كلية العلوم و التقنيات فاس
ⵜⴰⵖⵚⵍⵓⵍⵜ ⵏ ⵜⵇⴰⵙⵙⴰⵢⵉⵏ ⴷ ⵜⴰⵏⵣⵡⴰⵢⵉⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Thesis

Presented by

CHARROUD Anas

To obtain the

Doctorate in Applied Mathematics

Discipline: Mathematics and Informatics

Specialty: Applied Mathematics and Informatics

Entitled : Navigating the Road Ahead: Mathematical Solutions for Self-Driving Vehicle Localization and Mapping

Supervisor

Professor KARIM EL MOUTAOUAKIL

Defended 02-12-2023

Composition of the jury

P. Ali Yahyaouy Président

p. Mohamed Ettaouil Rapporteur & Examinateur

p. Ahmed Aberqi Rapporteur & Examinateur

p. Mohamed Lazaar Rapporteur & Examinateur

p. Jaouad Mekkaoui Examinateur

p. Abdelhai El Azzouzi Examinateur

p. Imad Badi Examinateur

p. Khalid Haddouch Examinateur

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to God for His blessings.

I would also like to thank my supervisor, Dr. Karim El Moutaoukil, for his invaluable guidance, support, and encouragement throughout my Ph.D. journey. Your insightful advice and unwavering belief in me have been instrumental in completing this thesis.

I would also like to thank my co-supervisor, Dr. Ali Yahyaoui, for his invaluable contribution and support. Your expertise and guidance have been truly invaluable.

I'd like to express my gratitude to Prof Vasile Palade for his invaluable guidance and support throughout this journey.

I would like to extend my heartfelt gratitude to the jury members for their time and effort in evaluating my thesis. Your insights and suggestions have been incredibly helpful in shaping this work.

I would like to extend my sincerest thanks to my parents and my brother for their unwavering love and support throughout my studies. Your encouragement and belief in me have been a source of strength and motivation.

I would also like to thank my friends for their support and encouragement during this journey. Particularly, Dr **Bouchra Daoudi**. Your kindness and understanding have been truly invaluable.

Finally, I would like to thank all those who have contributed to the completion of this thesis, including colleagues, lab mates, and academic advisors. Your support and collaboration have been essential in realizing this project.

Thank you all for being a part of my academic journey and for helping me reach this momentous achievement.

TABLE OF CONTENTS

0 Résumé de la thèse **22**

0.1 Introduction 22

0.1.1 Contexte et challenges 22

0.1.2 Objectifs et étendue de cette thèse 25

0.2 Revue de la littérature 26

0.3 Localisation et Cartographie de Véhicules Autonomes basées sur le Regroupement Flou K-moyennes : Une Approche Non-Sémantique 32

0.4 Méthode de Localisation et de Cartographie Rapide et Précise pour les Véhicules Autonomes basée sur un Filtre à Particules de Clustering Modifié 35

0.5 Méthode Rapide de Localisation et de Cartographie basée sur les Filtres à Particules Adaptatifs 40

0.6 XDLL : Méthode de Localisation et de Cartographie pour les Véhicules Autonomes Basée sur l'Apprentissage en Profondeur Expliqué à partir de Données LiDAR 45

0.7 Amélioration de la Localisation LiDAR basée sur l'Autoencodeur dans les Véhicules Autonomes 50

0.8 Conclusion 55

0.8.1 Défis et travaux futurs 56

0.8.1.1 Extraction de caractéristiques 57
0.8.1.2 Cartographie 57
0.8.1.3 Localisation 58

1 Introduction **59**
1.1 Context and challenges 59
1.2 Objectives and Scopes of this thesis 62
1.3 Contributions of this Thesis 63
1.4 Organization of this Thesis 68
1.5 List of publications 69

2 Literature review **70**
2.1 Introduction 70
2.2 Features extraction method for vehicle localization purpose 71
2.2.1 Semantic features 72
2.2.2 Non-Semantic features 75
2.2.3 Deep learning features 77
2.2.4 Discussion 79
2.3 Mapping 81
2.3.1 Offline maps 83
2.3.2 Online maps 86
2.4 Localization 88
2.4.1 Dead Reckoning 88
2.4.2 Triangulation 89
2.4.3 Motion sensors 90
2.4.4 Matching Data 91
2.4.4.1 Fingerprint 92

2.4.4.2 Point cloud matching 92
2.4.4.3 Image matching 93
2.4.5 Optimization-Based approches 96
2.4.5.1 Bundle adjustment based methods 97
2.4.5.2 Graph based methods 98
2.4.6 Probability-Based approches 100
2.4.6.1 Parametric methods -Kalman filter family- 101
2.4.6.2 Non-Parametric methods -Particle filter family- 103
2.4.6.3 Localization with filter-based methods 106
2.4.7 Cooperative localization 106
2.5 Security in Localization 108
2.6 Conclusions 109

I Non semantic features extraction method for localization and mapping purpose 112

3 Localisation and Mapping of Self-driving Vehicles based on Fuzzy K-means Clustering : A Non-semantic Approach 113

3.1 Introduction 113
3.2 Related work 116
3.3 Methodology 118
3.3.1 System Architecture: 118
3.3.2 Component explications: 118
3.4 Results 122
3.4.1 Experiments 122
3.4.2 Discussion 124
3.5 Conclusion 125

II vehicle localization: A stochastic approach 126

4 Fast and Accurate Localization and Mapping Method for Self-Driving Vehicles based on a Modified Clustering Particle Filter 127

4.1 Introduction . . . 127

4.2 Related work . . . 130

4.3 Methodology . . . 133

4.3.1 System Architecture: . . . 133

4.3.2 Mapping . . . 134

4.3.3 Localization: . . . 139

4.3.4 Evaluation . . . 143

4.4 Results . . . 144

4.4.1 Experiments . . . 144

4.4.2 Parameters discussion . . . 145

4.4.3 Short-term localization discussion . . . 145

4.4.4 Long-term localization discussion . . . 149

4.5 Conclusion . . . 150

5 Rapid Localization and Mapping Method Based on Adaptive Particle Filters 153

5.1 Introduction . . . 153

5.2 Related Work . . . 156

5.3 Methodology . . . 158

5.3.1 Features Extraction and Local Maps Creation . . . 161

5.3.2 Localization . . . 162

5.4 Results . . . 168

5.4.1 Experiments . . . 168

5.5 Discussion . . . 170
5.5.1 Parameters discussion: . . . 170
5.5.2 Discussion on Kitti and Pandaset accuracies: . . . 172
5.6 Conclusions and Future Work . . . 174

III vehicle localization: A deep learning approach 177

6 XDLL: Explained Deep Learning LiDAR-Based Localization and Mapping Method for Self-Driving Vehicles 178
6.1 Introduction . . . 178
6.2 Problem Description and Proposed Architecture . . . 181
6.3 Datasets and Parameters . . . 185
6.3.1 Selection of Datasets . . . 185
6.3.2 Data Processing and Parameters' Discussion . . . 186
6.3.3 Training the Model . . . 190
6.3.4 Model Explainability . . . 191
6.4 Results and Discussion . . . 194
6.4.1 How the Model Performs: . . . 194
6.4.2 Why This Output? . . . 196
6.4.3 What Will Happen If We Change That? . . . 198
6.5 Conclusions . . . 200

7 AutoLocal: A Modified Autoencoder-Based Approach to Automated Localization for Self-Driving Vehicles 202
7.1 Introduction . . . 202
7.2 Problem statement: . . . 206
7.3 Convergence of Algorithm 1 . . . 209

7.3.1 Stochastic Normalized Gradient Descent (SNGD): 209
7.3.2 Convergence of Strictly-Locally-Quasi-Convex (SQLC) with SNGD: 210
7.3.3 Main results: 213
7.4 Automatic Localization (AutoLocal) Architecture 220
7.5 Experiments 223
7.5.1 The choice of penalty parameter α: 224
7.5.2 Discussion and comparison 226
7.6 Conclusion: 229

IV Conclusions **231**

8 Conclusion and future directions **232**
8.1 Conclusions 232
8.2 Challenges and future work 233
8.2.1 Features extraction 233
8.2.2 Mapping 234
8.2.3 Localization 234

A Autonomous vehicles equipment **I**

RÉSUMÉ

La localisation et la cartographie des véhicules autonomes constituent un domaine de recherche important visant à permettre aux véhicules autonomes de naviguer avec précision et de comprendre leur environnement. La motivation de ce domaine est de créer des systèmes de transport plus sûrs et plus efficaces. En dotant les véhicules de la capacité à se localiser avec précision et à créer des cartes (maps) détaillées de leur environnement, nous pouvons améliorer leur capacité à naviguer dans des environnements urbains complexes, à éviter les obstacles et à prendre des décisions éclairées en temps réel. Cependant, la localisation et la cartographie précises sont des défis, notamment lorsque le système de positionnement par satellites (GNSS) présente des limites. Bien que le GNSS soit couramment utilisé pour la localisation, il peut présenter des problèmes de précision dans les canyons urbains ou dans des environnements où la visibilité des satellites est limitée. Surmonter ce défi du GNSS est essentiel pour développer des systèmes de conduite autonome robustes et fiables pouvant être utilisés dans des scénarios divers et dynamiques. Par conséquent, les chercheurs s'est tourné vers d'autres capteurs des véhicules tels que le Lidar, les caméras, le radar, etc.

Notre thèse explore le domaine de la localisation et de la cartographie des véhicules autonomes grâce à une revue de littérature approfondie, englobant diverses méthodes et aspects. Nous adoptons une architecture comprenant l'extraction de caractéristiques (features), la cartographie (mapping) et la localisation pour analyser en détail ces méthodes et en extraire leurs points de forces et leurs faiblesses. De plus, nous proposons une approche basée sur le Lidar pour l'extraction efficace et précise des caractéristiques, améliorant ainsi la vitesse et la précision du processus de localisation et de cartographie. En outre, nous explorons deux approches pour la localisation des véhicules : l'approche stochastique et

l'approche d'optimisation. Dans l'approche stochastique, nous apportons deux contributions majeures. Tout d'abord, nous étendons l'algorithme de filtre des particules (particle filter) qui va permetter une localisation rapide et précise du véhicule en sélectionnant des points spécifiques pour la mise à jour des poids à travers une sélection de points sigma. Ensuite, nous améliorons l'algorithme du filtre des particules en ne sélectionnant que les dix particules les plus proches des positions réelles et en régénérant des particules autour d'elles, offrant ainsi un meilleur contrôle et une meilleure durabilité. D'autre part, dans l'approche d'optimisation, nous présentons également deux contributions. Tout d'abord, nous proposons un modèle d'apprentissage profond qui prédit les positions des véhicules en se basant sur les mesures Lidar. Nous expliquons en détail le modèle en utilisant des techniques d'intelligence artificielle expliquable de pointe afin de comprendre les raisons derrière ses prédictions, accompagnées d'une analyse détaillée. Ensuite, nous utilisons un modèle d'autoencodeur modifié pour prédire les positions, en forçonnant l'espace latent (la couche cachée de reseau de neurons) pour approximer les positions en introduisant une fonction de perte spécifique combinant la perte de projection des points Lidar en 3D vers 2D et la perte de positions des points 2D obtenus. La convergence de ce modèle est vérifiée à l'aide de deux théorèmes. Les résultats obtenus par ce modèle ne dépassent pas une erreur de localisation de 0,5 m, ce qui prouve l'efficacité de cette technique.

Mots-clés: **Véhicules autonomes**, **Localisation**, **Cartographie**, **Extraction de caractéristiques**, **Lidar**, **GNSS (Système de Navigation par Satellites)**, **Approche stochastique**, **Approche d'optimisation**, **Apprentissage profond**, **Autoencodeur**, **Rapidité**, **Précision**, **Robustesse**, **Environnements urbains**, **Revue de littérature**, **Méthodologie**, **Avantages et inconvénients**, **Intelligence artificielle expliquable**, **Convergence**

ABSTRACT

Localization and mapping of self-driving vehicles is an important field of research that aims to enable autonomous vehicles to accurately navigate and understand their environment. The motivation for this field is to create safer and more efficient transportation systems. By equipping vehicles with the ability to accurately localize themselves and create detailed maps of their environment, we can improve the ability of autonomous vehicles to navigate complex urban environments, avoid obstacles, and make informed decisions in real time. However, accurately locating and mapping is challenging, especially when the Global Navigation Satellite System (GNSS) has limitations. Although GNSS is commonly used for positioning, it can have accuracy issues in street canyons or in environments with limited satellite visibility. Overcoming this GNSS challenge is critical to developing robust and reliable autonomous driving systems that can be used in diverse and dynamic scenarios. Consequently, the researcher shifted tower other vehicle's sensors equipments e.g Lidar, cameras, Radar, etc.

Our thesis investigates the field of self-driving vehicle localization and mapping through an extensive literature review, encompassing various methods and aspects. We adopt an architecture comprising feature extraction, mapping, and localization to thoroughly analyze these methods and extract their strengths and weaknesses. Additionally, we propose a Lidar-based approach for efficient and accurate feature extraction, enhancing the speed and precision of the localization and mapping process. Furthermore, we explore two approaches for vehicle localization: stochastic and optimization. In the stochastic approach, we make two key contributions. Firstly, we introduce a modified particle filter algorithm that achieves fast and accurate vehicle localization by selecting specific points for weight updating through sigma point selection. Secondly, we enhance the particle filter algorithm

by selecting only the ten closest particles to the real positions and re-generating particles around them, providing better control and sustainability. On the other hand, within the optimization approach, we present two contributions as well. Firstly, we propose a deep learning model that predicts vehicle positions based on Lidar measurements. We extensively explain the model using state-of-the-art Explainability AI techniques to uncover the reasons behind its predictions, accompanied by detailed analysis. Secondly, we employ a modified autoencoder model for position prediction, where we shape the latent space to approximate positions by introducing a specific loss function that combines the projection loss from 3D to 2D Lidar points and the positions loss of the obtained 2D points. The convergence of this model is verified through two theorems. The results obtained by this model do not exceed a localization error of 0.5 m, which proves the effectiveness of this technique.

Keywords: **Self-driving vehicles, Localization, Mapping, Feature extraction, Lidar, GNSS (Global Navigation Satellite System), Stochastic approach, Optimization approach, Deep learning, Autoencoder Fastness, Accuracy, Robustness, Urban environments, Literature review, Methodology, Pros and cons, Explainability AI, Convergence**

LIST OF FIGURES

1.1 Steps to follow to achieve a self-driving vehicle. 60

1.2 Example signal interference from environmental factors 64

2.1 Flowchart of feature extraction 72

2.2 Timeline of 3D object detection algorithms [11]. 77

2.3 Cartesian 2D example scenario. C_1 and C_1 are centers of circles (satellites). P is point of interest (vehicle) with coordinates (x, y) and r_1, r_2 are the distances from the center of circle, respectively from C_1 and C_1, to the desired point [103] 90

2.4 Example of a loop closure [137]. Image (a) shows a failure in loop closure. Image (b) shows a correct loop closure 97

2.5 Illustration of the V2V-V2I-V2P-V2X [161] 108

3.1 Showing the framework of the proposed approach, consisting of three main stages: the feature extraction (step 1), the global mapping (step 2), and the localisation stage (step 3) 119

3.2 Showing a sample of cluster center extracted by Fuzzy k-mean, with the data presented in red, and the center of the cluster in green 120

3.3 Showing a sample of transformation of a concatenating map (above) into a GMM map (bottom). The number of neurons depends on the sequence. 120

4.1 Presentation of the framework used in this work, which consists of two main components, mapping and localization. Mapping is responsible for extracting the features and creating the global map. Localization is responsible for calculating the positions 134

4.2 Example of the features extraction workflow. Here we took K-means as a clustering algorithm (for image clarity it is recommended to zoom in the image (d)) 136

4.3 Illustration of the process of creating the global map of the environment . 138

4.4 Explication of the workflow of the particle filter (the images are not real. They are generated to explain in depth the process.) 140

4.5 Global maps 151

5.1 Workflow architecture describing our proposed method. **(a)** Describing the vehicle's true position (in red), IMU's position (in blue) and objects in the vehicle's environment as picked up by the LiDAR (in blue). **(b)** The structure of the proposed approach consists of two main steps: data processing features extraction (A) + global mapping (B), described within the dark blue rectangle, which is a framework for LiDAR measurements processing and creating local and global maps. Secondly, localization (C) using adaptive particle filters. 159

5.2 Illustration of the feature extraction. 163

5.3 Illustration of the feature extraction. 164

5.4 Flowchart of the feature extraction from LiDAR measurement. 165

5.5 Explanation of the workflow of the particle filter (the images are not real; they are generated to explain the process in depth). (a) Generate particles. (b) Applying IMU transformation. (c) Compare the distances of the particles with the futures of the global map (features in green) and the real distance. (d) Resampling stage, where we regenerate particle around the closest one (particle in-cyan color) 167

5.6 Explanation of the workflow of the particle. Figure (a) shows the results of the mapping phase after concatenation and transformation, where the blue dots represent the global features map, the red triangle is the particle initialization, and the green line is the ground truth. Figure (b) shows an example of data association of scan 50 of the sequence 0001 of the Kitti dataset. 170

5.7 Silhouette values registered by fuzzy K-means algorithm for different cluster numbers in sequences 0001, 0034, 0027, and 0053 from the Kitti dataset. (a) Silhouette values in sequence 0001. (b) Silhouette values in sequence 0034. (c) Silhouette values in sequence 0027. (d) Silhouette values in sequence 0053. 171

5.8 Graphical display of the average absolute position and angular error relative to the vehicle path for each sequence. 175

5.9 Graphical display of the average absolute position and angular error relative to the vehicle path for each sequence. 176

6.1 Representation of the workflow architecture. Feature extraction from LiDAR scans (blue dots) and training the DL model using convolutions of a 1D layer, an LSTM layer, a GRU layer, and a fully connected network layer to estimate the real position (brown dot). 182

6.2 The architecture of the proposed model. 184

6.3 Example—sequence 0001 from Kitti dataset: Blue dots represent the real positions of the vehicle at each timestamp explained with Cartesian coordinates (x and y in meters). 188

6.4 An illustration of the feature-extraction process, which crops the LiDAR scan into a square of 10 m × 10 m and with a height of 0.5 m. Then, a fuzzy k-means algorithm was applied to the cropped points. Figure 6.5 shows clearly how fuzzy k-means is able to extract relevant features for the trajectory. (**a**) LiDAR 3D points scan. (**b**) The cropped scan. (**c**) Fuzzy k-means features extracted (in red) and the ground truth positions (in blue) 189

6.5 Inputs of the proposed model. Each frame contains 6 cluster-based features (five of them are the centers of clusters extracted using the fuzzy k-means algorithm, and the sixth is the mean of the five cluster centers). Each cluster contains 3D points. T is the number of frames. 190

6.6 Graphical representation of the loss variation over epochs. The red curve is for the training loss, and the green curve is for validation. 195

6.7 Mean and punctual impact for all four explainers. (**a**) Mean impact for every sample of the saliency explainer. (**b**) Punctual impact for every sample of the saliency explainer. (**c**) Mean impact for every sample of the integrated gradients explainer. (**d**) Punctual impact for every sample of the integrated gradients explainer. (**e**) Mean impact for every sample of the SmoothGrad explainer. (**f**) Punctual impact for every sample of the SmoothGrad explainer. (**g**) Mean impact for every sample of the VarGrad explainer. (**h**) Punctual impact for every sample of the VarGrad explainer. Punctual impact explains the effect of each cluster's coordinates on our model, such as in images (**b**,**d**,**e**,**h**). On the right side of each image, there is a color bar that shows the contributions of the clusters: purple means low contributions and yellow means high contributions. On the x axis, we have the impact calculated according to each method. 198

6.8 Graphical representation of the loss variation of the explained model. The red curve is for the training loss, and the green curve is for validation. . . 199

7.1 Example of autoencoder architecture. The blue phase is the encoder that transforms/projects the input data to the latent space (yellow box). The green phase sends back the data from the latent space to its original dimension. W_1 and W_2 are the corresponding vectors of weights of the hidden layer and output layer, respectively. 206

7.2 An adaptation of the autoencoder represented in Figure 1, where we have added another phase (dark blue color) to show the optimization process between the latent space and the real positions 208

7.3 An illustration of the learning process inside the model 222

7.4 The results obtained from applying the Bayesian optimization algorithm on three distinct intervals, namely the interval [0,1] (c), the interval [0,0.5] (a), and the interval [0.5,1] (b), are presented. 225

