. 2, 3).

[23] European Road Safety Observatory. *Annual statistical report on road safety in the EU 2020*. 2020. URL: https://ec.europa.eu/transport/road_safety/statistics-and-analysis/data-and-analysis/annual-statistical-report_en (visited on Feb. 28, 2022) (cit. on p. 2).

[24] Karl Friedrich Benz. "Vehicle with gas engine operation". German Patent Number DRP-37435. 1886 (cit. on p. 3).

[25] Volvo AB. "Three-point seat belt systems comprising two side lower and one side upper anchoring devices". US3043625A. 1962 (cit. on p. 3).

[26] Sadayuki TSUGAWA. "TRENDS AND ISSUES IN SAFE DRIVER ASSISTANCE SYSTEMS: Driver Acceptance and Assistance for Elderly Drivers". In: *IATSS Research* 30.2 (2006), pp. 6–18 (cit. on p. 4).

[27] Anup Doshi and Mohan Manubhai Trivedi. "On the Roles of Eye Gaze and Head Dynamics in Predicting Driver's Intent to Change Lanes". In: *IEEE Transactions on Intelligent Transportation Systems* 10.3 (2009), pp. 453–462 (cit. on p. 5).

[28] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2022 (cit. on p. 6).

[29] Enrique Marti, Miguel Angel de Miguel, Fernando Garcia, and Joshue Perez. "A Review of Sensor Technologies for Perception in Automated Driving". In: *IEEE Intelligent Transportation Systems Magazine* 11.4 (2019), pp. 94–108 (cit. on pp. 6, 18).

[30] On-Road Automated Driving (ORAD) committee. SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Apr. 2021 (cit. on p. 6).

[31] SAE International. *SAE J3016 Visual Chart*. 2021. URL: https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf (visited on Feb. 28, 2022) (cit. on p. 7).

[32] Harald Waschl, Ilya Kolmanovsky, and Frank Willems. "Control strategies for advanced driver assistance systems and autonomous driving functions". In: *Lecture notes in control and information sciences. Springer International, Cham, Switzerland* (2019) (cit. on p. 8).

[33] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. "Predictive Active Steering Control for Autonomous Vehicle Systems". In: *IEEE Transactions on Control Systems Technology* 15.3 (2007), pp. 566–580 (cit. on p. 8).

[34] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints". In: *IEEE Transactions on Vehicular Technology* 66.2 (2017), pp. 952–964 (cit. on p. 8).

[35] T. Hessburg and M. Tomizuka. "Fuzzy logic control for lateral vehicle guidance". In: *IEEE Control Systems Magazine* 14.4 (1994), pp. 55–63 (cit. on p. 8).

[36] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. “A Survey of Deep Learning Applications to Autonomous Vehicle Control”. In: *IEEE Transactions on Intelligent Transportation Systems* PP (Dec. 2019) (cit. on p. 8).

[37] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55 (cit. on p. 8).

[38] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, et al. “A Survey on 3D Object Detection Methods for Autonomous Driving Applications”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pp. 3782–3795 (cit. on p. 8).

[39] Jay S Sevak, Aerika D. Kapadia, Jaiminkumar B. Chavda, Arpita Shah, and Mrugendrasinh Rahevar. “Survey on semantic image segmentation techniques”. In: *2017 International Conference on Intelligent Sustainable Systems (ICISS)*. 2017, pp. 306–313 (cit. on p. 9).

[40] Anh Nguyen and Bac Le. “3D point cloud segmentation: A survey”. In: *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE. 2013, pp. 225–230 (cit. on p. 9).

[41] Xavier Rigoulet. *The Importance of Sensor Fusion for Autonomous Vehicles*. Ed. by Digital Nuage. Dec. 5, 2021. URL: `https://www.digitalnuage.com/the-importance-of-sensor-fusion-for-autonomous-vehicles` (visited on Feb. 28, 2022) (cit. on p. 9).

[42] S. Sand, A. Dammann, and C. Mensing. *Positioning in Wireless Communications Systems*. Wiley, 2014 (cit. on p. 11).

[43] Dean A. Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988 (cit. on p. 15).

[44] D. Pomerleau. “RALPH: rapidly adapting lateral position handler”. In: *Proceedings of the Intelligent Vehicles ’95. Symposium*. 1995, pp. 506–511 (cit. on p. 15).

[45] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer. “Vision and navigation for the Carnegie-Mellon Navlab”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.3 (1988), pp. 362–373 (cit. on p. 15).

[46] M. Xie, L. Trassoudaine, J. Alizon, M. Thonnat, and J. Gallice. “Active and intelligent sensing of road obstacles: Application to the European Eureka-PROMETHEUS project”. In: *1993 (4th) International Conference on Computer Vision*. 1993, pp. 616–623 (cit. on p. 15).

[47] R. Behringer. “The DARPA grand challenge - autonomous ground vehicles in the desert”. In: *IFAC Proceedings Volumes* 37.8 (2004). IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004, pp. 904–909 (cit. on p. 15).

[48] R. Behringer, S. Sundareswaran, B. Gregory, et al. “The DARPA grand challenge - development of an autonomous vehicle”. In: *IEEE Intelligent Vehicles Symposium, 2004*. 2004, pp. 226–231 (cit. on p. 15).

[49] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, et al. “Stanley: The Robot That Won the DARPA Grand Challenge”. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–43 (cit. on p. 16).

[50] Chris Urmson, Joshua Anhalt, Daniel Bartz, et al. “A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain”. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 45–102 (cit. on p. 16).

[51] Deborah Braid, Alberto Broggi, and Gary Schmiedel. “The TerraMax Autonomous Vehicle”. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 129–153 (cit. on p. 16).

[52] Isaac Miller, Sergei Lupashin, Noah Zych, et al. “Cornell University’s 2005 DARPA Grand Challenge Entry”. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 363–405 (cit. on p. 16).

[53] Lars B. Cremean, Tully B. Foote, Jeremy H. Gillula, et al. “Alice: An Information-Rich Autonomous Vehicle for High-Speed Desert Navigation”. In: *The 2005 DARPA Grand Challenge: The Great Robot Race*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 437–482 (cit. on p. 16).

[54] Chris Urmson, Joshua Anhalt, Drew Bagnell, et al. “Autonomous Driving in Urban Environments: Boss and the Urban Challenge”. In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–59 (cit. on p. 17).

[55] Michael Montemerlo, Jan Becker, Suhrid Bhat, et al. “Junior: The Stanford Entry in the Urban Challenge”. In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Ed. by Martin Buehler, Karl Iagnemma, and Sanjiv Singh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 91–123 (cit. on p. 17).