7.5 Graphical representation of training loss values over epochs. 226

7.6 Graphical representation of the model prediction (in orange) comparing with the real position in (in blue) 228

A.1 Graphical representation of training loss values over epochs. I

LIST OF TABLES

1 Summary of existing approaches to solve the localization problem. 58

2.1 Categorization of the state-of-art methods to extract relevant features for localization and mapping purposes. 80

2.2 Commercial companies developing digital mapping solutions for autonomous systems. 85

2.3 A description of some of the attacks that threaten vehicle security systems and possible solutions. 110

3.1 Localisation errors for various sequences categories. 123

3.2 Performance comparison of the proposed technique. The results of Weng et al. also Kümmerle et al. are not straightforwardly similar and are expressed for qualitative analysis only. 124

4.1 Time consumption (mm: ss) to execute the features extraction workflow with the specified clustering methods tested in the sequences 0001 from the Kitti data set [187] 137

4.2 Testing the performance of the method with different numbers of cluster particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs. 145

4.3 Testing the performance of the method with different numbers of particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs. 146

4.4 Accuracy and time consumption of the Localization based on the clustering in the Kitti dataset from different categories sequences. 147

4.5 Accuracy and time consumption of the Localization based on the sigma points in the Kitti dataset from different categories sequences 147

4.6 Accuracy and time consumption of the Localization based on the sigma points in the Pandaset dataset from various sequences 148

4.7 Evaluation of the accuracy of the strategies of Sefati et al. and A. Schaefer and of our methodology. The results of Weng et al. and Kümmerle et al. are not directly comparable and are presented for qualitative analysis only. 148

4.8 Accuracy evaluation of our method with state-of-art registration methods . 150

5.1 Time consumed in executing the feature extraction workflow with the specified clustering methods tested on the sequence 0001 from the Kitti data set [187]. 162

5.2 Kitti localization error obtained in different categories and the time consumption in each sequence. tFeat is the cost of time to calculate the features extraction. tloc is the cost of time to calculate the localization. 172

5.3 Performance comparison of the proposed technique. The results of Weng et al. [29] and Kümmerle et al. [27] are not straightforwardly similar and are expressed for qualitative analysis only. 173

5.4 Accuracy evaluation of our method on the Pandaset dataset. 174

6.1 Comparing several models to choose the best architecture. 184

6.2 The sequences used to train the model. 186

6.3 Time consumption (mm: ss) to execute the feature extraction for different clustering methods tested on the sequence 0001 from the Kitti dataset [**?**]. 190

6.4 Parameter tuning for the model. 191

6.5 Error investigation for short-term trajectories (Kitti dataset). 195

6.6 Error investigation for long-term trajectories (NCLT dataset). 196

6.7 Deletion criterion results. 198

6.8 Insertion criterion results. 199

6.9 Results predictions of the new explained model in short-term trajectories (Kitti dataset). 200

6.10 Predictions of the new explained model for the long-term trajectories (NCLT dataset). 200

7.1 Training Sequences Summary - Frame Count and Trajectory Length 222

7.2 The performance of the model for different values. 226

7.3 Accuracy evaluation of our method with state-of-art registration methods . 227

8.1 Summary of existing approaches to solve the localization problem. 235

CHAPTER 0

RÉSUMÉ DE LA THÈSE

0.1 Introduction

0.1.1 Contexte et challenges

Au cours de la dernière décennie, les véhicules autonomes ont connu des progrès impressionnants, avec des milliers de chercheurs travaillant dans différents laboratoires et entreprises, expérimentant dans divers scénarios environnementaux. Certaines questions qui se posent dans l'esprit de leurs utilisateurs potentiels concernent les avantages offerts par cette technologie et la possibilité de s'y fier ou non. Pour répondre à ces questions, il faut examiner le nombre d'accidents et combien de décès sont enregistrés quotidiennement à cause des véhicules non autonomes. Selon le site officiel de l'Organisation mondiale de la santé, environ 1,3 million de personnes, principalement des enfants et de jeunes adultes, meurent chaque année dans des accidents de véhicule. La plupart de ces accidents (93 %) sont connus pour se produire dans les pays à revenu faible et intermédiaire [1]. Les comportements humains, y compris les infractions au code de la route, la conduite sous l'influence de l'alcool ou de substances psychoactives et les réactions tardives, sont les principales causes d'accidents. Par conséquent, la technologie des véhicules autonomes a été conçue et développée pour remplacer l'interaction humain-véhicule. Malgré certains défis des véhicules autonomes, les enquêtes dans [2] ont indiqué que près de 84,4 % des personnes étaient favorables à l'utilisation de véhicules autonomes à l'avenir. Les statistiques de vente indiquent également que le chiffre d'affaires de ces véhicules était de 54

milliards de dollars en 2021 [3]. Ces résultats démontrent en effet la confiance croissante dans l'utilisation de cette technologie. Il a également été suggéré que cette technologie est capable de réduire la pollution, la consommation d'énergie, la consommation de temps, les taux d'accidents, tout en assurant la sécurité et le confort de ses utilisateurs [4].

La société des ingénieurs de l'automobile, Society of Automotive Engineers (SAE), divise la révolution technique résultant de cette technologie en cinq niveaux. Les deux premiers niveaux ne fournissent pas de nombreux services automatiques et présentent donc un faible potentiel de risque. Le défi commence au niveau trois, adopté par presque toutes les entreprises. En effet, le niveau trois est un travail mixte, où le système est préparé pour éviter les collisions, possède une direction assistée, un freinage, etc. D'autre part, les humains doivent être prudents pour toute décision difficile ou alerte provenant du système. Les niveaux quatre et cinq font référence aux véhicules qui doivent assumer la responsabilité de la conduite sans aucune interaction humaine. Par conséquent, ces niveaux sont plus complexes, surtout si l'on considère à quel point notre environnement peut être dynamique et imprévisible, avec une chance de collision venant de n'importe où [4] a affirmé qu'aucun véhicule n'avait obtenu de résultats robustes dans ces niveaux au moment de la rédaction de cet article.

La création d'un véhicule autonome est une tâche compliquée, où il est nécessaire d'effectuer des étapes prédéfinies de manière cohérente et d'atteindre les objectifs fixés de manière robuste. La principale question qui doit être posée est la suivante : comment les véhicules peuvent-ils fonctionner efficacement sans interaction humaine ? Les véhicules doivent continuellement détecter leur environnement, donc ils doivent être équipés de capteurs précis tels que le GPS, l'IMU, l'odométrie des roues, le Lidar, le radar, les caméras, etc. De plus, un ordinateur performant est nécessaire pour traiter l'immense volume de données collectées et extraire avec précision les informations pertinentes. Ces informations aideront les véhicules à prendre de meilleures décisions et à les exécuter grâce à des unités spécifiquement préparées à cet effet.

Pour parvenir à la conduite autonome, [5] a proposé une architecture qui combine la Localisation, la Perception, la Planification, le Contrôle et la Gestion du Système. La localisation est considérée comme une étape indispensable dans la création de systèmes autonomes, car nous avons besoin d'informations précises sur l'emplacement.

Dans cette thèse, nous nous concentrons sur l'aspect de la localisation et de la cartographie des véhicules autonomes, qui revêt une importance significative dans le déploiement

global et les performances des systèmes autonomes. Une localisation précise permet aux véhicules autonomes de comprendre leur position et leur orientation dans l'environnement environnant, ce qui leur permet de naviguer en toute sécurité et efficacement. La localisation est essentielle pour diverses tâches de conduite autonome, notamment la planification de trajets, qui consiste à trouver l'itinéraire optimal vers une destination souhaitée. Des informations de localisation précises sont nécessaires pour calculer les trajets les plus courts et les plus sûrs, en tenant compte de facteurs tels que les conditions de circulation, les règles de la route et les obstacles potentiels. Sans une localisation précise, les décisions de planification de trajets peuvent être compromises, entraînant des itinéraires inefficaces, des retards, voire des manœuvres dangereuses. Une localisation fiable est également cruciale pour faciliter la communication efficace entre les véhicules autonomes (VA). La localisation permet aux véhicules de déterminer avec précision leurs positions relatives les uns aux autres, ce qui soutient les manœuvres de coopération, telles que les changements de voie, les dépassements ou les mouvements coordonnés aux intersections. En partageant des informations de localisation précises, les VA peuvent coordonner leurs actions, améliorant ainsi la fluidité de la circulation et la sécurité globale..

Traditionnellement, les véhicules autonomes s'appuient sur les Systèmes de Navigation par Satellite (GNSS) pour la localisation. Cependant, en raison des problèmes graves associés au GNSS, il est nécessaire d'explorer d'autres approches qui utilisent d'autres capteurs du véhicule. Bien que le GNSS offre une méthode économique pour déterminer la position, il est confronté à des défis tels que le blocage du signal, des erreurs dépassant 3 mètres, des interférences multipath et des limitations dans les environnements urbains et intérieurs. De plus, le GNSS est vulnérable à la dégradation du signal, au brouillage et à la falsification, compromettant la fiabilité et la sécurité des véhicules autonomes.

La principale contribution de cette thèse est de surmonter ces limitations en privilégiant l'utilisation d'autres capteurs du véhicule, tels que les caméras, le LiDAR, le radar et les capteurs inertiels. Ces capteurs offrent la possibilité d'améliorer la précision, la fiabilité et la sécurité dans des environnements difficiles, notamment les zones urbaines, les espaces intérieurs et les régions à couverture satellite limitée. En intégrant les données de plusieurs capteurs, les véhicules autonomes peuvent atteindre une localisation robuste, réduisant la dépendance au GNSS et améliorant les performances globales dans différentes conditions de conduite [7].

0.1.2 Objectifs et étendue de cette thèse

Comme mentionné précédemment, il existe des cas où le Système de Navigation par Satellite Global (GNSS) échoue à localiser avec précision les véhicules autonomes (VA) en raison d'obstacles obstruant la ligne de mire entre les véhicules et les satellites. Ce problème perturbe le fonctionnement des VA et peut avoir des conséquences graves, y compris des accidents potentiels, car le véhicule perd sa capacité à percevoir avec précision sa position. Par conséquent, le véhicule devient incapable d'exécuter correctement des tâches telles que la planification de trajets, la navigation autour d'autres véhicules, voire le maintien du bon côté de la route.

Les chercheurs ont rencontré ce problème et ont proposé différentes méthodes et approches qui reposent sur différents équipements de véhicules, tels que les caméras, le LiDAR, le radar, les systèmes de navigation inertielle (INS) et l'odométrie des roues. Notre objectif dans cette thèse sera de fusionner les informations provenant de différents équipements de capteurs. Cependant, certaines méthodes se basent sur un seul capteur pour la localisation du véhicule, comme l'utilisation de la navigation à l'estime avec une unité de mesure inertielle (IMU) ou l'odométrie des roues. Néanmoins, les méthodes traditionnelles présentent des limites car elles reposent uniquement sur une source d'information, et leurs performances dépendent relativement de la structure de l'environnement.

Traiter plusieurs sources d'informations nécessite un prétraitement des données, qui inclut la manière dont nous convertissons les informations reçues en une entrée utile et comment les fusionner en une méthode unique. En particulier, de vastes informations telles que les mesures LiDAR ou de caméra nécessitent un processus d'extraction de caractéristiques afin d'extraire uniquement ce qui est utile et/ou suffisant pour le déploiement efficace de la méthode.

Basé sur la discussion ci-dessus, notre focus dans cette thèse impliquera l'étude de :

- Réaliser un examen approfondi de l'état de l'art englobant les méthodes d'extraction de caractéristiques, de cartographie et de localisation. Cela inclut une analyse approfondie et une évaluation des approches et techniques existantes dans le domaine de la localisation des VA, en mettant l'accent sur les méthodologies d'extraction de caractéristiques, les algorithmes de cartographie et les méthodes de localisation. En synthétisant et en évaluant de manière critique l'état actuel de la recherche, cette

thèse vise à fournir une compréhension globale des avancées, des défis et des opportunités dans ce domaine, guidant ainsi le développement de techniques novatrices et efficaces de localisation des VA.

- Développer et évaluer des méthodes de pointe pour l'extraction rapide et précise de caractéristiques à partir des capteurs de véhicules. L'accent sera mis sur l'incorporation des dernières avancées en matière de technologie de capteurs pour améliorer le processus d'extraction d'informations pertinentes.

- Explorer et utiliser des techniques stochastiques et probabilistes pour améliorer la localisation des véhicules autonomes (VA). Cela implique d'étudier des modèles et algorithmes probabilistes capables de gérer efficacement les incertitudes et les variations des mesures de capteurs pour une localisation robuste et fiable des VA.

- Examiner les approches et algorithmes d'optimisation pour améliorer la localisation des VA. Cela inclut l'exploration de techniques d'optimisation telles que l'optimisation non linéaire, l'optimisation convexe ou les algorithmes métaheuristiques pour améliorer la précision et l'efficacité des algorithmes de localisation des VA.

- Examiner l'application de techniques d'apprentissage automatique pour la localisation des VA. Cela implique d'explorer l'utilisation d'algorithmes d'apprentissage automatique tels que l'apprentissage profond ou l'apprentissage par renforcement, pour exploiter de grandes quantités de données de capteurs afin d'améliorer les performances de localisation des VA.

- Réaliser une investigation approfondie des théories et principes mathématiques sous-jacents aux méthodes proposées. Cela inclut l'analyse des fondements mathématiques, la compréhension des bases théoriques et l'évaluation de l'applicabilité des concepts mathématiques dans le contexte de la localisation des VA.

0.2 Revue de la littérature

Notre revue de la littérature est déjà rédigée sous la forme d'un article de synthèse intitulé "Localisation et Cartographie pour les Véhicules Autonomes : Une Revue".

- **Résumé et Contenu** La prolifération rapide des véhicules autonomes dans l'industrie automobile a révolutionné les expériences de conduite et résolu des défis complexes

de navigation. Atteindre de telles capacités nécessite une attention technologique soutenue et l'exécution sans faille de tâches complexes, dont l'une consiste en une localisation et une cartographie robustes. Les études récentes n'ont pas offert de description significative et complète dans leur étude des approches actuelles dans ce domaine. Par conséquent, cette revue vise à fournir une couverture adéquate des problèmes des véhicules autonomes dans ce domaine, en examinant les méthodes les plus récentes pour la cartographie et la localisation ainsi que les problèmes connexes d'extraction de caractéristiques et de sécurité des données.

Le discours commence par un examen des techniques modernes d'extraction de caractéristiques pertinentes à partir des véhicules équipés de capteurs, les classant en méthodes sémantiques, non sémantiques et d'apprentissage profond. Il est évident que des considérations telles que la représentativité, la rentabilité et l'accessibilité jouent des rôles essentiels dans la sélection des méthodes pour les tâches de localisation et de cartographie. L'étude se concentre sur les méthodes de construction de cartes environnementales pour les véhicules, englobant à la fois des solutions commercialement disponibles et des solutions développées académiquement. L'analyse distingue entre les environnements connus et inconnus, proposant des solutions adaptées à chaque scénario.

L'enquête explore différentes approches de localisation des véhicules, les classifiant en fonction de leurs caractéristiques mathématiques et de leurs priorités. Chaque segment se conclut en identifiant les défis associés et en esquissant d'éventuelles orientations futures. L'examen se termine en mettant en évidence les défis potentiels en matière de sécurité dans les véhicules autonomes et en évaluant les mécanismes de défense possibles pour contrer les atteintes à la sécurité.

De plus, l'examen explore l'impact des véhicules autonomes sur la pollution sonore et atmosphérique en milieu urbain. En optimisant les modèles de conduite, les voitures autonomes ont le potentiel de réduire les émissions sonores et les polluants atmosphériques en atténuant la congestion routière et en favorisant une accélération et une décélération efficaces. De plus, l'intégration de véhicules autonomes électriques et hybrides pourrait encore réduire les émissions et les niveaux de bruit, compte tenu de leurs caractéristiques respectueuses de l'environnement. L'examen souligne le potentiel d'efforts collaboratifs pour exploiter l'impact positif des véhicules autonomes sur la qualité sonore et atmosphérique urbaine.

Nous proposons dans ce travail une étude qui présente les principales contributions

suivantes :

- Une revue complète des méthodes d'extraction de caractéristiques dans le domaine de la conduite autonome.
- Fournir des détails clairs sur les solutions pour créer une carte pour les véhicules autonomes, en examinant les solutions pré-construites, qui peuvent être des cartes commerciales préexistantes ou générées par des solutions académiques, ou la solution de carte en ligne créée en temps réel avec le processus de positionnement.
- Fournir des informations de contexte et démontrer de nombreuses solutions pour résoudre les problèmes de localisation et de cartographie existants dans la littérature.
- Analyser et passer en revue les approches de localisation présentes dans la littérature, ainsi que les solutions associées pour garantir la sécurité des données.
- Explorer l'empreinte environnementale des véhicules autonomes, dévoilant les impacts et les solutions.

- **Résultats**

  Features extraction: Le Tableau 2 présente une catégorisation des articles examinés dans cette section. Les articles ont été regroupés en fonction des caractéristiques extraites. Nous avons observé que les articles extraient trois types de caractéristiques de l'environnement : des caractéristiques verticales, horizontales ou des courbes de route dans le cas du type sémantique. De plus, les articles ont utilisé des méthodes non sémantiques et d'apprentissage en profondeur pour représenter tout type d'objets présents dans l'environnement ou pour ne représenter qu'une partie d'entre eux. Par ailleurs, le tableau présente certaines méthodes et concepts utilisés pour extraire les caractéristiques. Nous avons analysé la robustesse des caractéristiques extraites pour aider aux tâches de localisation et de cartographie en utilisant trois critères déduits de notre enquête approfondie.

  - **Temps et coût énergétique** : faciles à détecter et à utiliser en termes de compilation et d'exécution.
  - **Représentativité** : détection de caractéristiques fréquemment présentes dans l'environnement pour assurer le processus de correspondance.
  - **Accessibilité** : faciles à distinguer de l'environnement.

Nous avons utilisé la colonne 'Robustesse' comme score attribué à chaque groupe d'articles. Le score est calculé en fonction des trois critères ci-dessus et de l'analyse des expériences dans les articles. Selon le même tableau, l'extraction de caractéristiques non sémantiques obtient le score de robustesse le plus élevé en ce qui concerne leur capacité à représenter l'environnement même avec moins de texture, c'est-à-dire dans le cas de quelques objets dans l'environnement, comme dans le désert. Cette compétence est due à la manière dont les caractéristiques sont extraites. Ces méthodes ne se limitent pas à l'extraction d'un seul type d'objet. Cependant, la carte créée par ces caractéristiques n'aura pas de sens. elles ne sont que des points de référence utiles pour le processus de localisation uniquement.

D'autre part, l'utilisation de caractéristiques sémantiques permet d'obtenir un score passable à utiliser dans les tâches de localisation, car elles consomment un peu plus de temps et d'énergie pour les exécutions car la plupart du temps, elles ne sont pas isolées dans l'environnement. De plus, elles ne peuvent pas être trouvées dans n'importe quel environnement, ce qui affecte difficilement le processus de localisation. Malgré tous ces points négatifs, ces techniques réduisent efficacement la quantité considérable de données de points (LiDAR ou caméra) par rapport aux caractéristiques non sémantiques. De plus, ces caractéristiques peuvent être utilisées pour d'autres tâches de perception.

Les méthodes d'apprentissage en profondeur obtiennent également un score passable en ce qui concerne leur efficacité pour représenter l'environnement. Comme les techniques non sémantiques, l'approche DL garantit la représentativité des caractéristiques dans tous les environnements. Cependant, les méthodes consomment beaucoup de temps et de ressources computationnelles pour être exécutées.

Cartographie

Les cartes hors ligne (également appelées cartes a priori ou pré-construites) sont générées à l'avance pour aider les systèmes de conduite autonome dans les tâches de navigation et de localisation. Selon [79], le chiffre d'affaires résultant de leur utilisation a atteint 1,4 milliard de dollars en 2021 et devrait atteindre 16,9 milliards de dollars d'ici 2030. Le nombre d'investissements dans ce domaine a tendance à augmenter chaque jour, et la concurrence entre les fabricants de cartes est à un

niveau élevé. Ces entreprises utilisent différentes technologies et stratégies de partenariat et de collaboration. Ces dernières années, les grands fabricants automobiles ont accéléré les efforts pour produire de nouvelles générations de systèmes d'aide à la conduite avancés (ADAS). Par conséquent, le développement de telles cartes de haute qualité est indispensable pour ces véhicules.

Les cartes sont sujettes à des erreurs provenant de diverses sources, notamment les changements dans la réalité, les erreurs de localisation, les cartes imprécises, les erreurs de mise à jour des cartes, etc. Nous pouvons définir une métrique de précision pour mesurer à quel point la carte correspond au monde réel. Selon [84], deux métriques de précision sont définies :

- **Précision globale** : (Précision absolue) position des éléments par rapport à la surface de la terre.
- **Précision locale** : (Précision relative) position des éléments par rapport aux éléments environnants sur la route.

Localisation

In our exploration of the localization aspects of self-driving vehicles, we have delved into various methodologies to enhance their positioning accuracy. These methods encompass both traditional and advanced techniques. Dead Reckoning, which involves continuously updating position based on previously known information, has been examined as a fundamental method. Additionally, the utilization of Triangulation and Motion sensors has been explored to accurately determine the vehicle's position. We have further investigated data matching techniques such as Fingerprint, Point cloud matching, and Image matching to align real-world observations with map data. Moreover, we have analyzed Optimization-Based approaches, including Bundle Adjustment and Graph-based methods, which optimize the vehicle's position estimation using intricate mathematical frameworks. Probability-Based approaches have also been covered, with a focus on Parametric methods like the Kalman filter family, which incorporate probabilistic models for accurate localization. Lastly, Cooperative localization methods, enabling vehicles to share information and improve their positioning collectively, have been examined. Overall, our exploration has encompassed a wide array of methodologies, spanning from traditional techniques

to cutting-edge approaches, to enhance the precision and reliability of self-driving vehicle localization.

- **Conclusion**

  Dans cette enquête, nous avons discuté et mis en avant les termes suivants : extraction de caractéristiques, cartographie et traitement de la localisation afin d'assurer une localisation et une navigation sécurisées.

  Les caractéristiques représentent les repères de la carte ou, plus précisément, ce que nous souhaitons que les véhicules voient dans l'environnement (poteaux, trottoirs, bâtiments, intensité lumineuse). Plus les caractéristiques sont pertinentes, plus la cartographie est précise, et par conséquent, la localisation du véhicule est plus précise. Cette enquête présente une nouvelle classification comprenant les approches sémantiques, non sémantiques et d'apprentissage en profondeur pour faciliter la compréhension de ces méthodes. Notre enquête a conclu que l'utilisation de caractéristiques non sémantiques est plus adaptée pour résoudre les tâches de localisation et de cartographie. Nous avons également ajouté que les caractéristiques sémantiques nécessitent du temps pour être extraites de l'environnement et ne peuvent pas exister dans tous les scénarios environnementaux, ce qui peut corrompre le processus de correspondance. Les méthodes d'apprentissage en profondeur sont d'excellents outils pour représenter l'environnement de manière sémantique ou non, selon l'approche utilisée. Bien que les performances soient élevées, la forte consommation de calcul rend leur exécution un peu plus difficile.

  De plus, la phase de cartographie, où une carte est créée en fonction des caractéristiques restantes, a également été examinée. Différentes techniques ont été étudiées, notamment les techniques pré-construites et en ligne, avec les intuitions mathématiques sous-jacentes. Nous avons constaté que les mises à jour des cartes doivent être effectuées fréquemment pour corriger les changements dans l'environnement. De plus, le choix des bonnes caractéristiques à inclure dans la carte est remis en question, car nous devons utiliser un minimum de caractéristiques pour chaque scène sans affecter le stockage en mémoire.

  Le dernier paramètre qui doit être démystifié est la localisation, où nous avons essayé de rassembler et de présenter, autant que possible, des méthodes visant à localiser les véhicules en fonction de différents capteurs. Les approches basées sur

la probabilité, l'optimisation, voire la localisation coopérative, sont plus robustes et fournissent des résultats précis par rapport à l'approche classique. Chaque approche a été brièvement décrite et certains articles récents liés à la littérature ont été étudiés. Enfin, nous avons passé en revue certaines des méthodes d'attaque susceptibles de menacer les systèmes de véhicules autonomes en général.

0.3 Localisation et Cartographie de Véhicules Autonomes basées sur le Regroupement Flou K-moyennes : Une Approche Non-Sémantique

- **Résumé** La localisation et la cartographie sont cruciales pour les véhicules autonomes, car elles fournissent au véhicule des informations précises sur sa position dans son environnement ainsi que sur les infrastructures pertinentes au sein de l'environnement identifié. Cet article démontre la capacité des caractéristiques non sémantiques à représenter les nuages de points et à les utiliser pour décrire l'environnement. Notre architecture proposée utilise l'approche des K-moyennes floues pour extraire des caractéristiques à partir des scènes LiDAR afin de réduire la carte des caractéristiques et de garantir que les caractéristiques soient identifiables dans chaque image. Ensuite, la cartographie globale est réalisée à l'aide du Modèle de Mélange Gaussien (GMM) pour faciliter l'association des données entre les images à cartographier, ce qui aide à effectuer avec précision les tâches de localisation à l'aide du filtre à particules. Les performances de la technique proposée sont comparées à d'autres méthodes de pointe sur différentes séquences de l'ensemble de données brut Kitti, avec des structures environnementales différentes, des conditions météorologiques et des changements saisonniers. Les résultats obtenus démontrent la supériorité de l'approche proposée en termes de vitesse et de représentativité des caractéristiques nécessaires à la localisation en temps réel. De plus, nous avons atteint des précisions compétitives par rapport à d'autres méthodes de littérature.

- **Contenu** Dans cette recherche, nous intégrons une autre catégorie de méthodes qui combinent les entrées de différents capteurs pour effectuer la localisation. Le filtre à particules (PF) est l'une de ces approches. Le PF présente plusieurs avantages qui soutiennent son utilisation pour la localisation. Ces avantages incluent la capacité du PF à gérer les trajectoires non linéaires [86], à suivre plusieurs objets variables

dans le temps [181], à gérer les occultations et les chevauchements [181]. De plus, il ne fait pas d'hypothèses préalables comme la distribution gaussienne des données dans l'approche du filtre de Kalman [181]. Le PF a été utilisé à ses débuts pour le traitement du signal afin d'estimer les états en fonction des variables observées [188]. Ensuite, il a été étendu pour résoudre certains problèmes liés aux systèmes dynamiques, y compris la tâche de localisation, et il a obtenu d'excellents résultats avec une précision de l'ordre de quelques centimètres [26], [27], [29], [31].

Pratiquement, le PF (également connu sous le nom de Monte Carlo séquentiel) fait l'hypothèse de Markov, un concept qui relie l'état (ou la position dans notre cas) X_t à l'état précédent X_{t-1}. L'algorithme commence par générer plusieurs milliers de particules qui représentent les candidats à la position. Chaque particule se voit attribuer un poids. Une valeur de poids élevée signifie que la particule a plus de possibilités d'être la position du véhicule.

Le PF offre l'opportunité d'explorer les avantages des données provenant de sources multiples. Les capteurs de mouvement indiquent aux particules la transformation (translation et rotation) de la dernière position à la position actuelle. Nous utilisons ensuite la mesure du capteur pour vérifier si les distances réelles aux objets sont similaires à celles que nous obtenons de nos mesures. Les particules avec de petites distances ont plus de chances d'être sélectionnées pour représenter la position du véhicule et ont des valeurs de poids élevées. Cependant, la quantité énorme de données que nous recevons des collecteurs complique ce processus, ce qui nous oblige à créer une couche d'extraction de caractéristiques pour réduire le temps et le coût de cette comparaison. Nous proposons pour la première fois la technique de regroupement flou K-moyennes pour extraire des caractéristiques pertinentes et regrouper des points de données similaires, ce qui facilite la recherche efficace de similarités dans les associations de données.

Les principales contributions de cette étude sont résumées comme suit :

- Exploration du pouvoir des caractéristiques non sémantiques pour réduire les caractéristiques de la carte en utilisant l'algorithme des K-moyennes floues. Les clusters restants sont plus représentatifs, en particulier dans les environnements moins texturés. De plus, ils sont rapides à extraire et faciles à utiliser pour les tâches d'association de données.
- Une méthode rapide pour créer une carte globale en utilisant le modèle de

mélange gaussien (GMM) et simplifie la tâche de localisation en se basant sur un filtre à particules modifié.

- **Résultats** Les résultats présentés dans le Tableau 1 montrent que la méthode proposée est capable d'effectuer rapidement les tâches de localisation et de cartographie, et d'atteindre une excellente précision même dans de grandes séquences d'images. Nos conclusions sur l'utilisation des caractéristiques non sémantiques pour représenter les environnements étendent le travail de l'approche proposée par Hungar et al. appelée GRAIL [39]. Hungar et al. ont comparé les caractéristiques des résultats de GRAIL avec douze formes [1] qui sont considérées comme insuffisantes pour couvrir tous les changements possibles dans l'environnement. La méthode de GRAIL a obtenu une erreur de position absolue moyenne de $1.11 + / - 0.41(m)$ et une erreur de rotation de $0.10 + / - 0.19$ (°) sur 2433 scans, ce qui n'est pas fourni par les auteurs pour effectuer une comparaison exacte.

  Dans notre cas, nous avons pris 1902 scans, et nous avons obtenu une erreur absolue moyenne de 1.079 (m) (calculée à partir du Tableau 1)[2] et 0.22 (°) dans les erreurs de rotation. Les K-moyennes floues accélèrent considérablement l'association de données entre les caractéristiques locales et les caractéristiques correspondantes sur la carte globale. Notre méthode proposée a également été comparée à celle de Kümmerle et al. [27], Weng et al. [29], et Sefati et al. [26], ainsi que Schaefer [31]. Selon le Tableau 2, notre méthode surpasse les autres en termes d'erreur de cap. Cependant, la méthode de Schaefer [31] surpasse notre approche en termes d'erreur de position absolue car l'association de données est plus précise comme le montre [31]. Néanmoins, la mise en œuvre de caractéristiques de localisation non sémantiques est plus avantageuse car elle permet d'atteindre une précision compétitive en moins de temps, elle est plus représentative et fonctionne même dans des environnements moins texturés.

- **Conclusion** Ce document introduit une méthode de localisation et de cartographie pour les véhicules autonomes, fondée sur l'extraction de caractéristiques non sémantiques. Nous avons utilisé les K-moyennes floues pour découvrir des motifs au sein des scans LiDAR. Les centroïdes de cluster restants ont été utilisés pour créer les caractéristiques de la carte locale. De plus, la carte globale a été créée en se basant sur un modèle de mélange gaussien (GMM), ce qui facilite l'association des

[1] Ces formes représentent les trajectoires possibles du véhicule.

[2] Nous avons calculé les erreurs moyennes pour l'ensemble des séquences

données. Le processus de localisation a été réalisé à l'aide d'un filtre à particules. Notre approche proposée a obtenu des résultats compétitifs en termes de précision, mais a réussi à le faire en un temps nettement inférieur par rapport aux méthodes de pointe évaluées sur les références de l'ensemble de données Kitti.

En effet, notre méthode novatrice non seulement a obtenu des résultats comparables en termes de précision par rapport aux méthodes actuelles, mais elle a également démontré un avantage significatif en termes de temps d'exécution. Cette performance améliorée en matière de temps d'exécution est particulièrement cruciale dans le contexte des véhicules autonomes, où la réactivité et la vitesse de traitement des données sont essentielles pour garantir une conduite en toute sécurité. En intégrant les avantages des K-moyennes floues pour l'extraction de caractéristiques locales et du modèle de mélange gaussien pour la création de la carte globale, notre approche offre une solution globale efficace pour la localisation et la cartographie dans des environnements variés.

De plus, en utilisant le filtre à particules pour la localisation, notre méthode tient compte des incertitudes et des variations dans les mesures de capteurs, ce qui améliore la robustesse et la fiabilité de la localisation. La combinaison de ces différentes étapes de la méthode permet de créer une approche holistique et performante pour la localisation et la cartographie dans des scénarios réels et variés. En conséquence, notre méthode non seulement améliore les performances globales de localisation, mais elle offre également une base solide pour des développements futurs et des améliorations continues dans le domaine des véhicules autonomes.

0.4 Méthode de Localisation et de Cartographie Rapide et Précise pour les Véhicules Autonomes basée sur un Filtre à Particules de Clustering Modifié

- **Résumé** Les systèmes de conduite autonome requièrent la création de perception, ce qui signifie qu'ils doivent apprendre à interagir avec leur environnement, recueillir des informations et accomplir des tâches pendant la conduite. La localisation et la cartographie sont des concepts essentiels que tout véhicule autonome doit percevoir, car ils fournissent des informations sur la position et la distance entre les objets. L'absence de GPS, par exemple, dans les tunnels et les canyons urbains, rend

nécessaire le développement de méthodes robustes basées sur d'autres équipements du véhicule, tels que le Lidar, les caméras, l'IMU, ou leur combinaison. Basé sur les mesures du Lidar, notre travail présente une architecture composée de deux phases principales de cartographie et de localisation ; la phase de cartographie crée une carte globale de l'environnement basée sur des caractéristiques non sémantiques en utilisant un algorithme de fuzzy c-means plutôt qu'un algorithme sémantique comme cela a été fait dans certaines œuvres de pointe. De plus, les clusters restants ont été filtrés à l'aide de l'algorithme DBSCAN. La phase de localisation adoptée ici suit l'architecture du filtre à particules ; mise à jour du mouvement, mise à jour de la mesure et rééchantillonnage pour estimer les positions. La principale contribution de ce travail est la nouvelle extension de la sélection de particules pour réduire le temps de calcul et maintenir la fiabilité à long terme de la localisation. Nous avons exposé une méthode pour sélectionner des particules pertinentes après les mises à jour de mouvement, basée sur deux approches : le regroupement avec l'algorithme k-means et l'algorithme des points sigma, qui ont été minutieusement examinés sur de courtes séquences de l'ensemble de données Kitti pour découvrir le meilleur. L'approche sélectionnée a été rigoureusement testée sur l'ensemble de données Pandaset. De plus, nous avons testé notre méthode sur un ensemble de données de longue séquence et l'avons comparée aux méthodes les plus récentes. L'analyse effectuée a démontré la rapidité de notre méthode et sa capacité à capturer les caractéristiques nécessaires à la localisation en temps réel. De plus, elle a surpassé les méthodes de localisation bien connues.

- **Contenu** Dans cet article, nous proposons un filtre à particules de regroupement modifié, une extension qui accélère le processus de calcul de la position en accélérant le processus d'appariement dans la phase de mise à jour des mesures, c'est-à-dire qu'au lieu d'apparier toutes les particules avec la carte des caractéristiques, nous n'utilisons que quelques particules prototypes qui peuvent être efficacement distinguées. Nous avons abordé le problème de sélection des particules avec deux approches : des méthodes basées sur le regroupement et des méthodes basées sur les points sigma, dont chacune est examinée en détail. Pour faciliter la compréhension du flux de travail de notre méthode, nous avons exposé une architecture composée de la cartographie et de la localisation. La phase de cartographie est chargée d'extraire les caractéristiques des mesures du Lidar (en particulier le regroupement par fuzzy c-means et le filtrage du bruit avec DBSCAN) et de créer une carte globale de

l'environnement. La phase de localisation est chargée de l'entrée de l'IMU et des résultats de la phase de cartographie pour déterminer la position corrigée.

- **Résultats** Le premier type d'expérience a été mené pour étudier les performances de nos deux modèles (basés sur les techniques de regroupement ou sur les points sigma) et pour tester dans quelle mesure chacun est capable de localiser précisément le véhicule sans gaspiller trop de temps et d'énergie. Pour ce faire, nous avons distingué un certain nombre de séquences courtes provenant des ensembles de données Kitti et Pandaset, contenant plusieurs changements environnementaux et différentes formes de trajectoires.

  Les résultats présentés dans le Tableau 4 et le Tableau 5 montrent la précision des deux modèles pour localiser le véhicule dans les séquences choisies ainsi que le temps consommé pour chaque modèle. Il a été démontré que nos modèles sont capables de déterminer précisément les positions en un court laps de temps de calcul ; **41 s** pour localiser le véhicule dans 544 images (105 s pour l'ensemble des séquences) avec une erreur de positionnement absolu moyenne de **0,24 m**, ce qui démontre l'efficacité de l'utilisation des techniques de regroupement pour réduire le temps de calcul des positions. En revanche, la localisation basée sur les points sigma a utilisé **5 s** pour localiser le véhicule dans 544 images (105 s pour l'ensemble des séquences) avec une erreur de positionnement absolu moyenne de **0,16 m**, ce qui est extrêmement rapide et précis, surtout pour les applications en temps réel. Grâce aux moyennes et aux covariances des particules après la mise à jour du mouvement, la méthode des points sigma parvient à choisir parfaitement les points "gagnants" qui contribuent efficacement au calcul des positions. Selon l'analyse effectuée ci-dessus, la localisation basée sur les points sigma surpasse la localisation basée sur le regroupement tant en termes de précision que de vitesse. Par conséquent, elle sera notre choix pour le reste de cette étude. Le Tableau 6 confirme notre affirmation ; une erreur moyenne de positionnement absolu de 0,18 m a été obtenue pour toutes les séquences dans l'ensemble de données de référence Pandaset [195] sur 400 images en 5 s d'exécution, ce qui démontre une fois de plus la robustesse et l'efficacité de notre méthode pour localiser le véhicule dans différents scénarios environnementaux. Nous avons testé manuellement la méthode de Scheafer sur la séquence 0001 (catégorie urbaine) de l'ensemble de données Kitti et obtenu une erreur de pose absolue de 0,06 m et une erreur de rotation absolue de 0,1 degré. L'exécution a duré 19 s, ce qui était plus long et énergivore que ce que nous avons obtenu dans le Tableau 7.

Une erreur de pose absolue de 0,054 m, une erreur de rotation de 0,01 degré en 1 s d'exécution. Nous pouvons également voir dans le Tableau 7 que nous avons des erreurs de rotation et de translation excellentes, dépassant ainsi toutes les méthodes concurrentes, ce qui est dû à la technique robuste qui distingue les points sigma des particules en fonction des moyennes et des covariances. Scheafer et al. enregistrent également une petite erreur de pose absolue par rapport à l'association de données suivie dans leur travail. L'extraction de caractéristiques polaires, comme dans le travail de Scheafer et al., contribue également de manière significative à obtenir ces résultats. Cependant, les poteaux, tels que les arbres, ou même l'utilisation de murs, n'existent pas dans tous les environnements, ce qui affecte particulièrement le processus de localisation. L'extraction de caractéristiques non sémantiques garantit l'existence d'un certain nombre de caractéristiques qui représenteront les données, assurant ainsi la continuité du flux de travail d'association de données. De plus, ce travail étendu surpasse celui de Charroud. A et al. [43] en termes de précision et de robustesse, où ils ont utilisé des modèles de mélange gaussien (GMM) pour trouver la carte globale, ce qui affecte le processus d'association de données. Même en termes de rapidité du processus, GMM avec un nombre fixe de clusters induit en erreur le processus d'appariement. En revanche, nous avons concaténé toutes les caractéristiques reçues de la carte locale pour garantir la fiabilité du processus d'appariement.

Nous avons testé notre méthode sur des séquences de durée moyenne et longue provenant de l'ensemble de données Kitti odometry (différentes des séquences courtes présentées dans le Tableau 5 et le Tableau 7). La méthode a été évaluée sur la séquence 00 ainsi que les séquences 05 à 10, totalisant 17554 images et 29 minutes d'exécution. Selon le Tableau 8, notre méthode a produit d'excellents résultats ; une erreur de positionnement absolu moyenne de 0,69 m sur toutes les séquences. Nous avons comparé notre méthode avec des méthodes d'enregistrement plus avancées, notamment LOAM_Velodyne [200], LeGO- LOAM [202], A- LOAM [203], SGLO [205], LO-Net [204], LO-Net-M [204] et la nôtre. La métrique utilisée par ces méthodes s'appelle t_{rel} ; la dérive de translation relative en moyenne sur des trajectoires de 100 à 800 m. Dans notre cas, cependant, nous avons pris l'erreur absolue moyenne sur l'ensemble des trajectoires. Étant donné que l'approche utilisée pour localiser le véhicule diffère entre nous et les méthodes d'enregistrement, l'analyse a été réalisée approximativement pour fournir une inspection générale des perfor-

mances des méthodes. Le Tableau 8 montre que notre méthode surpasse les autres méthodes dans le scénario de 4 séquences, y compris les trajectoires de longue durée. De plus, nous avons obtenu des résultats compétitifs dans d'autres séquences. Notre méthode enregistre une erreur de positionnement de 1,16 m dans la séquence 08, une longue trajectoire de 08 min : 38 s de conduite. Encore une fois, notre méthode prouve sa rapidité en effectuant la tâche de localisation en seulement 22s.