[56] Jeroen Ploeg, Steven Shladover, Henk Nijmeijer, and Nathan van de Wouw. “Introduction to the Special Issue on the 2011 Grand Cooperative Driving Challenge”. In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 989–993 (cit. on p. 17).

[57] Cristofer Englund, Lei Chen, Jeroen Ploeg, et al. “The Grand Cooperative Driving Challenge 2016: boosting the introduction of cooperative automated vehicles”. In: *IEEE Wireless Communications* 23.4 (2016), pp. 146–152 (cit. on p. 17).

[58] Julius Ziegler, Philipp Bender, Markus Schreiber, et al. “Making Bertha Drive—An Autonomous Journey on a Historic Route”. In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20 (cit. on p. 18).

[59] Alberto Broggi, Pietro Cerri, Mirko Felisa, et al. “The VisLab Intercontinental Autonomous Challenge: an extensive test for a platoon of intelligent vehicles”. In: *International Journal of Vehicle Autonomous Systems* 10.3 (2012), pp. 147–164 (cit. on p. 18).

[60] Sharon L Poczter, Luka M Jankovic, et al. “The google car: driving toward a better future?” In: *Journal of Business Case Studies (JBCS)* 10.1 (2014), pp. 7–14 (cit. on p. 18).

[61] Car and Driver. *BMW Level 3 Autonomous Driving Tech Is Coming in 2025*. 2022. URL: https://www.caranddriver.com/news/a39414801/bmw-autonomous-driving-tech-2025/ (visited on Mar. 30, 2022) (cit. on p. 18).

[62] Inside EVs. *Mercedes Is First To Sell A Level 3 Autonomous Vehicle In 2022*. 2022. URL: https://insideevs.com/news/553659/mercedes-level3-autonomous-driving-2022/ (visited on Mar. 30, 2022) (cit. on p. 18).

[63] The Verge. *Volvo confident it can get its ‘unsupervised’ highway driving mode approved in California*. 2022. URL: https://www.theverge.com/2022/1/5/22866190/volvo-ride-pilot-level-3-autonomous-subscription-california (visited on Mar. 30, 2022) (cit. on p. 18).

[64] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino Lo Bianco, and A. Piazzi. “Visual perception of obstacles and vehicles for platooning”. In: *IEEE Transactions on Intelligent Transportation Systems* 1.3 (2000), pp. 164–176 (cit. on p. 19).

[65] Sayanan Sivaraman and Mohan Manubhai Trivedi. “Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.4 (2013), pp. 1773–1795 (cit. on p. 19).

[66] Christian Häne, Torsten Sattler, and Marc Pollefeys. “Obstacle detection for self-driving cars using only monocular cameras and wheel odometry”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 5101–5108 (cit. on p. 19).

[67] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019 (cit. on p. 19).

[68] Carlos Guindel, David Martin, and Jose Maria Armingol. “Fast Joint Object Detection and Viewpoint Estimation for Traffic Scene Understanding”. In: *IEEE Intelligent Transportation Systems Magazine* 10.4 (2018), pp. 74–86 (cit. on p. 19).

[69] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017 (cit. on p. 19).

[70] Michele Mancini, Gabriele Costante, Paolo Valigi, and Thomas A. Ciarfuglia. “Fast robust monocular depth estimation for Obstacle Detection with fully convolutional networks”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4296–4303 (cit. on p. 19).

[71] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. "FastDepth: Fast Monocular Depth Estimation on Embedded Systems". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6101–6108 (cit. on p. 19).

[72] Zhen Li, Yuren Du, Miaomiao Zhu, Shi Zhou, and Lifeng Zhang. "A survey of 3D object detection algorithms for intelligent vehicles development". In: *Artificial Life and Robotics* (2021), pp. 1–8 (cit. on p. 19).

[73] Heiko Hirschmuller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341 (cit. on p. 19).

[74] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. "A Survey on Deep Learning Techniques for Stereo-Based Depth Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 1738–1764 (cit. on p. 19).

[75] F. Tombari, F. Gori, and L. Di Stefano. "Evaluation of stereo algorithms for 3D object recognition". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 990–997 (cit. on p. 19).

[76] G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, and A. Broggi. "Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis". In: *IEEE Transactions on Image Processing* 15.8 (2006), pp. 2364–2375 (cit. on p. 19).

[77] Claudio Caraffi, Stefano Cattani, and Paolo Grisleri. "Off-Road Path and Obstacle Detection Using Decision Networks and Stereo Vision". In: *IEEE Transactions on Intelligent Transportation Systems* 8.4 (2007), pp. 607–618 (cit. on p. 19).

[78] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8 (cit. on p. 19).

[79] Su Pang, Daniel Morris, and Hayder Radha. "CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10386–10393 (cit. on p. 19).

[80] Jorge Beltrán, Carlos Guindel, Irene Cortés, et al. "Towards Autonomous Driving: a Multi-Modal 360° Perception Proposal". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6 (cit. on p. 19).

[81] Bo Li. "3D fully convolutional network for vehicle detection in point cloud". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1513–1518 (cit. on p. 19).

[82] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1887–1893 (cit. on p. 19).

[83] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 (cit. on p. 19).

[84] Özgür Erkent and Christian Laugier. “Semantic Segmentation With Unsupervised Domain Adaptation Under Varying Weather Conditions for Autonomous Vehicles”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3580–3587 (cit. on p. 20).

[85] Aditya Ganeshan, Alexis Vallet, Yasunori Kudo, et al. “Warp-Refine Propagation: Semi-Supervised Auto-Labeling via Cycle-Consistency”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 15499–15509 (cit. on p. 20).

[86] Xinge Zhu, Hui Zhou, Tai Wang, et al. “Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR-based Perception”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1 (cit. on p. 20).

[87] Jianyun Xu, Ruixiang Zhang, Jian Dou, et al. “RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16004–16013 (cit. on p. 20).

[88] José M Álvarez Alvarez and Antonio M. Ĺopez. “Road Detection Based on Illuminant Invariance”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.1 (2011), pp. 184–193 (cit. on p. 20).

[89] Amol Borkar, Monson Hayes, and Mark T. Smith. “A Novel Lane Detection System With Efficient Ground Truth Generation”. In: *IEEE Transactions on Intelligent Transportation Systems* 13.1 (2012), pp. 365–374 (cit. on p. 20).

[90] Markus Schreiber, Carsten Knöppel, and Uwe Franke. “LaneLoc: Lane marking based localization using highly accurate maps”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 449–454 (cit. on p. 20).

[91] Qingquan Li, Long Chen, Ming Li, Shih-Lung Shaw, and Andreas Nüchter. “A Sensor-Fusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios”. In: *IEEE Transactions on Vehicular Technology* 63.2 (2014), pp. 540–555 (cit. on p. 20).