D'autre part, notre méthode a été capable de générer avec précision la carte globale de l'environnement pour ces trajectoires à long terme, ce qui a démontré les performances de notre flux de travail d'extraction de caractéristiques.

- **Conclusion**

  Dans cet article, nous avons proposé une méthode de localisation et de cartographie à sources multiples pour les véhicules autonomes basée sur l'extraction de caractéristiques non sémantiques. La méthode fuzzy c-means a été utilisée pour détecter les motifs dans les scans LiDAR. Les centres des clusters ont été filtrés par un algorithme DBSCAN pour éliminer le bruit et ont été utilisés pour créer les caractéristiques de la carte locale. De plus, la carte globale a été créée en fusionnant les caractéristiques de la carte locale et en les transformant en un système de coordonnées de référence, ce qui facilite l'association des données. Le processus de localisation a été effectué à l'aide d'un filtre particulaire avec des modifications, qui est une technique permettant de distinguer certaines caractéristiques pertinentes parmi les particules générées afin d'accélérer le processus de calcul et de suivre efficacement les positions de trajectoire. Nous avons proposé deux approches pour distinguer ces points ; des approches basées sur le regroupement et sur les points sigma. Les points sigma ont enregistré la meilleure précision en peu de temps de calcul, ce qui a été notre choix pour les expériences et les comparaisons suivantes.

  Notre approche proposée (points sigma) a obtenu des résultats compétitifs en termes de précision, mais en beaucoup moins de temps par rapport aux méthodes de pointe évaluées sur l'ensemble de données de référence Kitti et pandaset ; une erreur moyenne de positionnement absolu de 0,16 m pour l'ensemble des séquences à court terme de l'ensemble de données Kitti en 5 s d'exécution. De plus, nous avons obtenu une erreur moyenne de positionnement absolu de 0,18 m pour l'ensemble des séquences à court terme de l'ensemble de données Pandaset. Nous avons largement testé notre méthode sur des séquences à long terme et avons obtenu une erreur moyenne de positionnement absolu de 0,69 m pour l'ensemble des séquences, ce qui

démontre la robustesse, la fiabilité et l'efficacité de notre méthode. Par ailleurs, nous avons comparé notre méthode avec des travaux récents de pointe. En effet, notre méthode surpasse toutes les autres dans les séquences à court terme, en particulier la méthode de Schaefer et al. [39], où nous avons enregistré 0,077 m, alors que la méthode de Schaefer a enregistré 0,096 m. De plus, nous avons obtenu des résultats compétents dans les séquences à long terme.

Cependant, quelques déviations de précision se sont produites en raison d'une mauvaise initialisation du filtre particulaire ou d'un mauvais choix de cluster et de nombre de particules. Par conséquent, notre travail futur se concentrera principalement sur la recherche de la meilleure initialisation et l'étude des paramètres pour trouver la meilleure configuration basée sur un contrôle optimal ou en utilisant l'apprentissage automatique.

0.5 Méthode Rapide de Localisation et de Cartographie basée sur les Filtres à Particules Adaptatifs

- **Résumé** Avec le développement des véhicules autonomes, les technologies de localisation et de cartographie sont devenues cruciales pour doter le véhicule des connaissances appropriées à son fonctionnement. Dans cet article, nous étendons notre travail précédent en prétraitant une architecture de localisation et de cartographie pour les véhicules autonomes qui ne dépendent pas du GPS, en particulier dans des environnements tels que les tunnels, sous les ponts, les canyons urbains et les canopées d'arbres denses. L'approche proposée se compose de deux parties. Tout d'abord, un algorithme K-means est utilisé pour extraire des caractéristiques des scènes LiDAR afin de créer une carte locale pour chaque balayage. Ensuite, nous concaténons les cartes locales pour créer une carte globale de l'environnement et faciliter l'association des données entre les images. Ensuite, la tâche principale de localisation est effectuée par un filtre à particules adaptatif en quatre étapes : (a) la génération de particules autour d'un état initial (fourni par le GPS) ; (b) la mise à jour des positions des particules en fournissant le mouvement (translation et rotation) du véhicule à l'aide d'un dispositif de mesure inertielle ; (c) la sélection des meilleures particules candidates en observant à chaque instant la correspondance (également appelée poids des particules) de la carte locale (avec les distances en temps réel aux objets) et les distances des particules aux portions correspondantes

de la carte globale ; (d) la moyenne des particules sélectionnées pour obtenir la position estimée, et enfin, l'utilisation d'une méthode de rééchantillonnage sur les particules pour garantir la fiabilité de l'estimation de la position. La performance de la technique nouvellement proposée est étudiée sur différentes séquences des données brutes de Kitti et Pandaset avec différentes configurations environnementales, conditions météorologiques et changements saisonniers. Les résultats obtenus valident les performances de l'approche proposée en termes de vitesse et de représentativité de l'extraction des caractéristiques pour la localisation en temps réel par rapport à d'autres méthodes de pointe.

- **Contenu** Dans cette étude, nous proposons une technique de regroupement flou K-means pour représenter efficacement l'environnement, ce qui facilite le processus de recherche de parties similaires entre le balayage de la carte locale t et les parties correspondantes sur la carte globale. La carte globale contient des informations sur les objets dans la trajectoire. Après la génération de ces cartes (locale et globale), un filtre à particules adaptatif est utilisé pour localiser le véhicule le long de la trajectoire. Le filtre à particules adaptatif (APF) est une méthode de filtrage de particules soutenue par une technique de rééchantillonnage qui permet aux particules de suivre approximativement les positions réelles de la trajectoire et de maintenir la précision. L'APF utilise la carte locale et globale créée pour distinguer les meilleures particules candidates pour le calcul de la position. Dans cet article, nous étendons les travaux menés dans [43] sur la réduction de la carte des caractéristiques non sémantiques en utilisant K-means, la génération de la carte globale en utilisant GMM et la localisation en utilisant les filtres à particules en ajoutant une technique de rééchantillonnage au filtre à particules (c'est-à-dire en utilisant le filtre adaptatif) pour obtenir une solution de localisation plus précise. De plus, nous avons étudié plus en détail les différents paramètres du filtre à particules et l'effet de l'extension du rééchantillonnage sur les performances de notre technique de localisation et de cartographie proposée.

- **Résultats** Dans cette étude, nous proposons une technique de regroupement flou K-means pour représenter efficacement l'environnement, ce qui facilite le processus de recherche de parties similaires entre le balayage de la carte locale t et les parties correspondantes sur la carte globale. La carte globale contient des informations sur les objets dans la trajectoire. Après la génération de ces cartes (locale et globale), un filtre à particules adaptatif est utilisé pour localiser le véhicule le long de la

trajectoire. Le filtre à particules adaptatif (APF) est une méthode de filtrage de particules soutenue par une technique de rééchantillonnage qui permet aux particules de suivre approximativement les positions réelles de la trajectoire et de maintenir la précision. L'APF utilise la carte locale et globale créée pour distinguer les meilleures particules candidates pour le calcul de la position. Dans cet article, nous étendons les travaux menés dans [43] sur la réduction de la carte des caractéristiques non sémantiques en utilisant K-means, la génération de la carte globale en utilisant GMM et la localisation en utilisant les filtres à particules en ajoutant une technique de rééchantillonnage au filtre à particules (c'est-à-dire en utilisant le filtre adaptatif) pour obtenir une solution de localisation plus précise. De plus, nous avons étudié plus en détail les différents paramètres du filtre à particules et l'effet de l'extension du rééchantillonnage sur les performances de notre technique de localisation et de cartographie proposée. Notre processus d'extraction de caractéristiques, illustré à la Figure 5.3, réduit efficacement le temps et le coût de calcul, où chaque balayage de taille 1,8 Mo (en moyenne) a été traité en 0,4 s (en moyenne) et réduit en 790 Ko (en moyenne), ce qui démontre la légèreté de nos cartes globales et locales. De plus, le processus facilite la phase d'association de données pour être rapide, facilitant les enquêtes sur les problèmes de similarité.

Les résultats dans le Tableau 5.2 montrent que la méthode peut localiser de manière fiable le véhicule dans différentes séquences et catégories, telles que les changements météorologiques et saisonniers. De plus, la méthode prend 7 s pour localiser le véhicule dans 5 séquences et 544 images, ce qui est suffisamment rapide pour fonctionner dans un scénario en temps réel ; le capteur IMU prend 52 s pour enregistrer des informations à partir de 544 images, et notre méthode traite rapidement les données entrantes avec un temps d'exécution de 7 s. Les tests manuels de la méthode de Schaeffer et al. [31] sur la séquence 0001 (catégorie Ville) de l'ensemble de données Kitti ont montré une erreur de position absolue de 0,06 m et une erreur de rotation absolue de $(0, 1)^{\check{r}}$ en 19 s. Dans notre cas, nous avons obtenu 0,07 m pour l'erreur de position absolue et $(0, 013)^{\check{r}}$ pour l'erreur de rotation en un temps d'exécution de 6 s, comme présenté dans le Tableau 5.2, ce qui démontre la rapidité de notre méthode. Selon le Tableau 5.2, nous avons enregistré 0,25 m de l'erreur de position moyenne de toutes les séquences en seulement 7 s d'exécution, ce qui démontre la capacité de notre méthode à localiser le véhicule avec un minimum de 100 particules. Les résultats obtenus sont dus à la robuste technique de

rééchantillonnage qui a suivi avec précision la trajectoire. Dans une autre partie, nous avons testé la précision de notre méthode par rapport à quatre méthodes que nous avons trouvées pertinentes dans notre étude de l'état de l'art : Kümmerle et al. [27], Weng et al. [29], Sefati et al. [26], et Schaefer et al. [31]. Tous ont utilisé des caractéristiques sémantiques pour représenter les balayages LiDAR, telles que des poteaux, des murs, des arbres, etc. La comparaison des approches de Sefati et al. [26] et Schaefer et al. [31] avec notre méthode proposée était faisable, car ils ont utilisé les mêmes métriques d'erreur, et leur mise en œuvre était fournie dans leur article. Cependant, il était difficile de comparer notre méthode avec la méthode de Weng et al. [29] et Kümmerle et al. [27] sauf qualitativement, comme cela a été fourni par Schaefer et al. [31] dans l'article. Le Tableau 5.3 montre que notre méthode peut produire des résultats compétitifs. En fait, notre méthode a surpassé toutes les autres en termes d'erreur angulaire absolue moyenne et d'erreur angulaire quadratique moyenne. De plus, nous avons obtenu des résultats compétitifs pour l'erreur de positionnement absolu moyen et l'erreur de positionnement quadratique moyenne, qui ont surpassé toutes les méthodes comparées, sauf Schaefer et al. [31] (avec nos résultats 0,09 m et 0,11 m, respectivement) et Schaefer et al. [31] (0,11 m et 0,12 m, respectivement, pour notre méthode). Ces résultats sont justifiés par l'association de données robuste dans le travail de Schaefer et al. [31]. Cependant, travailler avec des caractéristiques sémantiques peut affecter le processus de localisation, en particulier lorsque ces caractéristiques ne sont pas présentes dans l'environnement. Nous avons largement testé notre méthode sur un deuxième ensemble de données, Pandaset, sur 5 séquences contenant chacune 80 images. Notre méthode a enregistré une erreur de positionnement absolue moyenne de 0,16 m en 5 s d'exécution, démontrant la performance de notre méthode lorsqu'elle traite différents scénarios environnementaux (voir le Tableau 5.4).

Nous avons étudié en profondeur les résultats de la séquence 0009. Dans les figures 5.8a, 5.8c, nous avons cartographié les variations de la position et de l'erreur angulaire dans chaque balayage, ce qui nous a permis de faire des observations pertinentes. Nous avons constaté que l'accélération affecte principalement la précision du positionnement. Du balayage 0 au 20, période pendant laquelle le véhicule décolle, et du balayage 110 au 170, période pendant laquelle le véhicule ralentit en prenant un virage sur la route (figure 5.8c). Dans les deux périodes, l'erreur varie de 0 m à 0,25 m et revient à 0 m, ce qui signifie que les particules ont besoin de temps pour

s'adapter à la nouvelle accélération du véhicule. Il en va de même en cas de dérive ou de freinage brusque, comme dans les figures 5.8c, 5.8d et les figures 5.9a, 5.9b séquences 0035 et 0027. Par conséquent, un bon modèle de positionnement doit prendre en compte la vitesse pour anticiper le mouvement du véhicule. De plus, nous observons que l'erreur de positionnement est systématiquement de 0,1 m du balayage 170 au 250 lorsque le véhicule tourne. Cependant, l'erreur angulaire enregistre des valeurs élevées de 0 à 0,4, indiquant la nécessité de déterminer l'intensité de la rotation. De plus, le même problème a été enregistré dans la séquence 0034 dans la figure 5.8 et la séquence 0053 dans la figure 5.9. À mesure que la recherche dans le domaine de la localisation et de la cartographie passe des études observationnelles aux applications directes dans des scénarios du monde réel, la nécessité de méthodes qui accélèrent le processus de localisation, comme la méthode que nous proposons dans cet article, augmente. Cependant, il y a quelques inconvénients à l'utilisation d'algorithmes de filtrage de particules, en particulier pour fixer le nombre de particules et l'identification du seuil.

- **Conclusion** Cet article présente une méthode de positionnement et de cartographie des véhicules autonomes basée sur l'extraction de caractéristiques non sémantiques. Un regroupement flou K-means a été utilisé pour extraire des caractéristiques des balayages LiDAR. Les caractéristiques des centroïdes de cluster ont été utilisées pour créer des caractéristiques de carte locale. De plus, un filtre à particules adaptatif a été utilisé dans le processus de localisation, qui comprenait une étape de rééchantillonnage après la fin des mises à jour de mesure afin d'augmenter la fiabilité de l'estimation de la position et de réduire le temps et les coûts énergétiques. La méthode de rééchantillonnage sélectionne les 10 particules les plus proches de la position réelle en vérifiant leurs poids (pourcentage) et régénère des particules autour d'elles en utilisant une distribution multinormale.

  La méthode proposée fournit des résultats de précision compétitifs en beaucoup moins de temps par rapport aux méthodes de pointe évaluées sur la base de données Kitti. Nous avons obtenu une erreur de 0,25 m dans l'erreur moyenne de position de toutes les séquences de l'ensemble de données Kitti en un temps d'exécution de 7 s. De plus, nous avons obtenu 0,15 m dans l'erreur moyenne de position de toutes les séquences de l'ensemble de données Pandaset en seulement 5 s d'exécution, ce qui démontre le potentiel de notre contribution au rééchantillonnage pour accélérer le processus de localisation et obtenir des résultats de pointe. La méthode pro-

posée fournit des résultats de précision compétitifs en beaucoup moins de temps par rapport aux méthodes de pointe qui ont été évaluées sur l'ensemble de données Kitti.

Cependant, la précision de la méthode proposée déviait parfois en raison d'une mauvaise initialisation des filtres à particules ou d'une mauvaise sélection de clusters ou du nombre de particules. Par conséquent, les futures recherches impliqueront de trouver l'initialisation optimale du filtre à particules et de déterminer la meilleure configuration des paramètres. Le filtre à particules souffre de l'initialisation aléatoire des particules, qui pourrait être modélisée par une équation différentielle stochastique pour contrôler la génération des particules. D'autre part, l'utilisation d'autres méthodes de regroupement ouvre une nouvelle approche pour la représentation des mesures LiDAR, ce qui peut fournir des perspectives sur l'utilisation d'autres techniques d'apprentissage automatique, telles que le suivi d'objets avec l'apprentissage en profondeur (ou l'apprentissage par transfert), pour assurer une meilleure correspondance de l'association des données. Ainsi, la position du véhicule pourrait être estimée de manière plus précise.

0.6 XDLL : Méthode de Localisation et de Cartographie pour les Véhicules Autonomes Basée sur l'Apprentissage en Profondeur Expliqué à partir de Données LiDAR

- **Résumé** Les véhicules autonomes nécessitent un système de positionnement robuste pour poursuivre la révolution dans le domaine des transports intelligents. Les systèmes de navigation par satellite à l'échelle mondiale (GNSS) sont les plus couramment utilisés pour accomplir cette tâche en raison de leur capacité à localiser avec précision le véhicule dans l'environnement. Cependant, des publications récentes ont révélé des cas graves où le GNSS échoue misérablement à déterminer la position du véhicule, par exemple sous un pont, dans un tunnel ou dans des forêts denses. Dans ce travail, nous proposons une architecture de cadre pour expliquer les modèles d'apprentissage en profondeur basés sur LiDAR (XDLL) qui prédisent la position des véhicules en utilisant seulement quelques points LiDAR dans l'environnement, ce qui garantit la rapidité et la précision nécessaires des

interactions entre les composants du véhicule. Le cadre proposé extrait des caractéristiques non sémantiques des balayages LiDAR à l'aide d'un algorithme de regroupement. Les clusters identifiés servent d'entrée à notre modèle d'apprentissage en profondeur, qui repose sur des couches LSTM et GRU pour stocker les points de trajectoire et des couches convolutionnelles pour lisser les données. Le modèle a été largement testé avec des trajectoires de courte et de longue durée à partir de deux ensembles de données de référence, Kitti et NCLT, contenant différents scénarios environnementaux. De plus, nous avons examiné les résultats obtenus en expliquant la contribution de chaque caractéristique de cluster à l'aide de plusieurs méthodes explicables, dont Saliency, SmoothGrad et VarGrad. L'analyse a montré que prendre la moyenne de tous les clusters comme entrée pour le modèle est suffisant pour obtenir une meilleure précision par rapport au premier modèle, et cela réduit également la consommation de temps. Le modèle amélioré est capable d'obtenir une erreur de positionnement absolue moyenne inférieure à un mètre pour toutes les séquences dans les trajectoires de courte et de longue durée.

- **Contenu** Notre méthode novatrice utilise des caractéristiques non sémantiques comme étape de prétraitement pour enseigner à un modèle d'apprentissage automatique les positions réelles des véhicules. Les entrées sont les caractéristiques extraites, et les sorties sont les positions. Le modèle a été largement testé sur des trajectoires de courte et de longue durée à l'aide de deux ensembles de données de référence : Kitti [187] et NCLT [190]. De plus, le modèle est expliqué en étudiant les caractéristiques qui contribuent positivement ou négativement au modèle de sortie. Sur la base des résultats de l'explication du premier modèle, un autre modèle a été construit et comparé pour évaluer l'influence des modifications apportées. L'article [215] aborde le concept d'intelligence artificielle explicable (IAE), qui fait référence au développement de systèmes d'IA capables de fournir des explications claires et interprétables pour leurs décisions et actions. Les auteurs fournissent un aperçu des différentes approches d'IAE qui ont été proposées, notamment les méthodes d'explication post hoc, les méthodes d'explication intégrées et les méthodes d'explication interactives. Ils explorent également les avantages et les défis potentiels de l'IAE, et discutent de l'importance du développement responsable de l'IA pour garantir la transparence, la responsabilité et la fiabilité des systèmes d'IA.

  Cet article présente les contributions suivantes :

- Dans la mesure de notre connaissance, c'est la première méthode qui utilise seulement quelques points de cluster LiDAR pour alimenter un modèle d'apprentissage en profondeur à des fins de localisation.
- L'article présente, dans la Section 6.2, une formulation mathématique robuste du problème de localisation, ce qui ouvre des opportunités pour développer davantage de solutions basées sur l'optimisation et les méthodes basées sur les équations différentielles stochastiques.
- Des méthodes d'explication de l'apprentissage en profondeur ont été utilisées pour trouver les caractéristiques de cluster les plus contributives afin d'optimiser le modèle proposé.

- **Résultats**

  Comment le Modèle Performe : Selon la Figure 6.6, la perte d'entraînement a diminué de 0,95m à 0,75m, ce qui montre l'importance de l'ajustement du modèle. Pendant ce temps, la perte de validation a diminué de 0,915 à 0,75 m, ce qui indique que le modèle n'a pas surajusté.

  La Table 6.5 examine les performances du modèle lors d'une série de tests sur l'ensemble de données Kitti. Nous avons choisi quelques petites séquences pour voir comment le modèle se comporte dans une telle situation. Nous avons déduit que notre modèle fonctionne bien pour prédire les positions sur ces petites séquences. L'erreur absolue moyenne de toutes les séquences n'a pas dépassé 1m, ce qui signifie que la majorité des valeurs d'erreur sont inférieures à 1m ; nous avons enregistré une valeur maximale allant jusqu'à 2 m. Ces séquences ont été choisies car elles contiennent les scénarios les plus difficiles, tels que les virages, les rond-points, l'accélération et les changements de conditions météorologiques.

  La Table 6.6 décrit une autre situation où nous avons testé approximativement notre modèle sur des trajectoires de long terme (plus de 3km) et dans différents scénarios et défis environnementaux. Le modèle a de nouveau enregistré une erreur absolue moyenne inférieure à 1m entre les positions prédites et les positions réelles. Cependant, existe-t-il une chance d'améliorer la précision ? La réponse est oui, si nous expliquons les résultats, par exemple, en découvrant quelles caractéristiques contribuent positivement ou négativement à la sortie du modèle.

  Pourquoi Cette Sortie ? Comme mentionné dans la Section 6.3.4, quatre méthodes explicatives ont été choisies pour analyser la contribution des caractéristiques (y

compris les axes x et y) pour fournir la sortie résultante : la saillance, les gradients intégrés, SmoothGrad et VarGrad. Nous avons également utilisé le noyau Shap et l'expliquant LIME, qui sont des méthodes très populaires pour les problèmes de régression. Cependant, nous avons obtenu des valeurs similaires pour chaque caractéristique, ce qui ne fournit pas beaucoup d'informations sur les contributions des caractéristiques. La Figure 6.7 montre l'impact moyen de chaque caractéristique (des axes x et y) selon chaque expliquant (colonne de gauche) et l'impact ponctuel de chaque caractéristique (des axes x et y) selon chaque expliquant (colonne de droite). Notre critère de sélection des caractéristiques était la valeur d'impact moyenne des coordonnées des caractéristiques ; sur l'axe x et y. Sur cette base, la saillance et les gradients intégrés (IG) ont nommé le cluster 2 comme la caractéristique la plus intéressante. Cependant, SmoothGrad a sélectionné la moyenne des clusters comme la caractéristique la moins importante, indiquant qu'elle contribue négativement au modèle. De plus, VarGrad a confirmé que la moyenne des clusters a une contribution positive majeure au modèle. Notez que chaque expliquant a sa propre philosophie pour calculer la valeur d'impact. Ainsi, comment pouvons-nous décider quelle caractéristique a la meilleure contribution ? Nous avons utilisé deux mesures de fidélité : la suppression et l'insertion, suivant un autre article [221]. Selon les Tables 6.7 et 6.8, les résultats de SmoothGrad et de VarGrad sont plus fiables selon les deux critères. SmoothGrad et VarGrad ont enregistré respectivement 89,61 et 105,72 dans la mesure de suppression (plus bas est meilleur), et 144,32 et 131,36 respectivement dans la mesure d'insertion (plus élevé est meilleur).

De la Figure 6.7, nous pouvons extraire plus d'informations des graphiques ponctuels (colonne de droite), en particulier de la Figure 6.7f,h. Nous remarquons que des valeurs plus élevées pour la moyenne de toutes les données de caractéristiques contribuent positivement, contrairement aux valeurs les plus basses, qui contribuent négativement.

Selon l'analyse ci-dessus, la moyenne de tous les clusters est la caractéristique la plus importante qui contribue positivement à la réduction de l'erreur de perte.

Du côté droit de chaque image, il y a une barre de couleur qui montre les contributions des clusters : le violet signifie des contributions faibles et le jaune signifie des contributions élevées. Sur l'axe x, nous avons l'impact calculé selon chaque méthode.

Que se Passera-t-il Si Nous Changeons Cela ? Nous avons ensuite exploré le fait que

la moyenne de tous les clusters est la caractéristique la plus importante pour créer un autre modèle avec la même architecture que le premier et comparé les résultats de prédiction des deux modèles pour étudier l'amélioration. La Figure 6.8 présente la courbe des valeurs de perte (entraînement et validation). La perte d'entraînement est passée de 13,49 à 0,68, et la perte de validation est passée de 4,41 à 0,68, ce qui est beaucoup plus bas que ce que nous avons obtenu dans le premier modèle.

La Table 6.9 montre les résultats de prédiction du modèle nouvellement développé. Nous avons constaté que l'EMA de presque toutes les séquences a été réduite par rapport à celle du premier modèle (Tableau 6.5). L'EMA moyenne de toutes les séquences était de 0,51m, alors qu'elle était de 0,58m pour le modèle précédent. Le nouveau modèle a amélioré la précision de prédiction des positions par rapport au premier modèle. Non seulement cela, mais nous avons supprimé les caractéristiques des clusters qui trompent le modèle et qui sont trop chronophages dans la phase d'extraction et de formation. Nous avons étayé ces conclusions en testant également le modèle sur des séquences à long terme. La Table 6.10 présente les résultats de prédiction pour les trajectoires à long terme, et notre modèle a obtenu des valeurs d'EMA inférieures par rapport au premier modèle.

- **Conclusion** La localisation et la cartographie dans les véhicules autonomes ont été largement abordées avec différentes approches, notamment des méthodes probabilistes, d'optimisation et d'autres méthodes d'apprentissage automatique. Dans cet article, nous avons présenté un nouveau flux de travail basé sur l'apprentissage en profondeur pour apprendre les positions des véhicules. Les caractéristiques d'entrée du modèle ont été extraites des balayages LiDAR à chaque instant. Le processus d'extraction était basé sur l'application d'un algorithme flou de k-moyennes qui extrait des caractéristiques à partir de clusters. L'architecture du modèle était basée sur une combinaison de couches de modèles LSTM et GRU et un lissage avec une convolution 1D. Nous avons réglé le modèle pour obtenir les meilleurs hyperparamètres, et nous l'avons entraîné avec différentes paramétrisations, telles que l'arrêt précoce, la réduction du taux d'apprentissage en cas de métriques constantes, etc. Le modèle a obtenu de très bons résultats à court et à long terme dans différents scénarios environnementaux exigeants, tels que des changements climatiques et divers scénarios de trajectoires. Le modèle est capable de maintenir l'erreur de positionnement absolue moyenne en dessous de 1 m pour toutes les séquences dans les trajectoires à court et à long terme.

Nous avons également fourni des explications possibles pour les résultats du modèle et examiné la contribution des clusters (caractéristiques). Nous avons constaté que la moyenne de tous les clusters extraits est la caractéristique la plus importante qui contribue positivement au résultat de prédiction. Nous avons créé un nouveau modèle expliqué qui prend uniquement la moyenne de tous les clusters en entrée et avons exécuté à nouveau la prédiction. Selon les résultats de comparaison des deux modèles, le nouveau modèle plus explicatif améliore la précision du positionnement des véhicules et réduit le temps et les ressources informatiques nécessaires pour former et utiliser ces modèles.

0.7 Amélioration de la Localisation LiDAR basée sur l'Autoencodeur dans les Véhicules Autonomes

- **Résumé** La capacité des véhicules autonomes à effectuer avec succès des tâches de navigation repose fortement sur la mise en œuvre d'un système de localisation solide. Les systèmes de navigation par satellite globaux ont été utilisés pour fournir ces informations au véhicule. Cependant, ils sont confrontés à des problèmes sérieux qui interfèrent avec le calcul de la position, comme dans les tunnels, sous les ponts, dans les canyons urbains. Cet article propose une méthode basée sur un autoencodeur pour prédire la position d'un véhicule basée sur un nuage de points LiDAR (converti en un système de référence) sans utiliser de capteurs supplémentaires. Le modèle proposé modifie la fonction objectif de l'autoencodeur pour inclure un objectif supplémentaire responsable de l'adaptation de la sortie de la couche d'encodeur (l'espace latent) pour prédire la position. De plus, un paramètre de pénalité a été utilisé dans la fonction objectif pour mettre l'accent sur l'apprentissage dans l'espace latent. La convergence de la fonction objectif proposée a été prouvée en montrant qu'il s'agit d'une fonction stochastique locale quasi-convexe, ce qui garantit l'efficacité de l'algorithme de descente de gradient normalisé stochastique pour atteindre la convergence. L'ensemble de données NCLT a été utilisé avec notre modèle dans PyTorch, et les performances du modèle sont évaluées pour différentes valeurs du paramètre de pénalité α. Le processus de recherche de paramètres repose sur l'algorithme d'optimisation bayésienne. Il est montré que $\alpha = 0.01$ est le meilleur choix. En comparaison avec diverses techniques d'enregistrement avancées, notre méthode obtient d'excellents résultats pour la majorité des séquences de l'ensemble

de données Kitti également. L'avantage de la méthode réside dans sa capacité à effectuer des prédictions directes sans optimisation itérative coûteuse, une étape courante dans les techniques d'enregistrement. De plus, la méthode a obtenu une erreur absolue moyenne d'environ 0.5, ce qui est souhaitable pour les véhicules autonomes. La méthode proposée a le potentiel de permettre une navigation plus précise et plus efficace des véhicules autonomes.

- **Contenu** Notre architecture AutoLocal repose sur un réseau neuronal autoencodeur avec une extension spéciale qui donne du sens aux résultats de l'espace latent. L'objectif principal de cette architecture est d'apprendre au modèle les positions des véhicules et d'utiliser cette connaissance pour faire des prédictions précises pour des données non vues auparavant. Pour atteindre cet objectif, nous avons modifié la fonction objectif de l'autoencodeur pour couvrir l'apprentissage de l'espace latent et montrer que l'algorithme converge vers le minimum (comme indiqué dans la dernière section). Dans ce travail, nous utilisons la mesure LiDAR en tant qu'entrée de l'autoencodeur, car elle fournit une représentation précise de l'environnement. Cependant, afin de ne pas surcharger le modèle de données, nous utilisons un algorithme de regroupement flou K-means pour regrouper tous les points, réduisant ainsi l'information à quelques clusters centraux. Nous avons choisi d'extraire cinq caractéristiques de cluster de chaque scène LiDAR car cela a été utilisé dans [43], [45], [44], [222]. De plus, nous avons également incorporé une sixième caractéristique, qui est la moyenne du cluster. Cette décision a été prise en se basant sur les conclusions de l'article [222], qui a démontré que la moyenne du cluster est une caractéristique cruciale qui contribue à l'estimation précise de la position du véhicule. Pour optimiser davantage notre architecture AutoLocal, nous avons limité la plage des données de mesure LiDAR prises en compte pour chaque scène. Plus précisément, nous ne tenons compte que des points dans la plage de -10 à 10 pour les coordonnées x et y. Ce faisant, nous sommes en mesure de réduire la charge computationnelle sur le modèle, tout en conservant suffisamment de données pour faire des prédictions précises sur la position du véhicule. Cette approche aide également à filtrer le bruit ou les points de données non pertinents qui pourraient être présents dans l'ensemble complet de données de mesure LiDAR, permettant au modèle de se concentrer uniquement sur les données les plus pertinentes pour la tâche.

  Le processus d'apprentissage expliqué ci-dessus est illustré dans la Figure 7.3 ; le

modèle apprend les positions du véhicule (représentées en rouge) en triangulant les caractéristiques bleues. La fiabilité des mesures LiDAR permet aux caractéristiques bleues de représenter avec précision divers objets dans l'environnement, y compris les véhicules en mouvement et les piétons, en faisant une méthode robuste d'extraction de caractéristiques. De plus, la caractéristique verte, qui représente la moyenne de toutes les caractéristiques bleues, contribue de manière significative au processus d'apprentissage. Cela est dû à sa proximité avec la position réelle, par rapport aux caractéristiques bleues individuelles, fournissant ainsi une justification supplémentaire aux résultats présentés dans [222].

Pratiquement, comme illustré dans la Figure 7.2, nous avons 6 neurones en entrée et en sortie du modèle, où chaque neurone est un vecteur de coordonnées tridimensionnelles (cluster). Le résultat de l'espace latent est un vecteur de coordonnées bidimensionnelles représentant la position du véhicule dans le système de coordonnées du monde réel. Nous extrayons les positions réelles des véhicules à partir de jeux de données ouverts tels que Kitti [187] et NCLT [190]. Nous utilisons les données NCLT pour l'entraînement et la validation car elles contiennent des trajectoires longues, et l'ensemble de données Kitti pour tester la performance du modèle sur des données non vues auparavant. Avec cette approche, nous nous attendons à ce que l'architecture AutoLocal fournisse des prédictions fiables, permettant aux véhicules autonomes de naviguer en toute sécurité et avec précision.

- **Résultats** l'objectif est de minimiser cette fonction objectif :

$$\min_{W}(\alpha \left\| g\left\langle W_2, f\left\langle W_1, x\right\rangle\right\rangle - x\right\|_2^2 + (1-\alpha)\left\| f\left\langle W_1, x\right\rangle - p\right\|_2^2)$$

Où un hyperparamètre supplémentaire est introduit, noté α, qui contrôle l'apprentissage de l'espace latent en choisissant différentes valeurs de α afin de donner la priorité à l'apprentissage de l'espace latent plutôt qu'à l'apprentissage de la reconstruction des centres de clusters, ce qui peut avoir un impact sur les performances globales du modèle. Cet article de recherche étudie l'impact de différentes valeurs de α sur les performances d'un autoencodeur. Plusieurs expériences ont été menées pour étudier cette relation, en utilisant l'algorithme d'optimisation bayésienne pour estimer les paramètres et déterminer la valeur optimale de α. Pour estimer précisément α, une approche stratégique a été adoptée en raison de la nature de sélection aléatoire de l'algorithme. Initialement, l'algorithme a été exécuté sur l'intervalle complet [0,1] pour identifier le meilleur sous-intervalle, tel que [0,0.5] ou [0.5,1], en utilisant

les données d'entraînement. Les paramètres de l'algorithme ont été maintenus par défaut.

Comme le montre la Figure 7.4 (image du bas), la perte globale diminue pour les valeurs plus petites de α (< 0.5). La meilleure valeur estimée de α a été trouvée être 0.02, résultant en une perte globale de 0.93. Cependant, en raison de la sélection aléatoire de l'algorithme, l'estimation peut ne pas être précise. Par conséquent, l'algorithme a été réexécuté sur l'intervalle [0,0.5], donnant une perte globale de 0.92 avec $\alpha = 0.0103$. Voir la Figure 7.4 (en haut à gauche). Pour assurer l'exploration du meilleur α dans l'autre intervalle, l'algorithme a été à nouveau exécuté sur l'intervalle [0.5,1], conduisant à une perte globale de 2.4 avec $\alpha = 0.5$. Voir la Figure 7.4 (en haut à droite). En conclusion, la valeur optimale de α représentant la perte globale a été déterminée à $\alpha = 0.01$.

En analysant le Tableau 7.2, une observation importante peut être faite : indépendamment des valeurs variables de α, la perte de reconstruction reste stable. Cela pourrait conduire à penser qu'elle peut être négligée et que seule la perte de position doit être considérée. Cependant, une telle notion est incorrecte car la phase de reconstruction joue un rôle crucial pour assurer une projection résiliente des clusters de coordonnées 3D vers les positions de coordonnées 2D. Il convient de noter que les meilleures pertes pour le modèle ont été obtenues lorsque α a été fixé à 0.01. L'évolution des trois pertes à travers les époques pour $\alpha = 0.01$ est représentée dans la Figure 7.5.

Ces dernières années, de nombreuses méthodes de pointe ont été proposées pour estimer les positions des véhicules en mouvement à l'aide de capteurs LiDAR. Ces méthodes reposent principalement sur des techniques d'enregistrement pour aligner les nuages de points LiDAR et estimer les positions. Dans cette étude, nous comparons notre méthode proposée à plusieurs de ces méthodes de pointe, dont LOAM [200], LOAM Velodyne [201], LeGO-LOAM [202], A-LOAM [203], SGLO [205], LO-Net et LO-Net-M [204]. Ces méthodes ont montré des résultats prometteurs dans divers environnements et scénarios. Par conséquent, nous évaluons notre méthode par rapport à celles-ci pour démontrer l'efficacité et la robustesse de chaque approche.

Le Tableau 7.3 présente l'évaluation de précision de notre méthode par rapport aux méthodes d'enregistrement de pointe sur le jeu de données Kitti. L'évaluation est effectuée en fonction de l'erreur absolue moyenne en mètres.

En termes de résultats d'évaluation, notre méthode surpasse la plupart des méthodes

comparées dans presque toutes les séquences testées, obtenant la plus faible erreur absolue moyenne. Plus précisément, dans les séquences 00, 06, 08, 09 et 10, notre méthode a obtenu la plus faible erreur absolue moyenne parmi toutes les méthodes comparées, démontrant l'efficacité de notre méthode proposée. De plus, dans les séquences 05 et 07, notre méthode a obtenu des performances compétitives avec la méthode la mieux performante. Ces résultats suggèrent que notre méthode proposée peut estimer avec précision la position et surpasser les méthodes d'enregistrement de pointe dans la plupart des scénarios testés.

Les méthodes d'enregistrement mentionnées dans le tableau reposent sur des informations proches des positions réelles, telles que les mesures IMU et d'autres capteurs. En revanche, notre méthode n'utilise que les points LiDAR pour prédire les positions, ce qui en fait une solution plus légère. De plus, les techniques d'enregistrement sont souvent connues pour leur coût computationnel élevé, tandis que notre méthode peut faire des prédictions directes sans nécessiter de calcul supplémentaire. La Figure 7.6 présente les résultats graphiques de la prédiction des trajectoires des véhicules (couleur orange) par rapport aux positions réelles (couleur bleue).

- **Conclusion** Dans ce travail, nous avons proposé une extension du modèle autoencodeur qui permet de prédire la position du véhicule à partir du nuage de points LiDAR. Nous parvenons à cela en modifiant la fonction objective principale de l'autoencodeur et en ajoutant une autre fonction objective qui est responsable de permettre à la couche d'encodage de mieux prédire les positions. De plus, nous introduisons un paramètre de pénalité qui est multiplié par les fonctions objectives afin d'accentuer l'apprentissage de l'espace latent. Cela nous permet de former notre modèle à apprendre les relations entre les données d'entrée et la position du véhicule de manière plus efficace.

  Pour démontrer la convergence de la nouvelle fonction objective, nous avons utilisé l'algorithme du Stochastic Natural Gradient Descent (SNGD) et exploré sa convergence vers le minimum pour les fonctions stochastiques localement quasi-convexes (SQLC). Nous montrons que la nouvelle fonction objective est une fonction (SQLC), ce qui facilite l'analyse de la convergence.

  Nous avons formé notre modèle à l'aide de l'ensemble de données NCLT, qui contient un grand nombre de trajectoires séquentielles et convient bien à cette tâche. Notre

modèle a été implémenté à l'aide de PyTorch, et nous avons utilisé l'optimiseur Adam pour mettre à jour les poids du réseau pendant l'entraînement.

Nous avons étudié les performances du modèle pour différentes valeurs du paramètre de pénalité α. Nous avons constaté que $\alpha = 0.01$ est le meilleur choix pour former notre modèle avec une fonction de perte minimale.

Nous avons évalué notre méthode sur l'ensemble de données KITTI et l'avons comparée à plusieurs techniques d'enregistrement de pointe, dont LOAM, LOAM Velodyne, LeGO-LOAM, A-LOAM, SGLO, LO-Net et LO-Net-M. Notre méthode a obtenu les meilleures performances en termes d'erreur absolue moyenne pour la plupart des séquences KITTI, surpassant toutes les autres méthodes.

Un avantage important de notre méthode est qu'elle ne nécessite pas de capteurs supplémentaires, tels que les IMU ou le GPS, pour prédire les positions du véhicule. Cela contraste avec les techniques d'enregistrement qui dépendent de ces capteurs pour fournir des informations supplémentaires sur le mouvement du véhicule. De plus, notre méthode est efficace du point de vue computationnel, car elle peut effectuer des prédictions directes sans nécessiter d'optimisation itérative, étape courante dans les techniques d'enregistrement.

Dans l'ensemble, notre méthode présente une solution prometteuse pour la conduite autonome, car elle offre une prédiction précise et efficace en termes de calcul de la position en utilisant uniquement des données LiDAR brutes. Cependant, notre méthode enregistre certaines perturbations pour prédire certaines parties de la trajectoire, en particulier dans un rond-point ou lorsque le véhicule tourne brusquement, ce qui nécessite des recherches plus approfondies dans nos travaux futurs.

0.8 Conclusion

Cette thèse explore le problème de la localisation et de la cartographie dans les véhicules autonomes, en particulier les défauts des systèmes de navigation par satellite (GNSS) pour accomplir cette tâche dans certains environnements particuliers qui coupent le signal entre les satellites et les véhicules.

En menant une revue de littérature exhaustive, nous avons exploré différentes méthodes et aspects dans ce domaine. En adoptant une architecture comprenant l'extraction de caractéristiques, la cartographie et la localisation, nous avons analysé ces méthodes, identifiant leurs points forts et leurs faiblesses (**contribution 1 (Revues de la littérature)**). Sur cette base, notre approche proposée basée sur le Lidar pour l'extraction de caractéristiques a montré des résultats prometteurs pour améliorer la vitesse et la précision du processus de localisation et de cartographie (**contribution 2**).