[92] Alberto Hata and Denis Wolf. “Road marking detection using LIDAR reflective intensity data and its application to vehicle localization”. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 584–589 (cit. on p. 20).

[93] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. “Fast LIDAR-based road detection using fully convolutional neural networks”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1019–1024 (cit. on p. 20).

[94] M.A. Garcia-Garrido, M.A. Sotelo, and E. Martin-Gorostiza. “Fast traffic sign detection and recognition under changing lighting conditions”. In: *2006 IEEE Intelligent Transportation Systems Conference*. 2006, pp. 811–816 (cit. on p. 20).

[95] Alberto Broggi, Pietro Cerri, Paolo Medici, Pier Paolo Porta, and Guido Ghisio. “Real Time Road Signs Recognition”. In: *2007 IEEE Intelligent Vehicles Symposium*. 2007, pp. 981–986 (cit. on p. 20).

[96] Hilario Gomez-Moreno, Saturnino Maldonado-Bascon, Pedro Gil-Jimenez, and Sergio Lafuente-Arroyo. “Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.4 (2010), pp. 917–930 (cit. on p. 21).

[97] Nathaniel Fairfield and Chris Urmson. “Traffic light mapping and detection”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5421–5426 (cit. on p. 21).

[98] R. Madhavan and H.F. Durrant-Whyte. “Natural landmark-based autonomous vehicle navigation”. In: *Robotics and Autonomous Systems* 46.2 (2004), pp. 79–95 (cit. on p. 21).

[99] Yongtao Yu, Jonathan Li, Haiyan Guan, Cheng Wang, and Jun Yu. “Semiautomated Extraction of Street Light Poles From Mobile LiDAR Point-Clouds”. In: *IEEE Transactions on Geoscience and Remote Sensing* 53.3 (2015), pp. 1374–1386 (cit. on p. 21).

[100] Federico Tombari, Nicola Fioraio, Tommaso Cavallari, et al. “Automatic detection of pole-like structures in 3D urban environments”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 4922–4929 (cit. on p. 21).

[101] Robert Spangenberg, Daniel Goehring, and Raúl Rojas. “Pole-based localization for autonomous vehicles in urban scenarios”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2161–2166 (cit. on p. 21).

[102] Alexander Schaefer, Daniel Büscher, Johan Vertens, Lukas Luft, and Wolfram Burgard. “Long-term vehicle localization in urban environments based on pole landmarks extracted from 3-D lidar scans”. In: *Robotics and Autonomous Systems* 136 (2021), p. 103709 (cit. on p. 21).

[103] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005 (cit. on p. 21).

[104] Berthold Horn, Hugh Hilden, and Shahriar Negahdaripour. “Closed-Form Solution of Absolute Orientation using Orthonormal Matrices”. In: *Journal of the Optical Society of America A* 5 (July 1988), pp. 1127–1135 (cit. on p. 21).

[105] Jose Luis Blanco. “A tutorial on SE(3) transformation parameterizations and on-manifold optimization”. In: (Sept. 2010) (cit. on p. 21).

[106] Du Q Huynh. “Metrics for 3D rotations: Comparison and analysis”. In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164 (cit. on pp. 22, 34).

[107] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. “A micro Lie theory for state estimation in robotics”. In: *arXiv preprint arXiv:1812.01537* (2018) (cit. on p. 22).

[108] Sebastian Thrun. “Probabilistic Robotics”. In: *Communications of the ACM* 45.3 (Mar. 2002), pp. 52–57 (cit. on pp. 22, 30, 131).

[109] Michael L. Anderson, Kevin M. Brink, and Andrew R. Willis. “Real-Time Visual Odometry Covariance Estimation for Unmanned Air Vehicle Navigation”. In: *Journal of Guidance, Control, and Dynamics* 42.6 (2019), pp. 1272–1288 (cit. on p. 22).

[110] Mostafa Osman, Ahmed Hussein, Abdulla Al-Kaff, Fernando García, and Dongpu Cao. “A Novel Online Approach for Drift Covariance Estimation of Odometries Used in Intelligent Vehicle Localization”. In: *Sensors* 19.23 (2019) (cit. on p. 22).

[111] Sean Campbell, Niall O’Mahony, Anderson Carvalho, et al. “Where am I? Localization techniques for Mobile Robots A Review”. In: *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. 2020, pp. 43–47 (cit. on p. 22).

[112] Johann Laconte, Abderrahim Kasmi, Romuald Aufrère, Maxime Vaidis, and Roland Chapuis. “A Survey of Localization Methods for Autonomous Vehicles in Highway Scenarios”. In: *Sensors* 22.1 (2022) (cit. on p. 22).

[113] R Rao and Automot Eng. “Steering linkage design, A method of determining the configuration of the steering linkage so that the geometry conforms to Ackermann principle”. In: *Automobile Engineer* 58 (1968), pp. 31–33 (cit. on p. 23).

[114] Kok Seng Chong and L. Kleeman. “Accurate odometry and error modelling for a mobile robot”. In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4. 1997, 2783–2788 vol.4 (cit. on p. 23).

[115] Kooktae Lee and Woojin Chung. “Calibration of kinematic parameters of a Car-Like Mobile Robot to improve odometry accuracy”. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 2546–2551 (cit. on p. 23).

[116] Máté Fazekas, Balázs Németh, Péter Gáspár, and Olivier Sename. “Vehicle odometry model identification considering dynamic load transfers”. In: *2020 28th Mediterranean Conference on Control and Automation (MED)*. 2020, pp. 19–24 (cit. on p. 23).

[117] J. Borenstein and L. Feng. “Gyrodometry: a new method for combining data from gyros and odometry in mobile robots”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 1. 1996, 423–428 vol.1 (cit. on p. 23).

[118] Martin Brossard and Silvère Bonnabel. “Learning Wheel Odometry and IMU Errors for Localization”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 291–297 (cit. on p. 23).

[119] Jinglin Shen, David Tick, and Nicholas Gans. “Localization through fusion of discrete and continuous epipolar geometry with wheel and IMU odometry”. In: *Proceedings of the 2011 American Control Conference*. 2011, pp. 1292–1298 (cit. on p. 23).

[120] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 3565–3572 (cit. on p. 24).

[121] Tong Qin, Peiliang Li, and Shaojie Shen. “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020 (cit. on p. 24).

[122] S. Sukkarieh, E.M. Nebot, and H.F. Durrant-Whyte. “A high integrity IMU/GPS navigation loop for autonomous land vehicle applications”. In: *IEEE Transactions on Robotics and Automation* 15.3 (1999), pp. 572–578 (cit. on p. 24).