Nous avons présenté deux approches distinctes pour la localisation des véhicules : la stochasticité et l'optimisation. Dans l'approche stochastique, notre algorithme de filtre à particules modifié et la sélection de points spécifiques pour la mise à jour des poids ont démontré une localisation rapide et précise des véhicules. De plus, en ne sélectionnant que les particules les plus proches des positions réelles et en régénérant des particules autour d'elles, nous avons obtenu un meilleur contrôle et une meilleure durabilité (**contribution 3, 4**). Dans l'approche d'optimisation, nous avons introduit un modèle d'apprentissage en profondeur qui prédit les positions des véhicules en fonction des mesures Lidar. En utilisant des techniques d'intelligence explicative, nous avons obtenu des aperçus des prédictions du modèle et fourni une analyse détaillée. De plus, notre utilisation d'un modèle autoencodeur modifié, guidé par une fonction de perte spécifique, a permis une prédiction efficace de la position en exploitant les pertes de projection et de position des points Lidar (**contribution 5, 6**).

0.8.1 Défis et travaux futurs

Dans cette section, nous présentons les défis et les orientations futures de l'extraction de caractéristiques, de la cartographie et de la localisation, car nous sommes convaincus que toute méthode devrait utiliser cette architecture.

De plus, les résultats de notre enquête montrent l'efficacité de nos méthodes. Nous sommes heureux de noter que notre approche a surpassé l'état de l'art existant sur une large gamme de séquences de données, notamment Kitti et NCLT, soulignant ainsi la robustesse de nos techniques proposées dans des scénarios réels.

En regardant vers l'avenir, l'intégration de l'extraction de caractéristiques, de la cartographie et de la localisation est essentielle pour toute méthode opérant au sein de cette architecture. Alors que nous progressons, nous reconnaissons que relever les défis dans

chacun de ces aspects et tracer la voie de la recherche future contribuera collectivement à l'évolution des technologies de véhicules autonomes efficaces.

0.8.1.1 Extraction de caractéristiques

Afin de se localiser dans l'environnement, le véhicule doit explorer les informations reçues des capteurs. Cependant, la grande quantité de données reçues rend impossible leur utilisation en temps réel lors de l'exécution du véhicule, car celui-ci a besoin d'une interaction instantanée avec l'environnement, par exemple pour accélérer, freiner, diriger le volant. C'est pourquoi les systèmes embarqués ont besoin de méthodes d'extraction de caractéristiques efficaces qui distingueront les caractéristiques pertinentes pour une meilleure exécution du processus de localisation.

Après avoir passé en revue et analysé des articles pertinents, les défis suivants ont été identifiés :

- Les caractéristiques doivent être robustes face à tout effet externe, comme les changements climatiques, d'autres objets en mouvement, etc., par exemple les arbres qui bougent en raison du vent.
- Fournir la possibilité de réutilisation dans d'autres tâches.
- Le système de détection doit être capable d'extraire des caractéristiques même avec quelques objets dans l'environnement.
- Les algorithmes proposés pour l'extraction de caractéristiques ne doivent pas nuire au système en exigeant du temps de calcul et d'exécution.
- Une question qui doit être prise en compte concerne les caractéristiques sûres et dangereuses. Chaque caractéristique doit fournir un degré de sécurité (exprimé en pourcentage), ce qui aide à déterminer la nature de la caractéristique et où elle appartient, par exemple appartenant à la route est plus sûr que d'être sur les murs.

0.8.1.2 Cartographie

- Le stockage des données est un grand défi pour les VA. Nous devons trouver les informations minimales nécessaires pour exécuter l'algorithme de localisation.

Table 1: Summary of existing approaches to solve the localization problem.

Approach	Sensors	Score	Main Challenges
Dead Reckoning	GPS INS Wheel odometry	Low	Prone to error in case drifting
Triangulation	GNSS	High	Limited in some environment senarios
Motion tracking	LiDAR Cameras	Passable	High cost of time and energy A few objects in the environment mislead the matching process.
Optimization	LiDAR Cameras	High	High computational time to solve the matrix calculation
Probability	LiDAR Cameras	High	Problem of initialization
Cooperative localization	Vehicle connectivity	High	Limitation if there is a long distance between vehicles

- Les cartes sont sujettes à des changements fréquents, donc un système de mise à jour devrait être mis en place pour mettre à jour et maintenir les changements. De plus, le système de connectivité du véhicule devrait soutenir cette tâche et fournir les changements aux autres véhicules.
- Préserver les informations sur la localisation du véhicule et garantir la confidentialité.

0.8.1.3 Localisation

Nous avons passé en revue dans cette section les approches utilisées pour résoudre le problème de la localisation dans les véhicules autonomes (même pour tous les systèmes autonomes dans certains cas). Nous avons résumé dans la Tableau 4 ces approches, les capteurs applicables, leur précision dans les scénarios d'environnement réel et les principaux défis à prendre en compte dans la recherche à venir. Utiliser l'une de ces approches doit assurer une interaction rapide avec les autres tâches du véhicule, réduisant ainsi le temps et la consommation d'énergie pour les véhicules et préservant la durée de vie de la batterie à long terme. De plus, nous devons garantir la fiabilité des méthodes et fournir des options pour fournir aux autres véhicules des informations, ce qui crée le défi de la confidentialité et de la sécurité dans ce domaine d'application. De plus, un autre défi intéressant est le conflit entre les informations reçues des capteurs et ce que nous obtenons des véhicules connectés à proximité. Lequel est le plus susceptible de fournir la bonne position ? Une enquête devrait également avoir lieu dans ce domaine.

CHAPTER 1

INTRODUCTION

1.1 Context and challenges

In the last decade, self-driving vehicles have shown impressive progress, with many thousands of researches working in different laboratories, companies, and experimenting in various environmental scenarios. Some of the questions in the minds of their potential users relate to the advantages that are offered by this technology and whether we can rely on it or not. To respond to these questions, one needs to look at the number of accidents and how many deaths were registered daily from non-autonomous vehicles. According to the World Health organization's official website, approximately 1.3 million people, mainly children and young adults, die yearly in vehicle crashes. Most of these accidents (93%) are known to occur in low- and middle-income countries [1]. Human behaviors including traffic offenses, driving while under the influence of alcohol or psychoactive substances, and late reactions, are the major causes of crashes. Consequently, self-driving technology has been conceived and developed to replace human-vehicle interaction. Despite some of the challenges of self-driving vehicles, the surveys in [2] have indicated that about 84.4% of people were in favour of using self-driving vehicles in future. Sale statistics also state that the turnover of these vehicles was $54 billion in 2021) [3]. This outcome does demonstrate the growing confidence in the usage of this technology. It has also been suggested that this technology is capable of reducing pollution, energy, time consumption, accidents' rates while also ensuring the safety and comfort of its users [4].

The Society of Automotive Engineers (SAE) divides the technical revolution resulting

from this technology into five levels. The first two levels do not provide many automatic services, and therefore have a low risk potential. The challenge does starts from level three, with almost all companies tending to adopt it. Indeed, level three is a mixed work, where the system is prepared to avoid collisions, has assisted steering wheels, braking, etc. On the other part, humans must be prudent for any hard decision or alert from the system. Levels four and five refer to vehicles that are required to take the responsibility of driving without any human interaction. Hence, these levels are more complex, especially if one was to consider how dynamic and unpredictable our environment can be, where there may be a chance of collision from anywhere.[4] affirmed that no vehicle achieved robust results in these levels at the time of writing this paper.

Creating a self-driving vehicle is a complicated task, where it is necessary to perform predefined steps consistently and to achieve the stated goals in a robust manner. The principal question that should be answered is: How can vehicles work effectively without human interaction? Vehicles need to continuously sense the environment, hence, they should be equipped with accurate sensors like GPS, IMU, Wheel odometry, Lidar, radar, cameras, etc. Moreover, a performant computer is needed to handle the immense volume of data collected and accurately extract relevant information. This information will help the vehicles make better decisions and execute them by specific units prepared for this purpose.

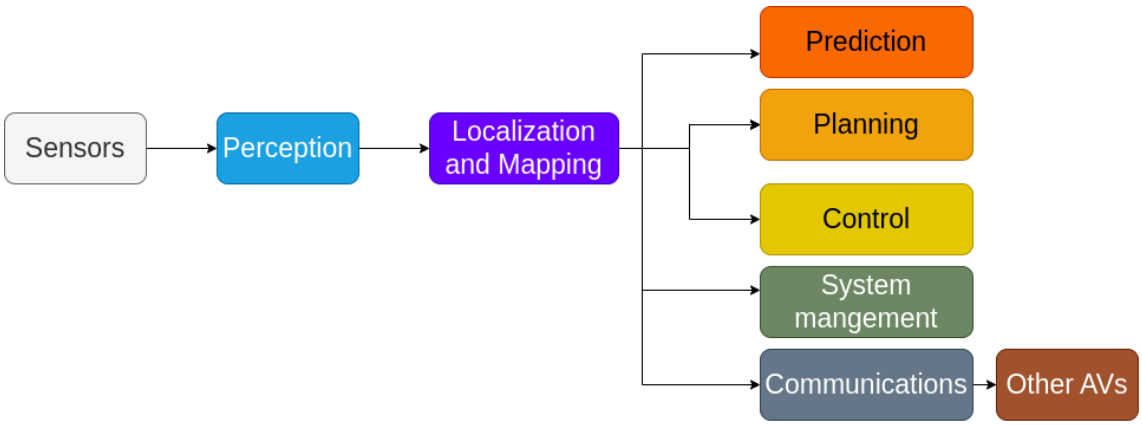


Figure 1.1: Steps to follow to achieve a self-driving vehicle.

To achieve autonomous driving, [5] proposed an architecture that combines Localisation, Perception, Planning, Control, and System Management. Localization is deemed an

indispensable step in the process of creating autonomous systems, since we need accurate information about the location.

In this thesis, we focus on the ascpect of Localization and mapping of self-driving vehicles which hold significant importance in the overall deployment and performance of autonomous systems. Accurate localization enables self-driving vehicles to understand their position and orientation in the surrounding environment, allowing them to navigate safely and effectively. Localization is essential for various self-driving tasks, including path planning, which involves finding the optimal route to a desired destination. Precise localization information is necessary for calculating the shortest and safest paths, considering factors such as traffic conditions, road rules, and potential obstacles. Without accurate localization, path planning decisions may be compromised, leading to inefficient routes, delays, or even unsafe maneuvers. Reliable localization is also crucial for facilitating effective communication between autonomous vehicles (AVs). Localization enables vehicles to accurately determine their relative positions to each other, supporting cooperative maneuvers, such as merging, lane changes, or coordinated movements at intersections. By sharing precise localization information, AVs can coordinate their actions, enhancing traffic flow and overall safety. Figure.1 illustrates the importance of localization and mapping.

Self-driving vehicles traditionally rely on Global Navigation Satellite Systems (GNSS) for localization. However, due to the serious issues associated with GNSS, there is a need to explore alternative approaches that utilize other vehicle sensors. While GNSS provides a cost-effective method for determining position, it faces challenges such as signal blockage, errors exceeding 3 meters, multipath interference, and limitations in urban and indoor environments. Additionally, GNSS is vulnerable to signal degradation, jamming, and spoofing, compromising the reliability and safety of self-driving vehicles.

The main contribution of this thesis is to overcome these limitations by shifting towards leveraging other vehicle sensors, such as cameras, LiDAR, radar, and inertial sensors. These sensors offer the potential for improved accuracy, reliability, and safety in challenging environments, including urban areas, indoor spaces, and regions with limited satellite coverage. By integrating data from multiple sensors, self-driving vehicles can achieve robust localization, reducing dependence on GNSS and enhancing overall performance in various driving conditions. [7].

1.2 Objectives and Scopes of this thesis

As mentioned earlier, there are instances where the Global Navigation Satellite System (GNSS) fails to accurately locate autonomous vehicles (AVs) due to obstacles obstructing the line of sight between the vehicles and the satellites. In Figure 1.2, we showcase scenarios where the transmitted signals are interrupted by environmental obstacles. This issue disrupts the functioning of AVs and can have severe consequences, including potential accidents, as the vehicle loses its ability to accurately perceive its position. Consequently, the vehicle becomes unable to execute tasks correctly, such as path planning, navigating around other vehicles, and even staying on the correct side of the road.

Researchers have encountered this issue and proposed various methods and approaches that rely on different vehicle equipment, such as cameras, LiDAR, radar, inertial navigation systems (INS), and wheel odometry. Our focus in this thesis will be on fusing information from different sensor equipment. However, some methods rely on a single sensor for vehicle localization, such as using dead reckoning with an inertial measurement unit (IMU) or wheel odometry. Nevertheless, traditional methods have limitations as they rely solely on one source of information, and their performance is relatively dependent on the environment's structure.

Dealing with multiple source of information require a pre processing of the data which include the way how we convert the received information to a useful input and how to fuse them together in one method. In particular, huge information like Lidar or camera measurement require a features extraction process in order to extract only what it is useful or/and sufficient for the efficient deployment of the method

Based on the discussion above, Our focus in this thesis wil involve the investigation of:

- Perform a comprehensive state-of-the-art review encompassing feature extraction, mapping, and localization methods. This includes conducting a thorough analysis and evaluation of existing approaches and techniques in the field of AV localization, with a focus on feature extraction methodologies, mapping algorithms, and localization methods. By synthesizing and critically assessing the current state of research, this thesis aims to provide a comprehensive understanding of the advancements, challenges, and opportunities in the field, thus guiding the development of novel and effective AV localization techniques.

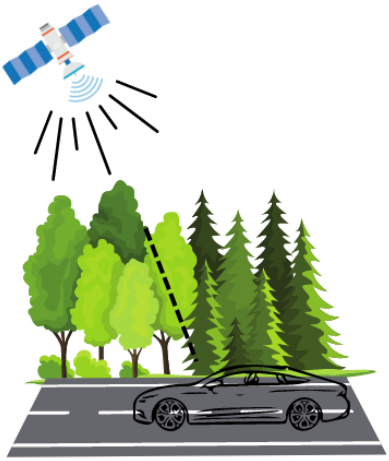
- Develop and evaluate state-of-the-art methods for fast and accurate feature extraction from vehicle sensors. The focus will be on incorporating the latest advancements in sensor technology to enhance the process of extracting relevant information.
- Explore and utilize stochastic and probabilistic techniques to improve the localization of autonomous vehicles (AVs). This involves investigating probabilistic models and algorithms that can effectively handle uncertainties and variations in sensor measurements for robust and reliable AV localization.
- Investigate optimization approaches and algorithms to enhance AV localization. This includes exploring optimization techniques such as nonlinear optimization, convex optimization, or metaheuristic algorithms to improve the accuracy and efficiency of AV localization algorithms.
- Investigate the application of machine learning techniques for AV localization. This involves exploring the use of machine learning algorithms, such as deep learning or reinforcement learning, to leverage large amounts of sensor data for improved AV localization performance.
- Conduct a comprehensive investigation of the mathematical theories and principles underlying the proposed methods. This includes analyzing the mathematical foundations, understanding the theoretical underpinnings, and assessing the applicability of mathematical concepts in the context of AV localization.

1.3 Contributions of this Thesis

In order to achieve the self-driving localization and mapping objectives discussed above, this thesis presents those contributions summarized below:

- **Localisation and Mapping of Self-driving Vehicles based on Fuzzy K-means Clustering: A Non-semantic Approach:**

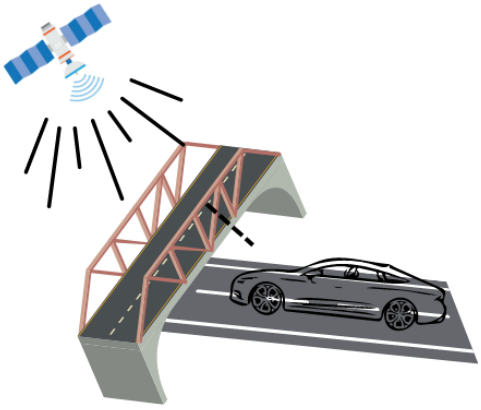
  The main contribution of this approach is the proposal and demonstration of a new architecture for localization and mapping in autonomous vehicles using non-semantic features from LiDAR scenes. The architecture involves the use of the Fuzzy K-means approach to extract features, ensuring their identification in each frame. A global mapping is achieved using the Gaussian Mixture Model (GMM) to facilitate



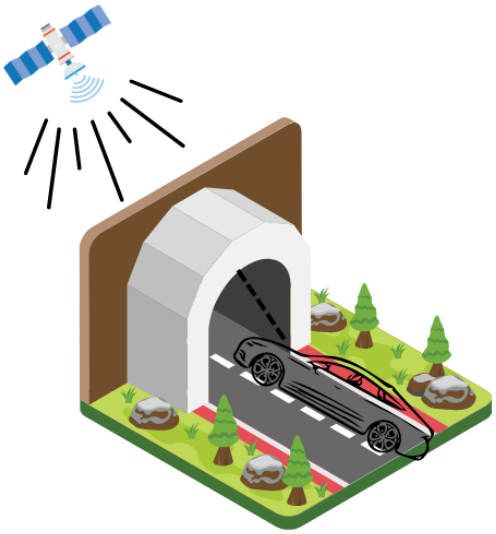
(a) Dense trees



(b) Urban canyon



(c) under the bridge



(d) Inside the tunnel

Figure 1.2: Example signal interference from environmental factors

data association between frames, aiding accurate localization through the particle filter. The proposed technique is evaluated against other state-of-the-art methods using various sequences from the Kitti raw dataset, considering different environmental structures, weather conditions, and seasonal changes. The results show that the proposed approach outperforms others in terms of speed and representation of features required for real-time localization, while also achieving competitive accuracies.

- **Fast and accurate localization and mapping method for self-driving vehicles based on a modified clustering particle filter:**

  The main contribution of this approach is the presentation of an architecture for perception in self-driving systems, focusing on localization and mapping. The architecture utilizes non-semantic features extracted from Lidar measurements and consists of two main phases: mapping and localization. In the mapping phase, a global map of the environment is created using a fuzzy c-means algorithm, deviating from previous approaches that relied on semantic algorithms. The remaining clusters are then filtered using the DBSCAN algorithm. In the localization phase, a particle filter architecture involves motion update, measurement update, and resampling to estimate positions. The novel extension introduced in this work consists of the selection of particles to reduce computational time and maintain long-term localization reliability. Two approaches, clustering with the k-means algorithm and the sigma points algorithm, are compared to identify the most effective one. The proposed method is thoroughly tested on short sequences from the Kitti dataset and the Pandaset dataset. Additionally, it is evaluated against recent methods using a long sequence dataset. The analysis demonstrates the speed of the proposed method and its ability to capture the features necessary for real-time localization, surpassing the performance of well-known localization methods.

- **Rapid Localization and Mapping Method Based on Adaptive Particle Filters:**

  The main contribution of this approach is the extension of a localization and mapping architecture for autonomous vehicles, particularly in GPS-denied environments such as tunnels, bridges, urban canyons, and dense tree canopies. The proposed approach consists of two parts. Firstly, a K-means algorithm is used to extract features from LiDAR scenes, creating local maps for each scan. These local maps are then

concatenated to form a global map of the environment, facilitating data association between frames. Secondly, the main localization task is performed using an adaptive particle filter in four steps: (a) generating particles around an initial state provided by GPS, (b) updating particle positions based on vehicle motion obtained from an inertial measurement device, (c) selecting the best candidate particles based on match rates between the local map and real-time distances to objects, as well as distances to corresponding parts of the global map, and (d) averaging the selected particles to estimate the position and employing a resampling method to ensure position estimation reliability. The performance of the proposed technique is evaluated using different sequences from the Kitti and Pandaset raw datasets, encompassing diverse environmental setups, weather conditions, and seasonal changes. The results obtained validate the effectiveness of the proposed approach in terms of speed and the representativeness of feature extraction for real-time localization, surpassing other state-of-the-art methods.

- **XDLL: Explained Deep Learning LiDAR-Based Localization and Mapping Method for Self-Driving Vehicles:**

  The main contribution of this passage is the proposal of a framework architecture for explaining deep learning LiDAR-based (XDLL) models used for positioning self-driving vehicles. The framework addresses the limitations of global navigation satellite systems (GNSS) in challenging environments such as bridges, tunnels, and dense forests. Instead, it utilizes LiDAR scans and extracts non-semantic features using a clustering algorithm. These identified clusters serve as input to a deep learning model that incorporates LSTM and GRU layers to store trajectory points and convolutional layers to smooth the data. The proposed model is extensively tested using short- and long-term trajectories from benchmark datasets, namely Kitti and NCLT, which encompass different environmental scenarios. Additionally, the obtained results are analyzed using explainable methods such as Saliency, SmoothGrad, and VarGrad to understand the contribution of each cluster feature. The analysis demonstrates that taking the mean of all the clusters as an input for the model improves accuracy compared to the initial model and reduces time consumption. Remarkably, the improved model achieves a mean absolute positioning error of less than one meter for all sequences in both short- and long-term trajectories.