[123] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte. “The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications”. In: *IEEE Transactions on Robotics and Automation* 17.5 (2001), pp. 731–747 (cit. on p. 24).

[124] Sinpyo Hong, Man Hyung Lee, Ho-Hwan Chun, Sun-Hong Kwon, and J.L. Speyer. “Observability of error States in GPS/INS integration”. In: *IEEE Transactions on Vehicular Technology* 54.2 (2005), pp. 731–743 (cit. on p. 24).

[125] João Paulo Silva do Monte Lima, Hideaki Uchiyama, and Rin-ichiro Taniguchi. “End-to-End Learning Framework for IMU-Based 6-DOF Odometry”. In: *Sensors* 19.17 (2019) (cit. on p. 24).

[126] Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, and Shenghai Yuan. “AbolDeepIO: A Novel Deep Inertial Odometry Network for Autonomous Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2020), pp. 1941–1950 (cit. on p. 24).

[127] Davide Scaramuzza and Roland Siegwart. “Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles”. In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1015–1026 (cit. on p. 24).

[128] Alberto Pretto, Emanuele Menegatti, Maren Bennewitz, Wolfram Burgard, and Enrico Pagello. “A visual odometry framework robust to motion blur”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 2250–2257 (cit. on p. 24).

[129] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems”. In: *IEEE Transactions on Robotics* 33.2 (2017), pp. 249–265 (cit. on p. 24).

[130] Jakob Engel, Vladlen Koltun, and Daniel Cremers. “Direct Sparse Odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2018), pp. 611–625 (cit. on p. 24).

[131] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890 (cit. on p. 24).

[132] Andrew Howard. “Real-time stereo visual odometry for autonomous ground vehicles”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 3946–3952 (cit. on p. 25).

[133] Bernd Kitt, Andreas Geiger, and Henning Lategahn. “Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme”. In: *2010 IEEE Intelligent Vehicles Symposium*. 2010, pp. 486–492 (cit. on p. 25).

[134] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, et al. “Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 340–349 (cit. on p. 25).

[135] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1278–1289 (cit. on p. 25).

[136] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. “DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2043–2050 (cit. on p. 25).

[137] Noha Radwan, Abhinav Valada, and Wolfram Burgard. “VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4407–4414 (cit. on p. 25).

[138] Lu Sun, Junqiao Zhao, Xudong He, and Chen Ye. “DLO: Direct LiDAR Odometry for 2.5D Outdoor Environment”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1–5 (cit. on p. 25).

[139] Huan Yin, Yue Wang, Xiaqing Ding, et al. “3D LiDAR-Based Global Localization Using Siamese Neural Network”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.4 (2020), pp. 1380–1392 (cit. on p. 25).

[140] Bo Zhou, Zhongqiang Tang, Kun Qian, Fang Fang, and Xudong Ma. “A LiDAR Odometry for Outdoor Mobile Robots Using NDT Based Scan Matching in GPS-denied environments”. In: *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. 2017, pp. 1230–1235 (cit. on p. 25).

[141] Martin Velas, Michal Spanel, and Adam Herout. “Collar Line Segments for fast odometry estimation from Velodyne point clouds”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4486–4495 (cit. on p. 25).

[142] Qing Li, Shaoyang Chen, Cheng Wang, et al. “LO-Net: Deep Real-Time Lidar Odometry”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8465–8474 (cit. on p. 26).

[143] Michelle Valente, Cyril Joly, and Arnaud de La Fortelle. “An LSTM Network for Real-Time Odometry Estimation”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1434–1440 (cit. on p. 26).

[144] Ji Zhang and Sanjiv Singh. “LOAM: Lidar Odometry and Mapping in Real-time.” In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9 (cit. on pp. 26, 43).

[145] Tixiao Shan and Brendan Englot. “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4758–4765 (cit. on p. 26).

[146] Jean-Emmanuel Deschaud. “IMLS-SLAM: Scan-to-Model Matching Based on 3D Data”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2480–2485 (cit. on pp. 26, 44).

[147] B. Douillard, A. Quadros, P. Morton, et al. “Scan segments matching for pairwise 3D alignment”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3033–3040 (cit. on p. 26).

[148] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, et al. “SuMa++: Efficient LiDAR-based Semantic SLAM”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4530–4537 (cit. on p. 26).

[149] Steven W. Chen, Guilherme V. Nardari, Elijah S. Lee, et al. “SLOAM: Semantic Lidar Odometry and Mapping for Forest Inventory”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 612–619 (cit. on p. 27).

[150] Ji Zhang and Sanjiv Singh. “Visual-lidar odometry and mapping: low-drift, robust, and fast”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2174–2181 (cit. on p. 27).

[151] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. “LIMO: Lidar-Monocular Visual Odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 7872–7879 (cit. on p. 27).

[152] Haoyang Ye, Yuying Chen, and Ming Liu. “Tightly Coupled 3D Lidar Inertial Odometry and Mapping”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 3144–3150 (cit. on p. 27).

[153] Tixiao Shan, Brendan Englot, Drew Meyers, et al. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5135–5142 (cit. on p. 27).

[154] Frank Neuhaus, Tilman Koß, Robert Kohnen, and Dietrich Paulus. “MC2SLAM: Real-Time Inertial Lidar Odometry Using Two-Scan Motion Compensation”. In: *Pattern Recognition*. Ed. by Thomas Brox, Andrés Bruhn, and Mario Fritz. Cham: Springer International Publishing, 2019, pp. 60–72 (cit. on p. 27).

[155] Nagendra R. Velaga, Mohammed A. Quddus, Abigail L. Bristow, and Yuheng Zheng. “Map-Aided Integrity Monitoring of a Land Vehicle Navigation System”. In: *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2012), pp. 848–858 (cit. on p. 27).

[156] Hao Jing, Yang Gao, Sepeedeh Shahbeigi, and Mehrdad Dianati. “Integrity Monitoring of GNSS/INS Based Positioning Systems for Autonomous Vehicles: State-of-the-Art and Open Challenges”. In: *IEEE Transactions on Intelligent Transportation Systems* (2022), pp. 1–22 (cit. on p. 27).

[157] Giovanni A. Santos, João Paulo C. L. da Costa, Daniel V. de Lima, et al. “Improved localization framework for autonomous vehicles via tensor and antenna array based GNSS receivers”. In: *2020 Workshop on Communication Networks and Power Systems (WCNPS)*. 2020, pp. 1–6 (cit. on p. 27).

[158] Sergio Baselga and Luis García-Asenjo. “GNSS Differential Positioning by Robust Estimation”. In: *Journal of Surveying Engineering* 134.1 (2008), pp. 21–25. eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9453%282008%29134%3A1%2821%29 (cit. on p. 27).

[159] W Stempfhuber and M Buchholz. “A precise, low-cost RTK GNSS system for UAV applications”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXVIII-1/C22 (Sept. 2012) (cit. on p. 27).

[160] Ali Al-Shaery, Shun Zhang, and Chris Rizos. “An enhanced calibration method of GLONASS inter-channel bias for GNSS RTK”. In: *GPS solutions* 17.2 (2013), pp. 165–173 (cit. on p. 27).

[161] Jacek Paziewski and Pawel Wielgosz. “Investigation of some selected strategies for multi-GNSS instantaneous RTK positioning”. In: *Advances in Space Research* 59.1 (2017), pp. 12–23 (cit. on p. 28).

[162] Donald W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441 (cit. on p. 28).

[163] Kokichi Sugihara. “Some location problems for robot navigation using a single camera”. In: *Computer vision, graphics, and image processing* 42.1 (1988), pp. 112–129 (cit. on p. 28).

[164] R. Sim and G. Dudek. “Mobile robot localization from learned landmarks”. In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. Vol. 2. 1998, 1060–1065 vol.2 (cit. on p. 28).

[165] Giovanni Adorni, Stefano Cagnoni, Stefan Enderle, et al. “Vision-based localization for mobile robots”. In: *Robotics and Autonomous Systems* 36.2 (2001). Viewing, Sensing, Coordination and Learning in EuroRoboCup 2000, pp. 103–119 (cit. on p. 28).

[166] I. Loevsky and I. Shimshoni. “Reliable and efficient landmark-based localization for mobile robots”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 520–528 (cit. on p. 28).

[167] Xu Zhong, Yu Zhou, and Hanyu Liu. “Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots”. In: *International Journal of Advanced Robotic Systems* 14.1 (2017), p. 1729881417693489. eprint: https://doi.org/10.1177/1729881417693489 (cit. on p. 28).

[168] Huijuan Zhang, Chengning Zhang, Wei Yang, and Chin-Yin Chen. “Localization and navigation using QR code for mobile robot in indoor environment”. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, pp. 2501–2506 (cit. on p. 29).

[169] Seok-Ju Lee, Girma Tewolde, Jongil Lim, and Jaerock Kwon. “QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument”. In: *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. 2015, pp. 965–970 (cit. on p. 29).

[170] Payam Nazemzadeh, Daniele Fontanelli, David Macii, and Luigi Palopoli. “Indoor Localization of Mobile Robots Through QR Code Detection and Dead Reckoning Data Fusion”. In: *IEEE/ASME Transactions on Mechatronics* 22.6 (2017), pp. 2588–2599 (cit. on p. 29).

[171] M. Ocana, L.M. Bergasa, M.A. Sotelo, J. Nuevo, and R. Flores. “Indoor Robot Localization System Using WiFi Signal Measure and Minimizing Calibration Effort”. In: *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.* Vol. 4. 2005, pp. 1545–1550 (cit. on p. 29).

[172] Chouchang Yang and Huai-rong Shao. “WiFi-based indoor positioning”. In: *IEEE Communications Magazine* 53.3 (2015), pp. 150–157 (cit. on p. 29).

[173] S. Gezici, Zhi Tian, G.B. Giannakis, et al. “Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks”. In: *IEEE Signal Processing Magazine* 22.4 (2005), pp. 70–84 (cit. on p. 29).

[174] Sivanand Krishnan, Pankaj Sharma, Zhang Guoping, and Ong Hwee Woon. “A UWB based Localization System for Indoor Robot Navigation”. In: *2007 IEEE International Conference on Ultra-Wideband*. 2007, pp. 77–82 (cit. on p. 29).

[175] S. Se, D.G. Lowe, and J.J. Little. “Vision-based global localization and mapping for mobile robots”. In: *IEEE Transactions on Robotics* 21.3 (2005), pp. 364–375 (cit. on p. 29).

[176] M. Montemerlo and S. Thrun. “Simultaneous localization and mapping with unknown data association using FastSLAM”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1985–1991 vol.2 (cit. on p. 29).

[177] Sebastian Thrun and Michael Montemerlo. “The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures”. In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 403–429. eprint: `https://doi.org/10.1177/0278364906065387` (cit. on p. 29).

[178] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43 (cit. on pp. 29, 82).

[179] F. Schuster, C. G. Keller, M. Rapp, M. Haueis, and C. Curio. “Landmark based radar SLAM using graph optimization”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 2559–2564 (cit. on p. 29).

[180] Henning Lategahn, Markus Schreiber, Julius Ziegler, and Christoph Stiller. “Urban localization with camera and inertial measurement unit”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 719–724 (cit. on p. 29).

[181] Robert Spangenberg, Daniel Goehring, and Raúl Rojas. “Pole-based localization for autonomous vehicles in urban scenarios”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2161–2166 (cit. on p. 29).

[182] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker. “Improving vehicle localization using semantic and pole-like landmarks”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 13–19 (cit. on p. 29).

[183] Jan-Hendrik Pauls, Benjamin Schmidt, and Christoph Stiller. “Automatic Mapping of Tailored Landmark Representations for Automated Driving and Map Learning”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 6725–6731 (cit. on p. 29).

[184] Fabian Poggenhans, Niels Ole Salscheider, and Christoph Stiller. “Precise Localization in High-Definition Road Maps for Urban Regions”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2167–2174 (cit. on p. 29).

[185] Niklas Stannartz, Jui-Lin Liang, Mirko Waldner, and Torsten Bertram. “Semantic Landmark-based HD Map Localization Using Sliding Window Max-Mixture Factor Graphs”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 106–113 (cit. on p. 30).

[186] Julius Kümmerle, Marc Sons, Fabian Poggenhans, et al. “Accurate and Efficient Self-Localization on Roads using Basic Geometric Primitives”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5965–5971 (cit. on p. 30).

[187] Henning Lategahn and Christoph Stiller. “Vision-Only Localization”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.3 (2014), pp. 1246–1257 (cit. on p. 30).

[188] Gregory Chirikjian and Marin Kobilarov. “Gaussian approximation of non-linear measurement models on Lie groups”. In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 6401–6406 (cit. on p. 30).

[189] J. Josiah Steckenrider. “Simultaneous Estimation and Modeling of Robotic Systems with Non-Gaussian State Belief”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 5403–5409 (cit. on p. 30).

[190] Haoqian Huang, Jiacheng Tang, Bo Zhang, et al. “A Novel Nonlinear Algorithm for Non-Gaussian Noises and Measurement Information Loss in Underwater Navigation”. In: *IEEE Access* 8 (2020), pp. 118472–118484 (cit. on p. 30).