- **Localization and Mapping for Self-Driving Vehicles: A Survey:**

The main purpose of this survey is to provide a comprehensive overview of the current approaches in the field of localization and mapping for autonomous vehicles. The review highlights the importance of these tasks in achieving robust navigation and better driving experiences. The survey acknowledges the lack of meaningful and comprehensive descriptions in recent studies, and aims to address this gap by examining the most recent methods in mapping, localization, feature extraction, and data security. The survey begins by discussing contemporary methods of extracting relevant features from sensors and categorizes them into semantic, non-semantic, and deep learning approaches. It emphasizes the importance of representativeness, low cost, and accessibility in selecting appropriate methods for localization and mapping tasks. The focus then shifts to methods for building the vehicle's environment map, considering both commercial and academic solutions. The survey distinguishes between known and unknown environments, offering suitable solutions for each case. Additionally, the survey explores different approaches to vehicle localization and classifies them based on their mathematical characteristics and priorities. Each section concludes by addressing the related challenges and presenting potential future directions for research and development. Finally, the survey concludes by highlighting the security challenges that may arise in self-driving vehicles and assesses potential defense mechanisms to prevent security attacks. This demonstrates the survey's comprehensive coverage of the problems and considerations surrounding localization, mapping, and data security in autonomous vehicles.

- **Autoencoder-based LiDAR Localization for Improved Navigation in Self-Driving Vehicles:**

  This paper focuses on the crucial role of localization systems in enabling successful navigation for self-driving vehicles. It highlights the limitations faced by global navigation satellite systems in certain environments, such as tunnels, bridges, and urban canyons. To address these challenges, the paper proposes an autoencoder-based method for predicting a vehicle's position using LiDAR point cloud data, without the need for additional sensors. The proposed model modifies the objective function of the autoencoder by introducing an additional objective function responsible for adapting the output of the encoder layer (latent space) to predict the position accurately. The paper also incorporates a penalty parameter to emphasize learning in the latent space. The convergence of the proposed objective function is proven, establishing it as a stochastic-local-quasi-convex function and

ensuring the effectiveness of the stochastic natural gradient descent algorithm in achieving convergence. The model is evaluated using the NCLT dataset in PyTorch, with performance assessment conducted for different values of the penalty parameter, denoted as α. The results demonstrate that $\alpha = 0.01$ yields the best performance. Compared to various advanced registration techniques, the proposed method achieves excellent results in the majority of KITTI sequences. Notably, the method's advantage lies in its ability to make direct predictions without requiring iterative optimization, which is a common step in registration techniques. The method achieves a desirable mean absolute error of approximately 0.5, highlighting its effectiveness for autonomous-driving vehicles. Overall, the proposed method offers the potential to enable more accurate and efficient navigation for autonomous vehicles, addressing the limitations of existing localization systems and providing promising results in terms of prediction accuracy and efficiency.

1.4 Organization of this Thesis

The thesis is structured into several chapters, each addressing specific aspects of localization and mapping for self-driving vehicles.

- **Chapter 1**: serves as the introduction, providing an overview of the thesis, outlining its objectives and scopes, and highlighting its contributions to the field.
- **Chapter 2** is devoted to the literature review, which thoroughly explores existing research and technologies. It discusses various studies and advancements in the field
- **Chapter 3** present our contribution to extract relevant features for localization and mapping purpose.
- **Chapter 4** present our contribution to tackle the localization and mapping problem based on a modified clusterning particle filter.
- **Chapter 5** present our contribution to tackle the localization and mapping problem based on an adaptative particle filter.
- **Chapter 6** present our contribution to tackle the localization and mapping problem based on a deep learning model.

- **Chapter 7** present our contribution to tackle the localization and mapping problem based on a modified autoencoder .
- **Chapter 8** serves as the conclusion, summarizing the findings, discussing the challenges encountered, and providing suggestions for future research directions in the domain of localization and mapping for self-driving vehicles.

1.5 List of publications

- A. Charroud, A. Yahyaouy, K. E. Moutaouakil and U. Onyekpe, "Localisation and Mapping of Self-driving Vehicles based on Fuzzy K-means Clustering: A Non-semantic Approach," 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 2022, pp. 1-8, doi: 10.1109/ISCV54655.2022.9806102.
- Charroud, A., Moutaouakil, K.E. & Yahyaouy, A. Fast and accurate localization and mapping method for self-driving vehicles based on a modified clustering particle filter. Multimed Tools Appl 82, 18435–18457 (2023). https://doi.org/10.1007/s11042-022-14111-4
- Charroud, A., El Moutaouakil, K., Yahyaouy, A., Onyekpe, U., Palade, V., & Huda, M. N. (2022). Rapid Localization and Mapping Method Based on Adaptive Particle Filters. Sensors, 22(23), 9439. https://doi.org/10.3390/s22239439
- Charroud, A., El Moutaouakil, K. E., Palade, V., & Yahyaouy, A. (2023). XDLL: Explained Deep Learning LiDAR-Based Localization and Mapping Method for Self-Driving Vehicles. Electronics, 12(3), [567]. https://doi.org/10.3390/electronics12030567
- El Moutaouakil, K., Palade, V., Safouan, S., & Charroud, A. (2023). FP-Conv-CM: Fuzzy Probabilistic Convolution C-Means. Mathematics, 11(8), 1931. https://doi.org/10.3390/ma
- El Moutaouakil, K. E., El Ouissari, A. E., Palade, V., Charroud, A., Olaru, A., Baïzri, H., Chellak, S., & Cheggour, M. (2023). Multi-Objective Optimization for Controlling the Dynamics of the Diabetic Population. Mathematics, 11(13), 2957. https://doi.org/10.3390/math11132957
- Localization and Mapping for Self-Driving Vehicles: A Survey (Submitted)
- Autoencoder-based LiDAR Localization for Improved Navigation in Self-Driving Vehicles (Revised- Major revision)

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this comprehensive survey, we have delved into the intricacies of safe localization and navigation by exploring key terms such as feature extraction, mapping, and localization processing. Our aim was to provide a deeper understanding of these concepts and their implications for ensuring reliable and accurate vehicle localization.

Features play a vital role in the mapping and localization process as they represent the landmarks or essential elements that vehicles need to identify in their environment. These features can include objects like poles, curbs, buildings, as well as intensity variations. The quality and relevance of these features directly impact the accuracy of mapping and, consequently, vehicle localization. Through our investigation, we have introduced a novel classification framework that encompasses semantic, non-semantic, and deep learning approaches, enabling a clearer comprehension of these methods.

Our survey has led us to the conclusion that non-semantic features prove to be more suitable for solving localization and mapping tasks. While semantic features have their merits, such as providing a rich contextual understanding, they can be time-consuming to extract and may not be present in all environmental scenarios. Furthermore, the reliance on semantic features can potentially introduce errors in the matching process. Deep learning methods, on the other hand, offer excellent capabilities for representing the environment, either semantically or not, depending on the chosen approach. However,

the computational demands associated with deep learning can pose challenges in their execution.

Moving on to the mapping stage, we have examined various techniques, including pre-built and online mapping approaches, while considering the underlying mathematical principles. Our research highlights the necessity of frequent map updates to account for changes in the environment. Furthermore, selecting the most appropriate features to include in the map becomes a crucial consideration, as it requires striking a balance between utilizing sufficient features for scene representation without overwhelming memory storage capacity.

The final parameter we sought to demystify is localization, which involves determining the precise position of vehicles using different sensors. Our investigation has covered a wide range of methods, such as probabilistic approaches, optimization techniques, and cooperative localization. These approaches have proven to be more robust and capable of delivering accurate results compared to classical methods. We have provided concise descriptions of each approach and explored recent papers in the literature that contribute to this field.

Lastly, we have also reviewed potential attack methods that pose threats to autonomous vehicle (AV) systems in general, shedding light on the importance of ensuring robust security measures in conjunction with localization and mapping advancements.

By examining features extraction, mapping, and localization processing, this survey serves as a comprehensive resource for researchers, practitioners, and stakeholders involved in the development and deployment of autonomous vehicles.

2.2 Features extraction method for vehicle localization purpose

Feature extraction is the process of providing a relevant representation of the raw data. In other words, it is manipulating data to be easy to learn from models while being more robust to generalize. It is the same here with sensors' data. Extracting suitable information is very useful, and it will reduce cost, energy and aid the models to be fast and accurate in the execution. In this section, we give an overview of what was done previously in the area. However, we limit the discussions to two sensors, namely, Lidar

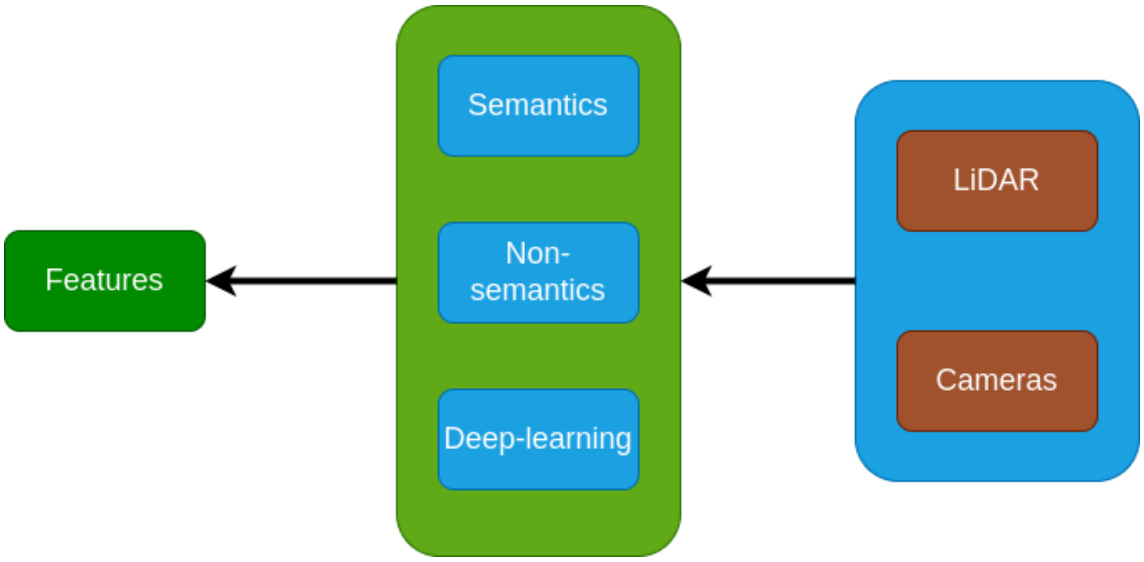


Figure 2.1: Flowchart of feature extraction

and cameras, because we believe that they are more suitable and useful in practice. This part also provides a clear and easy architecture to understand these features. See the flowchart in figure.2.

2.2.1 Semantic features

Semantic features essentially deals with extracting a known target (poles, facades, curbs, etc.) from the raw data, in order to localize the vehicle more accurately.

Lidar features. [21] have extracted lane marking as a feature to localize the vehicle by using the Radon transform to approximate polylines of the lane. Then, they have applied the Douglas - Peukers algorithm to refine the shape points that estimate the road center lines. In the work carried out by J. Im et .al [22], attempts were being made to detect two kinds of features. The first category was produced using lidar reflectivity from layer 17 to 32 to extract road marking. Indeed, a binarization layer was applied on the extracted ground plane. This binarization is done by using the Otsu thresholding method [23]. Finally, they used the Hough transform to infer road marking lines. On the other hand, the second category focuses on the extraction of the building walls because they resemble lines in the 2D horizontal plane. So, they have projected all these features into a 2D probabilistic map. However, the existence of the poles like street trees affects the creation of the map of lines. To solve this problem, unnecessary structures were eliminated by applying the Iterative-End-Point-Fit (IEPF) algorithm. This is considered as one of the

most reliable algorithms for line fitting, and it gives the best accuracy in a short calculation time. Zhang et al. [24] explored the height of curbs between 10-15cm, and the smoothness and continuity of points cloud around the curbs to propose a method to track curbs. This approach is also used in [25] with some modifications. The existence of curbs on almost every road could be regarded as very useful. The authors of [26] have used the combination of information from Lidar and cameras to extract poles like traffic signs, trees, and street lamps, since they are more representative for localization purposes. The authors have projected, in each layer, the point clouds in a 2D grid map after removing the ground plan, and they assumed that the connected grid cell of points, in the same height, is supposed to be a candidate object. To ensure the reliability of this method, they have tried to create a cylinder for each landmark candidate and form the pole's shapes. From the camera, they have attempted to calculate a dense disparity mapping and apply pattern recognition on the result of the map to detect landmarks. Kummerle et al. [27] adopts the same idea as [26], where they attempted to extract poles and facades from the Lidar scans. They also detected road marking by a stereo camera. In [28] the authors extracted, as the first kind of features, building facades, traffic poles, and trees by calculating the variance of each voxel, then, they checked the size of a fixed threshold, and concluded on the closeness of the points. Those voxels are grouped and arranged in the same vertical direction. Hence, each cluster that respects the variance condition in each voxel is considered as a landmark. The second kind is interested in the reflectivity of the ground, which is extracted with the RANSAC algorithm from lidar intensity. A performant algorithm was made in [29] to detect poles. Firstly, a 3D voxel map was applied on the set of points cloud; then, they eliminated cells with fewer points. Thereafter, they determined the cluster's boundaries by calculating the intensity of points in the pole's shape candidate to deduce the highest and lowest parts. Finally, they checked if the points clouds in the entire core of the pole candidates did satisfy the density condition to be considered as pole candidates. [30] performed a method with three steps: voxelization, horizontal cluster, and vertical cluster (voxelization is a preprocessing step). Typically, it is intended to remove the ground plan and regroup each part of the points cloud into voxels. Then, pic the voxels with the number of points greater than a fixed threshold. Also, they exploited pole characteristics in an urban area, like the isolation, distinguishable from the surrounding to extract horizontal, vertical features. [31] is an interesting article that provides highly accurate results of detecting poles landmarks. Indeed, the authors have used a 3D probabilistic occupancy grid map based mainly on the Beta distribution applied for each layer. Then, they have calculated the difference between the average

occupancy value inside the pole and the maximum occupancy value of the surrounding poles, which is a value between $[-1, 1]$. Greater value means higher probability that it is a pole. A GitHub link for the method was provided in [31]. [32] is another method repose on poles landmarks. It is a high-precision method to localize self-driving vehicles. The method assumed five robust conditions; common urban objects, time-invariant location, time-invariant appearance, viewpoint-invariant observability, and high frequency of occurrence. First, they removed the ground plan using the M-estimator Sample Consensus (MSAC). Secondly, they performed an occupied grid map, which isolates the empty spaces. Afterwards, they performed clustering with the connected component analysis (CCA) to group the connected occupied grid into an object. Finally, PCA was used to infer the position of the poles. The idea is to use the obtained eigenvalues in these three conditions.

$$\lambda_z \gg \lambda_x, \lambda_y > 0$$
$$\lambda_x \approx \lambda_y$$
$$\frac{\lambda_z}{\min(\lambda_x, \lambda_y)} \leq s_{th}$$

These conditions formed precisely the dimensions of poles. The article [33] has detected three kinds of features. First, planar features, like walls, building facades, fences, and other planar surfaces. Second, pole features include streetlights, traffic signs, tree trunks. Third, curbs shapes. After removing the ground plan using the Cloth Simulation Filter, the authors have concluded that all landmarks are vertical and tall above the ground plan and contain more points than its surrounding. They filtered cells that do not respect the above conditions. Planar features, curb features are considered as lines that could be approximated by a RANSAC algorithm or running a Hough transform.

Cameras features. Due to its low cost, and lightweight, the camera is an important sensor widely used by researchers. [13] divided cameras into 4 principal types: monocular camera, stereo camera, RGB-D camera, and event camera. Extracting features from the camera sensor is another approach that should be investigated. [8] is one of the best works done in visual-based localization where the authors have given an overall idea about the recent trend in this field; they divide features into local and global features. Local features search to extract precise information from a local region in the image, which is more robust for the generalization task and more stable with image variations. Secondly, global features focus on the extraction features from the whole image. These features

could either be semantical or not. Most researchers working with semantics try to extract contours, line segments, and objects in general. In [34], the authors extracted two types of linear features, edges, and ridges by using a Hough transform. Similarly, authors of [35] have used a monocular camera to extract lanes' markings and pedestrian crossing lines. Polylines approximate these features. On the other hand, the article [36] performed a visual localization using cameras, where they brought out edges from environment scenes as a features input. Each image input is processed as follows: the first step consists of solving the radial distortion issue, which is the fact that straight lines bend into circular arcs or curves [37]. This problem can be solved by the radial distortion correction methods and projective transformation into to the bird's eye view. Moreover, they calculated the gradient magnitude, which creates another image called gradient image split into several intersecting regions. After that, a Hough transform is applied to refine the edges of the line segments. Finally, a set of conditions depending on geometric and connectivity of the segment was checked to find the last representation of the edge polylines. Performing a localization for mobile devices is done in [38] by using features that satisfy some characteristics, such as: Permanent (statics), Informative (distinguishable from others), Widely available. From the features, we can derive: alignment tree, autolib station, water fountain, street lights, traffic lights, etc.

2.2.2 Non-Semantic features

Lidar features. In the article [39], the method consists of four main steps. They started with preprocessing, which aligns each local neighborhood to a local reference frame using Principal Component Analysis (PCA). The smallest principal component is taken as a normal. Secondly, Pattern Generation is where the remaining local neighborhood points are transformed from 3D into 2D within a grid map; each cell contains the maximum value of reflectivities. Furthermore, the Descriptor Calculation is performed by the DAISY descriptor, which works by convolving the intensity pattern by a Gaussian kernel. After that, a gradient of intensities is calculated for eight radial orientations. Moreover, a smoothening was performed using Gaussian kernel of three different standard deviations. Finally, a normalization step is applied to maintain the value of the gradients within the descriptor. 'GRAIL' is able to compare the query input with twelve

distinctive shapes that can be used as relevant features for localization purposes. Hungar et al. [40] used a non-semantical approach where a reduction of the time of execution by selecting points whose sphere of the local neighborhood with radius r, including an amount of point that will exceed a fixed threshold was carried out. After that, they distinguish between the remaining patterns into curved and non-curved by using k-medoids clustering. The authors use a DBSCAN clustering to aggregate similar groups to infer features. Lastly, the creation of the key features and map features relied on different criteria, including Distinctiveness, Uniqueness, Spatial Diversity, Tracking Stability, Persistence. A 6D pose estimation was performed by [41], where they describe the vehicle's roadside and considered it a useful feature for the estimation model. To achieve this, they proposed preprocessing the point cloud through ROI (Region of Interest) filtering in order to remove long-distance background points. After that, a RANSAC algorithm was performed to find the corresponding equation of the road points. Thanks to the Radius Outlier Removal filter, which the authors used to reduce the noise by removing isolated points and reducing their interference. Meanwhile, the shape of the vehicles was approximated with the help of the euclidean clustering algorithm presented in [42]. Charroud .A et al [43], [44] have removed the ground plan of all the LiDAR scan to reduce the huge amount of points, and they have used a Fuzzy k-means clustering technique to extract relevant features from the LiDAR scan. An extension of this work [45] add a downsampling method to speed up the calculation process of the Fuzzy k-means algorithm.

Descriptors methods can be considered an interesting idea since they are widely used to extract meaningful knowledge directly from a set of points' cloud. These methods do enable a separation of each feature point regarding the perturbation caused by noise and varying density, and the change of the appearance of the local region [46]. They also added that four main criteria should be used while performing features description, citing the descriptiveness, robustness, efficiency, and compactness. These criteria are riddled with reliability, representativity, cost of time and storage space. These descriptors can be used in the points pair-based approach, which is the task of exploiting any relationship between two points like distance or angle between normal, or boundary to boundary relations, or relations between two lines [47]. Briefly, with respect to this methods, we can cite CoSPAIR [48], PPFH [49], HoPPF [50], PPTFH [51]. Alternatively, they can be included to extract local features. For example, TOLDI [52], BRoPH [46], and 3DHoPD [53]

Cameras features. Extracting non-semantic features from the camera sensor is widely

treated in the literature. However, this article focuses on methods that work to find pose estimation. Due to the fastness of the ORB compared to other descriptors like SIFT or SURF, many researchers have adopted this descriptor to represent image features. For this purpose, [54] attempted to extract key features by using an ORB descriptor, to achieve an accurate matching between the extracted map and the original map. Another matching features' method was proposed in [55], where the authors sort to extract holistic features from front view images by using the ORB layer and BRIEF layer descriptor to find a great candidate node, while local features from downward view images were detected using FAST-9, which are fast enough to cope with this operation. Gengyu et al. [56] extract ORB features and converted them to visual words based on DBoW3 [57]. [58] used ImageNet-trained CNN features (more details in the next sub-section) to extract object landmarks from images, as it is considered to be more powerful for localization tasks. Then they extract SIFT features for further features' improvement.