[191] Kichun Jo, Yongwoo Jo, Jae Kyu Suhr, Ho Gi Jung, and Myoungho Sunwoo. “Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3377–3392 (cit. on p. 30).

[192] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45 (cit. on p. 31).

[193] Simon J. Julier and Jeffrey K. Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Ed. by Ivan Kadar. Vol. 3068. International Society for Optics and Photonics. SPIE, 1997, pp. 182–193 (cit. on p. 31).

[194] E.A. Wan and R. Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 2000, pp. 153–158 (cit. on p. 31).

[195] R. Van der Merwe and E.A. Wan. “The square-root unscented Kalman filter for state and parameter-estimation”. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Vol. 6. 2001, 3461–3464 vol.6 (cit. on p. 31).

[196] K.M. Nagpal and P.P. Khargonekar. “Filtering and smoothing in an H/sup infinity / setting”. In: *IEEE Transactions on Automatic Control* 36.2 (1991), pp. 152–166 (cit. on p. 31).

[197] Nicholas Metropolis and S. Ulam. “The Monte Carlo Method”. In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341 (cit. on p. 32).

[198] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*. Vol. 1. 2. Springer (cit. on p. 32).

[199] R.C. Luo, Chih-Chen Yih, and Kuo Lan Su. “Multisensor fusion and integration: approaches, applications, and future research directions”. In: *IEEE Sensors Journal* 2.2 (2002), pp. 107–119 (cit. on p. 32).

[200] Zhangjing Wang, Yu Wu, and Qingqing Niu. “Multi-Sensor Fusion in Automated Driving: A Survey”. In: *IEEE Access* 8 (2020), pp. 2847–2868 (cit. on p. 32).

[201] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. “Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review”. In: *Sensors* 21.6 (2021) (cit. on p. 32).

[202] Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer, and Homayoun Najjaran. “Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review”. In: *Sensors* 20.15 (2020) (cit. on p. 32).

[203] Simon Lynen, Markus W. Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. “A robust and modular multi-sensor fusion approach applied to MAV navigation”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 3923–3929 (cit. on p. 32).

[204] Rafael Toledo-Moreo, Miguel A. Zamora-Izquierdo, Benito Ubeda-Minarro, and Antonio F. Gomez-Skarmeta. “High-Integrity IMM-EKF-Based Road Vehicle Navigation With Low-Cost GPS/SBAS/INS”. In: *IEEE Transactions on Intelligent Transportation Systems* 8.3 (2007), pp. 491–511 (cit. on p. 32).

[205] W. Li and H. Leung. “Constrained unscented Kalman filter based fusion of GPS/INS/digital map for vehicle localization”. In: *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*. Vol. 2. 2003, 1362–1367 vol.2 (cit. on p. 32).

[206] Weide You, Fanbiao Li, Liqing Liao, and Meili Huang. “Data Fusion of UWB and IMU Based on Unscented Kalman Filter for Indoor Localization of Quadrotor UAV”. In: *IEEE Access* 8 (2020), pp. 64971–64981 (cit. on p. 32).

[207] Alexandre Ndjeng Ndjeng, Alain Lambert, Dominique Gruyer, and Sebastien Glaser. “Experimental comparison of Kalman Filters for vehicle localization”. In: *2009 IEEE Intelligent Vehicles Symposium*. 2009, pp. 441–446 (cit. on p. 32).

[208] Junn Yong Loo, Chee Pin Tan, and Surya Girinatha Nurzaman. “H-infinity based Extended Kalman Filter for State Estimation in Highly Non-linear Soft Robotic System”. In: *2019 American Control Conference (ACC)*. 2019, pp. 5154–5160 (cit. on p. 32).

[209] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. “Efficient Multi-Robot Localization Based on Monte Carlo Approximation”. In: *Robotics Research*. Ed. by John M. Hollerbach and Daniel E. Koditschek. London: Springer London, 2000, pp. 153–160 (cit. on p. 32).

[210] Miguel Ángel de Miguel, Fernando García, and José María Armingol. “Improved LiDAR probabilistic localization for autonomous vehicles using GNSS”. In: *Sensors* 20.11 (2020), p. 3145 (cit. on p. 32).

[211] Kichun Jo, Yongwoo Jo, Jae Kyu Suhr, Ho Gi Jung, and Myoungho Sunwoo. “Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3377–3392 (cit. on p. 32).

[212] Hyungjin Kim, Bingbing Liu, Chi Yuan Goh, Serin Lee, and Hyun Myung. “Robust Vehicle Localization Using Entropy-Weighted Particle Filter-based Data Fusion of Vertical and Road Intensity Information for a Large Scale Urban Area”. In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1518–1524 (cit. on p. 33).

[213] F. Caballero, L. Merino, I. Maza, and A. Ollero. “A particle filtering method for wireless sensor network localization with an aerial robot beacon”. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 596–601 (cit. on p. 33).

[214] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. “A Study of Vicon System Positioning Performance”. In: *Sensors* 17.7 (2017) (cit. on p. 33).

[215] Andreas Pfrunder, Paulo V. K. Borges, Adrian R. Romero, Gavin Catt, and Alberto Elfes. “Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2601–2608 (cit. on p. 34).

[216] Oliver Wulf, Andreas Nuchter, Joachim Hertzberg, and Bernardo Wagner. “Ground truth evaluation of large urban 6D SLAM”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 650–657 (cit. on p. 34).

[217] Feng Lu and Evangelos Milios. “Globally consistent range scan alignment for environment mapping”. In: *Autonomous robots* 4.4 (1997), pp. 333–349 (cit. on p. 34).

[218] Rainer Kümmerle, Bastian Steder, Christian Dornhege, et al. “On measuring the accuracy of SLAM algorithms”. In: *Autonomous Robots* 27.4 (2009), pp. 387–407 (cit. on pp. 34, 35, 37).

[219] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012 (cit. on pp. 34, 36, 42, 43).

[220] S. Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380 (cit. on p. 34).

[221] Edwin Olson and Michael Kaess. “Evaluating the performance of map optimization algorithms”. In: *RSS Workshop on Good Experimental Methodology in Robotics*. Vol. 15. 2009 (cit. on p. 35).

[222] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580 (cit. on p. 36).

[223] José-Luis Blanco-Claraco, Francisco-Ángel Moreno-Dueñas, and Javier González-Jiménez. “The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario”. In: *The International Journal of Robotics Research* 33.2 (2014), pp. 207–214. eprint: `https://doi.org/10.1177/0278364913507326` (cit. on p. 36).

[224] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. “The Cityscapes Dataset”. In: *CVPR Workshop on The Future of Datasets in Vision*. 2015 (cit. on p. 36).

[225] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, et al. “Scalability in perception for autonomous driving: Waymo open dataset”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454 (cit. on p. 36).

[226] Zhi Yan, Li Sun, Tomáš Krajník, and Yassine Ruichek. “EU Long-term Dataset with Multiple Sensors for Autonomous Driving”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10697–10704 (cit. on p. 36).

[227] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. “1 year, 1000 km: The Oxford RobotCar dataset”. In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15. eprint: `https://doi.org/10.1177/0278364916679498` (cit. on p. 36).

[228] Will Maddern, Geoffrey Pascoe, Matthew Gadd, et al. “Real-time Kinematic Ground Truth for the Oxford RobotCar Dataset”. In: *arXiv preprint arXiv: 2002.10152* (2020) (cit. on p. 36).

[229] Elisabeth Uhlemann. “Time for Autonomous Vehicles to Connect [Connected Vehicles]”. In: *IEEE Vehicular Technology Magazine* 13.3 (2018), pp. 10–13 (cit. on p. 37).

[230] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. “Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds”. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014, pp. 241–246 (cit. on p. 37).

[231] Hanif Ullah, Nithya Gopalakrishnan Nair, Adrian Moore, et al. “5G Communication: An Overview of Vehicle-to-Everything, Drones, and Healthcare Use-Cases”. In: *IEEE Access* 7 (2019), pp. 37251–37268 (cit. on p. 37).

[232] Andras Kokuti, Ahmed Hussein, Pablo Marín-Plaza, Arturo de la Escalera, and Fernando García. "V2X communications architecture for off-road autonomous vehicles". In: *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2017, pp. 69–74 (cit. on p. 37).

[233] H. Hartenstein and L.P. Laberteaux. "A tutorial survey on vehicular ad hoc networks". In: *IEEE Communications Magazine* 46.6 (2008), pp. 164–171 (cit. on p. 37).

[234] Yasser Toor, Paul Muhlethaler, Anis Laouiti, and Arnaud De La Fortelle. "Vehicle Ad Hoc networks: applications and related technical issues". In: *IEEE Communications Surveys Tutorials* 10.3 (2008), pp. 74–88 (cit. on p. 37).

[235] Hatem Abou-zeid, Farhan Pervez, Abdulkareem Adinoyi, Mohammed Aljlayl, and Halim Yanikomeroglu. "Cellular V2X Transmission for Connected and Autonomous Vehicles Standardization, Applications, and Enabling Technologies". In: *IEEE Consumer Electronics Magazine* 8.6 (2019), pp. 91–98 (cit. on p. 38).

[236] Yasir Mehmood, Farhan Ahmad, Ibrar Yaqoob, et al. "Internet-of-Things-Based Smart Cities: Recent Advances and Challenges". In: *IEEE Communications Magazine* 55.9 (2017), pp. 16–24 (cit. on p. 38).

[237] Chien-Ming Chou, Chen-Yuan Li, Wei-Min Chien, and Kun-chan Lan. "A Feasibility Study on Vehicle-to-Infrastructure Communication: WiFi vs. WiMAX". In: *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. 2009, pp. 397–398 (cit. on p. 38).

[238] José Javier Anaya, Pierre Merdrignac, Oyunchimeg Shagdar, Fawzi Nashashibi, and José E. Naranjo. "Vehicle to pedestrian communications for protection of vulnerable road users". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 1037–1042 (cit. on p. 38).

[239] Hiroki Nishiyama, Thuan Ngo, Shoki Oiyama, and Nei Kato. "Relay by Smart Device: Innovative Communications for Efficient Information Sharing Among Vehicles and Pedestrians". In: *IEEE Vehicular Technology Magazine* 10.4 (2015), pp. 54–62 (cit. on p. 38).

[240] Ahmed Hussein, Fernando García, José María Armingol, and Cristina Olaverri-Monreal. "P2V and V2P communication for Pedestrian warning on the basis of Autonomous Vehicles". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 2034–2039 (cit. on p. 38).

[241] Hao Li and Fawzi Nashashibi. "Multi-vehicle cooperative perception and augmented reality for driver assistance: A possibility to 'see' through front vehicle". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2011, pp. 242–247 (cit. on p. 38).

[242] Seong-Woo Kim, Zhuang Jie Chong, Baoxing Qin, et al. "Cooperative perception for autonomous vehicle control on the road: Motivation and experimental results". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 5059–5066 (cit. on p. 38).

[243] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. “Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019, pp. 514–524 (cit. on p. 38).

[244] Andreas Rauch, Felix Klanner, and Klaus Dietmayer. “Analysis of V2X communication parameters for the development of a fusion architecture for cooperative perception systems”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 685–690 (cit. on p. 39).

[245] Gokulnath Thandavarayan, Miguel Sepulcre, and Javier Gozalvez. “Generation of Cooperative Perception Messages for Connected and Automated Vehicles”. In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 16336–16341 (cit. on p. 39).

[246] Seong-Woo Kim, Wei Liu, Marcelo H. Ang, Emilio Frazzoli, and Daniela Rus. “The Impact of Cooperative Perception on Decision Making and Planning of Autonomous Vehicles”. In: *IEEE Intelligent Transportation Systems Magazine* 7.3 (2015), pp. 39–50 (cit. on p. 39).

[247] Kai Liu, Hock Beng Lim, Emilio Frazzoli, Houling Ji, and Victor C. S. Lee. “Improving Positioning Accuracy Using GPS Pseudorange Measurements for Cooperative Vehicular Localization”. In: *IEEE Transactions on Vehicular Technology* 63.6 (2014), pp. 2544–2556 (cit. on p. 39).

[248] Mohsen Rohani, Denis Gingras, Vincent Vigneron, and Dominique Gruyer. “A New Decentralized Bayesian Approach for Cooperative Vehicle Localization Based on Fusion of GPS and VANET Based Inter-Vehicle Distance Measurement”. In: *IEEE Intelligent Transportation Systems Magazine* 7.2 (2015), pp. 85–95 (cit. on p. 39).

[249] Hao Li and Fawzi Nashashibi. “Cooperative Multi-Vehicle Localization Using Split Covariance Intersection Filter”. In: *IEEE Intelligent Transportation Systems Magazine* 5.2 (2013), pp. 33–44 (cit. on p. 39).

[250] Gloria Soatti, Monica Nicoli, Nil Garcia, et al. “Implicit Cooperative Positioning in Vehicular Networks”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.12 (2018), pp. 3964–3980 (cit. on p. 39).

[251] Pranay Sharma, Augustin-Alexandru Saucan, Donald J. Bucci, and Pramod K. Varshney. “Decentralized Gaussian Filters for Cooperative Self-Localization and Multi-Target Tracking”. In: *IEEE Transactions on Signal Processing* 67.22 (2019), pp. 5896–5911 (cit. on p. 39).

[252] Yulong Huang, Yonggang Zhang, Bo Xu, Zhemin Wu, and Jonathon A. Chambers. “A New Adaptive Extended Kalman Filter for Cooperative Localization”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.1 (2018), pp. 353–368 (cit. on p. 39).

[253] Jiang Liu, Bai-gen Cai, and Jian Wang. “Cooperative Localization of Connected Vehicles: Integrating GNSS With DSRC Using a Robust Cubature Kalman Filter”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.8 (2017), pp. 2111–2125 (cit. on p. 39).

[254] Mariam Elazab, Aboelmagd Noureldin, and Hossam S. Hassanein. “Integrated Cooperative Localization for Connected Vehicles in Urban Canyons”. In: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6 (cit. on p. 39).

[255] Aamir Ahmad, Gian Diego Tipaldi, Pedro Lima, and Wolfram Burgard. “Cooperative robot localization and target tracking based on least squares minimization”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5696–5701 (cit. on p. 39).

[256] J.W. Fenwick, P.M. Newman, and J.J. Leonard. “Cooperative concurrent mapping and localization”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1810–1817 vol.2 (cit. on p. 39).

[257] I. Rekleitis, G. Dudek, and E. Milios. “Probabilistic cooperative localization and mapping in practice”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1907–1912 vol.2 (cit. on p. 39).

[258] Francesco Sottile, Henk Wymeersch, Mauricio A. Caceres, and Maurizio A. Spirito. “Hybrid GNSS-Terrestrial Cooperative Positioning Based on Particle Filter”. In: *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*. 2011, pp. 1–5 (cit. on p. 39).

[259] Gia-Minh Hoang, Benoît Denis, Jérôme Härri, and Dirk T. M. Slock. “Robust data fusion for cooperative vehicular localization in tunnels”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1372–1377 (cit. on p. 39).

[260] G.M. Hoang, B. Denis, J. Härri, and D. T.M. Slock. “Select thy neighbors: Low complexity link selection for high precision cooperative vehicular localization”. In: *2015 IEEE Vehicular Networking Conference (VNC)*. 2015, pp. 36–43 (cit. on pp. 39, 103).

[261] G.M. Hoang, B. Denis, J. Härri, and D. T.M. Slock. “Cooperative localization in GNSS-aided VANETs with accurate IR-UWB range measurements”. In: *2016 13th Workshop on Positioning, Navigation and Communications (WPNC)*. 2016, pp. 1–6 (cit. on pp. 40, 108).

[262] J. Behley, M. Garbade, A. Milioto, et al. “Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset”. In: *The International Journal on Robotics Research* 40.8-9 (2021), pp. 959–967 (cit. on pp. 42–44).

[263] L. Laboshin. *Open-source implementation of Laser Odometry and Mapping (LOAM)*. `https://github.com/laboshinl/loam_velodyne`. 2022 (cit. on p. 43).

[264] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* `https://github.com/MichaelGrupp/evo`. 2017 (cit. on pp. 46, 125, 134).

[265] Hui Zhou, Xinge Zhu, Xiao Song, et al. “Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation”. In: *arXiv preprint arXiv:2008.01550* (2020) (cit. on p. 55).

[266] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, et al. “Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking”. In: *arXiv preprint arXiv:2109.03805* (2021) (cit. on p. 56).

[267] Yingying Chen, John-austen Francisco, Wade Trappe, and Richard P. Martin. “A Practical Approach to Landmark Deployment for Indoor Localization”. In: *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Vol. 1. 2006, pp. 365–373 (cit. on p. 65).

[268] Maximilian Beinhofer, Jörg Müller, and Wolfram Burgard. “Near-optimal landmark selection for mobile robot navigation”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4744–4749 (cit. on p. 65).

[269] Maximilian Beinhofer, Jörg Müller, and Wolfram Burgard. “Effective Landmark Placement for Accurate and Reliable Mobile Robot Navigation”. In: *Robot. Auton. Syst.* 61.10 (Oct. 2013), pp. 1060–1069 (cit. on p. 65).

[270] Kaarthik Sundar, Shriram Srinivasan, Sohum Misra, Sivakumar Rathinam, and Rajnikant Sharma. “Landmark Placement for Localization in a GPS-denied Environment”. In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 2769–2775 (cit. on p. 65).

[271] Lionel Génevé, Olivier Kermorgant, and Édouard Laroche. “Limits of trilateration-based sensor placement algorithms”. In: *2017 IEEE Sensors Applications Symposium (SAS)*. 2017, pp. 1–6 (cit. on p. 65).

[272] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3 (cit. on p. 68).

[273] Morgan Quigley, Ken Conley, Brian Gerkey, et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5 (cit. on p. 69).

[274] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16 (cit. on pp. 69, 116).

[275] Donald W Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441 (cit. on p. 82).

[276] Jonathan Goodman and Jonathan Weare. “Ensemble samplers with affine invariance”. In: *Communications in applied mathematics and computational science* 5.1 (2010), pp. 65–80 (cit. on p. 83).

[277] Raphael Falque, Mitesh Patel, and Jacob Biehl. “Optimizing Placement and Number of RF Beacons to Achieve Better Indoor Localization”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2304–2311 (cit. on p. 85).

[278] Gia-Minh Hoang, Benoît Denis, Jérôme Härri, and Dirk T.M. Slock. “On communication aspects of particle-based cooperative positioning in GPS-aided VANETs”. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, pp. 20–25 (cit. on p. 103).

[279] Han Li, Liuyan Han, Ran Duan, and Geoffrey M. Garner. “Analysis of the Synchronization Requirements of 5g and Corresponding Solutions”. In: *IEEE Communications Standards Magazine* 1.1 (2017), pp. 52–58 (cit. on p. 107).

[280] H. W. Kuhn and Bryn Yaw. “The Hungarian method for the assignment problem”. In: *Naval Res. Logist. Quart* (1955), pp. 83–97 (cit. on p. 108).

[281] N.M. Kwok, Gu Fang, and W. Zhou. “Evolutionary particle filter: re-sampling from the genetic algorithm perspective”. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 2935–2940 (cit. on p. 109).

[282] Somayyeh Sadegh Moghaddasi and Neda Faraji. “A hybrid algorithm based on particle filter and genetic algorithm for target tracking”. In: *Expert Systems with Applications* 147 (2020), p. 113188 (cit. on p. 109).

[283] Thomas Moore and Daniel W. Stouch. “A Generalized Extended Kalman Filter Implementation for the Robot Operating System”. In: *IAS*. 2014 (cit. on p. 134).