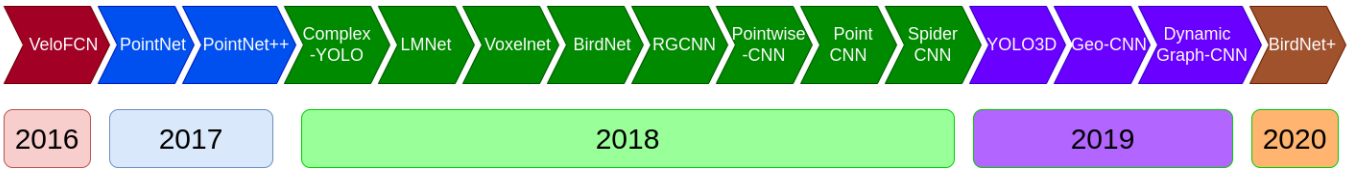


Figure 2.2: Timeline of 3D object detection algorithms [11].

2.2.3 Deep learning features

It is worth mentioning the advantage of working with deep learning methods, which try to imitate the working process of the human brain. Deep learning (DL) is widely spread in various application domains, such as medical, financial, molecular biology, etc. It can be used for different tasks, such as object detection and recognition, segmentation, classification, etc. In the followings, we survey recent papers that use DL for vehicle localization and mapping.

Lidar and/or Cameras features. One of the most interesting articles done in this field is [11], where the authors provided an overall picture of object detection with lidar sensors using deep learning. This survey divides the state-of-art algorithms into three parts:

projection-based methods, which project the set of points cloud into a 2D map respecting a specific viewpoint; voxel-based methods, which involve the task of making data more structured and easier to use by discretizing the space into a fixed voxel grid map; Finally, point-based methods, which work directly on the set of points' cloud. Our literature investigation concludes that most of the methods are based on the CNN architecture with modifications in the preprocessing stage.

- **MixedCNN-based Methods**: CNN or (Convolutional neural network) is one of the most common methods used in computer vision. These types of methods use mathematical operations called 'convolution' to extract relevant features [59]. VeloFCN [60] is a projection-based method and one of the earliest methods that uses CNN for 3D vehicle detection. The authors used a three convolution layer to down-sample the input front of the view map, then up-sample with a deconvolution layer. The output of the last procedure is fed into a regression part to create a 3D box for each pixel. Meanwhile, the same results were entered for classification to check if the corresponding pixel was a vehicle or not. Finally, they grouped all candidates' boxes and filtered them by a Non-Maximum Suppression (NMS) approach. In the same orientation, LMNet [61] increased the zone of detection to find road objects by taking into consideration five types of features: reflectance, range, distance, side, height. Moreover, they change the classical convolutional by the dilated ones. The Voxelnet [62] method begins with the process of voxelizing the set of points cloud and passing it through the VFE network (explained below) to obtain robust features. After that, a 3D convolutional neural network is applied to group voxels features into a 2D map. Finally, a probability score is calculated using an RPN (Region Proposal Network). The VFE network aims to learn features of points by using a multi-layer-perceptron and a max-pooling architecture to obtain point-wise features. This architecture concatenates features from MLP output and MLP + Maxpooling. This process is repeated several times to facilitate the learning. The last iteration is fed to an FCN to extract the final features. BirdNet [63] generates a three-channel bird eye's view image, which encodes the height, intensity, and density information. After that, a normalization was performed to deal with the inconsistency of the laser beams of the LiDAR devices. BirdNet uses a VGG16 architecture to extract features, and they adopt a Fast-RCNN to perform object detection and orientation. BirdNet+ [64] is an extension of the last work where they attempted to predict the height and vertical position of the centroid object in addition to the

processing of the source (BirdNet) method. This field is also approached by transfer learning, like in Complex-YOLO [65], and YOLO3D [66].
Other CNN-based method include regularized graph CNN (RGCNN) [67], Pointwise-CNN [68], PointCNN[69], , Geo-CNN [70], Dynamic Graph-CNN [71] and Spider-CNN [72]

- **Other Methods**: These techniques are based on different approaches. [73] is a machine learning-based method where the authors try to voxelize the set of points cloud into 3D grid cells. They extract features just from the non-empty cells. These features are a vector of six components: mean and variance of the reflectance, three shape factors, and a binary occupancy. The authors proposed an algorithm to compute the classification score, which takes in the input of a trained SVM classification weight and features, then a voting procedure is used to find the scores. Finally, a non-maximum suppression (NMS) is used to remove duplicate detection. Interesting work is done in [74], who tried to present a new architecture of learning that directly extracts local and global features from the set of points cloud. The 3D object detection process is independent of the form of the points cloud. PointNet shows a powerful result in different situations. PointNet++ [75] extended the last work of PointNet, thanks to Furthest Point Sampling (FPS). The authors created a local region by clustering the neighbor point and then applied the PointNet method in each cluster region to extract local features.

Figure.3 represent a timeline of the 3D object detection algorithm.

2.2.4 Discussion

Table 2 provides a categorization for the surveyed papers in this section. The papers were grouped regarding the features extracted. We noticed that the papers extracted three kind of features objects from the environment: vertical, horizontal, or road curves features in case of semantic type. Also, papers have used non-semantic and deep learning methods to represent any kind of objects that exist in the environment or to represent just a part of them. Moreover, the table provides some methods and concepts used to extract the features. We have analysed the robustness of the extracted features to help the localization and mapping tasks by using three criteria deduced from our state-of-art investigation.

Table 2.1: Categorization of the state-of-art methods to extract relevant features for localization and mapping purposes.

Paper	Features type	Concept	Methods	Features-extracted	Time and energy cost	Representa-tiveness	Accessibility	Robustness
[21], [22], [33], [34], [35], [36].	- Semantic	- General	- Radon transform - Douglas & Peukers algorithm - Binarization - Hough transform - Iterative-End-Point-Fit (IEPF) - RANSAC	- Road lanes - lines - ridges Edges - pedestrian crossings lines	- Consume a lot	- High	- Hard	- Passable
[24], [25], [33].	- Semantic	- General	- The height of curbs between 10cm and 15cm - RANSAC	- Curves	- Consume a lot	- High	- Hard	- Passable
[26], [27], [28], [29], [30], [31], [32], [33], [38].	- Semantic	- Probabilistic - General	- Probabilistic Calculation - Voxelisation	- Building facades - Poles	- Consume a lot	- Middle	- Hard	- Low
[39], [40], [41], [43], [44],[45], [48], [49], [50], [51], [52], [46], [53], [54], [55], [56], [57].	- Non-Semantic	- General	- PCA - DAISY - Gaussian kernel - K-medoids - K-means - DBSCAN - RANSAC - Radius Outlier Removal filter - ORB - BRIEFFAST-9	- All the environment	- Consume less	- High	- Easy	- High
[58], [59], [60], [61], [62],[63], [64] , [65], [66], [67], [68],[69], [70], [71], [72], [73], [74], [75].	- Deep-learning	- Probabilistic - Optimization - General	- CNN - SVM - Non-Maximum Suppression - Region Proposal Network - Multi Layer Perceptron - Maxpooling - Fast-RCNN - Transfer learning	- All the environment	- Consume a lot	- High	- Easy	- High

- **Time and energy cost**: easy to detect and easy to use in terms of compilation and execution.
- **Representativeness**: detecting features that frequently exist in the environment to ensure the matching process.
- **Accessibility**: easy to distinguish from the environment.

We have used the column 'Robustness' as a score given to each cluster of papers. The score is calculated based on the three criteria above and the analysis of experiments in the papers. According to the same table, extracting non-semantic features have the highest robustness score regarding their ability to represent the environment even with less texture, i.e. in the case of few objects in the environment like in the desert. This competence is due to the way the features are extracted. Those methods do not limit themselves to extracting one type of object. However, the map created by those features will not have a meaning. they are just helpful reference points for the localization process only.

On another hand, using semantic features helps get a passable score to be used in localization tasks since they consume a bit more time and energy for executions because most of the time they are not isolated in the environment. One thing more is they can not be found in any environment which hardly affects the localization process. Although all these negative points, these techniques reduce effectively the huge amount of points data (LiDAR or Camera) compared with the non-semantic ones. Also, those features can be used for other perception tasks.

Deep learning methods also get a passable score regarding their efficiency to represent the environment. Like the non-semantic techniques, the DL approach ensures the representativeness of the features in all environments. However, the methods consume a lot of time and computational resources to be executed.

2.3 Mapping

Mapping is the task of finding a relevant representation of the environment's surroundings. This representation can be generated according to different criteria. For example, in autonomous driving systems, maps play an indispensable role in providing highly pre-

cise information about the surrounding of the vehicles, which aids in dealing with vehicles localization, vehicles control, motion planning, perception, and system management. Mapping also helps in better understanding of the environment. Maps seek information from different sensors, which are divided into three groups. The first one employs the Global Navigation Satellite System (GNSS) sensors with an HD map that relies on layers on a Geographic Information System (GIS) [18], GIS is a framework which provides the possibility of analyzing spatial and geographic data. The second group is based on the Range and vision-based sensors such as LiDAR, Cameras, RADAR, etc. that helps in the creation of a point cloud map. The third group uses cooperative approaches. Each vehicle will generate its local map and then assemble them through Vehicle-to-Vehicle (V2V) communications to generate the global map, which is lower in cost and more flexible and robust. Data received from these sensors are classified into stable objects in time, i.e., immovable, such as buildings, traffic lights, bridges, curbs, pavements, traffic signs, lanes, poles-landmarks. This information is tremendously essential to change lines, avoid obstacles, respect road traffic in general. Secondly, the dynamic objects change coordinates over time, like vehicles, pedestrians, cyclists. This information is helpful in the context of using (V2V) vehicles-to-vehicles or (V2I) vehicles-to-infrastructure communications. Other than that, it is not recommended to be mapped. Wong et al. [76] add another class named temporary objects, which are features that exist in a time period sense, like parked vehicles, temporary road works, and traffic cones. Maps are prone to be inaccurate due to errors of sensor measurements, hence, that can lead to failures in the positioning of objects on the map, which makes driving impossible in this situation. 10 cm of accuracy is recommended by Seif and Hu [77]. Moreover, the massive amount of data received from different sensors need high storage space and computational processing units. Fortunately, this can be covered by the new generation of computational capacities like NVIDIA cards. [78] discussed the required criteria for getting a robust map with respect to three aspects: storage efficiency, usability, and centimeter- level accuracy.

In the literature, we found that maps for autonomous driving systems are divided into offline maps and online maps, which will be discussed in detail below.

2.3.1 Offline maps

Offline maps (also called a priori or pre-built maps) are generated in advance to help autonomous driving systems in the navigation and localization tasks. According to [79], the turnover resulting from the their usage increased to 1.4$ billion in 2021 and is intended to reach 16.9$ billion by 2030. The number of investments in this field tends to grows daily, and the competition between maps manufacturers is at a higher level. These companies use different technologies and strategies of partnership and collaboration. In the last few years, major auto manufacturers have accelerated the efforts to produce new generations of advanced driver assistance systems (ADAS). In consequence, developing such great maps is a must for these vehicles.

According to [80], three types of maps can be identified, including digital maps, enhanced digital maps, (HD) high-definition maps. Digital maps (cartography) are topometric maps that encode street elements and depict major road structures, such as google maps, OpenStreetMap, etc. These maps also provide the possibility of finding the distance from one place to another. However, these maps are useless for autonomous vehicles due to the limited information provided, the low update frequency, and lack of connectivity. So, it cannot be accessed from external devices. The enhanced digital map is a little bit more developed than the first one. Furthermore, from what we saw, these maps are characterized by additional data information, including poles-landmarks, road curvature, lane level, metal barrier. The most important one is the HD map which is a concept developed by the Mercedes-Benz research planning workshop [81]. This map consists of a 3D representation of the environment which is broken down into five layers, i.e. Base map layer, Geometric map layer, Semantic map layer, Map priors layer, Real-time knowledge layer, as are clearly explained in [82], [80].

To build an HD map, we should take into consideration four main principles mentioned in [82],[80]. First, mapping a precomputation that facilitates the work of the autonomous driving system in real-time by solving, partially or completely, some problems in the offline stage in a highly accurate manner. The second principle is mapping to improve safety, highlighting the importance of providing accurate information about the surrounding objects to ensure safety, especially if we are talking about levels four and five autonomous vehicles, which need more maintenance and surveillance. Alternatively, a map is another sensor used at the moment of driving. The third principle is that a map should be ro-

bust enough against the dynamic objects that cause run-time trouble. Finally, a map is a global shared state working as a team; each (AV) system provides its information. So, this idea can solve many problems of the large computational memory and reduce energy, etc. Maps should be underpinned with hardware and software components. As aforementioned, hardware components are the data source of the map, while the software components are intended to analyze and manipulate, and even power the hardware [83].

Maps are prone to errors from various sources, including the change of reality, localization error, inaccurate map, map update error, etc., and we can define an accuracy metric to measure how the map is equal to the real-word. According to [84] two metric accuracy are defined:

- **Global accuracy** : (Absolute accuracy) position of features according to the face of the earth.
- **Local accuracy** : (Relative accuracy) position of features according to the surrounding elements on the road.

In Table 3, we detail some companies that are developping digital maps solutions for autonomous systems.

However, these maps are usually not publicly available, and they need to be generated frequently which arise the need to an in-house solution, which can be created with vehicles sensors and with the help of academic solutions. So, all feature extraction algorithms we have already discussed are candidates for solving localization and mapping tasks. Features that are widely used for the purpose of creating a map are lanes [21] [22] [34], buildings [22] [28] [33], curbs [24] [25] [33], poles [26] [27] [28] [29] [30] [32] [33], or a combination of these. These maps are referred to as features' maps. There is a lot of types of maps in the literature, like the probabilistic occupancy map, which is a lightweight, and low-cost map that gives the probability that a cell is occupied or not, such as the work in [31]. Voxel grid map is another type of maps based on 3D voxelization of the environment; each voxel is equivalent to a pixel in a 2D environment. Also, these maps are used most of the time to discretize 3D points cloud, which reduces the work into voxels and offers more opportunities to employ relevant algorithms [73]. M. Schreier [85] adds more map representations depending on various levels of abstractness including parametric free space maps, interval maps, elevation maps, stixel world, multi-level surface maps, raw sensor

Table 2.2: Commercial companies developing digital mapping solutions for autonomous systems.

Map	Original country	Description/Key Features
HERE	Netherlands	Here is one of the companies that provides HD maps solutions and promise its clients to ensure the safety and get the driver truth by providing relevant information to offer vehicles more option and to decide comfortably. Here uses machine learning to validate map data against the real-word in the real-time, this technology achieve around 10-20 centimeter of the accuracy in the identification of the vehicles and their surrounding. Moreover, this map contains three main layers Road Model, HD Lane Model, and HD Localisation Model [88], [89].
Tomtom	Netherlands	Support many applications in an (ADAS) like Hans-free driving, advanced lane guidance, lane split, curves speed warning, etc. Also, provide a high precise vehicle positioning with an accuracy of 1m or better compared to reality and 15cm in the relative accuracy. Also, Tomtom take the advantage of using the RoadDNA where it converts the 3D points cloud into a 2D raster image which made it much easier to implement in-vehicles. Tomtom consist of three-layer Navigation data, Planning data,RoadDNA [90], [91].
Sanborn	USA	Exploit the data received from different sensors Cameras, Lidar to generate a high-precise 3D base-map. This map attain 7-10cm of the absolute accuracy [92].
Ushr	USA/Japan	Ushr use only stereo camera imaging techniques to reduce the cost of acquiring data, and they use advanced machine vision and machine learning enable to achieve 90% of automation of data processing, also this map has mapped over 200000 miles under 10cm level of absolute accuracy [93].
NVIDIA	USA	NVIDIA map detect semantics road features and provides information about vehicles position with robustness and centimeter-level of accuracy. Also, NVIDIA offers the possibility to build and update a map using sensors available on the car [94].
Waymo	USA	Waymo affirmed that a map for (AV) contain much more information than the traditional one and deserve to have a continuous maintain and update, due to its complexity and the huge amount of data received. Waymo extract relevant features on the road from lidar sensor, which help to accurately find the vehicle position by matching the real-time features with the pre-built map features, this map attains a 10cm of accuracy without using a GPS [89].
Zenrin	Japan	Zenrin is one of the leaders' companies in the creation of map solution in japan, founded in 1948 and has subsidiaries worldwide in Europe, America, and Asia. The company adopt the Original Equipment Manufacturers (OEM) for the automobile industry. Zenrin offer 3D digital maps containing Road networks, POI addresses, Image content, Pedestrian network, etc. [95].
NavInfo	China	Explore the fusion of three modules HD-GNSS, DR engine, and Map fusion to provide high accurate positioning results. Also, they allow parking the vehicle by a one-click parking system. NavInfo is based on the self-developed HD map engine and provide a scenarios library based on HD map, simulation test platform, and other techniques [96].
Lvl5	USA	In lvl5, they believe that vehicles did not need to have a Lidar sensor, unlike Waymo. The company The company focus on cameras sensor and computer vision algorithm to analyze and manipulate videos captured and converted into a 3D map. The HD maps created change multiple times in a day which is a big advantage compared with other companies. lvl5 get an absolute accuracy in the range of 5-50cm [97],[76] .
Atlatec	Germany	To obtain a highly accurate map, Atlatec uses cameras sensors to collect data. After that, a loop closure is applied to get a consistent result. They use a combination of artificial intelligent (AI) and manual work to get a detailed map of the road objects. This map arrived to 5cm in the relative accuracy [98],[81].

data. Note that it is possible to combine the information from maps to gain a better understanding of the environment scene.

2.3.2 Online maps

The localization and mapping problems were first treated separately, and good results were reported. However, in recent years, simultaneous localization and mapping (SLAM) have emerged to cover the problem of unexciting of maps or if there is no access to the own map of the environment. This problem is more difficult because we try to estimate the map and find the position of the vehicles at the same time. For this reason, we call it online maps. In reality, there are many unknown environments, especially for the in-house robots. So we should create our map [86]. Let us look from the mathematics perspective, mainly from the probability and the optimization view. We define the motion control by u_k (vehicle motion information using IMU, or Wheel odometry, etc.) and the measurement by z_k (provide a description of the environment using LiADR, Camera, etc.) and the state by x_k (vehicle position). So we can distinguish between two main forms of SLAM problems with the same importance.
The first one is the online SLAM which estimates only the pose at the time k and the map m. Mathematically, we search:

$$b(x_k, m) = p(x_k, m \mid z_{0:k}, u_{0:k}) \qquad \text{which is called "belief"}$$

The second type is the full SLAM, which approximates the entire trajectory $x_{0:k}$ and the map m, expressed by:

$$b(x_{0:k}, m) = p(x_{0:k}, m \mid z_{0:k}, u_{0:k})$$

The online SLAM is just an integration of all past poses from the full SLAM:

$$\begin{aligned} &p\left(x_t, m \mid z_{0:t}, u_{0:t}\right) \\ &\quad = \iint \cdots \int p\left(x_{1:t}, m \mid z_{0:t}, u_{0:t}\right) dx_0 dx_1 \ldots dx_{t-1} \end{aligned}$$

which is a beneficial formulation.
The idea behind the online SLAM is to use the Markov assumption (i.e., the current position depends only on the last acquired measurement). With this assumption, SLAM's methods are also based on the Bayes theorem so that we can write [86] [87]:

$$\text{posterior } = \frac{\text{likelihood } \cdot \text{ prior}}{\text{marginal_likelihood}}$$

Theh applying this theorem on the belief and affecting each term with its definition with some details. This step is called **Measurement_Update**.

$$p\left(x_k, m \mid z_{0:k}, u_{0:k}\right) = \frac{p\left(z_k \mid x_k, m\right) p\left(x_k, m \mid z_{0:k-1}, u_{0:k}\right)}{p\left(z_k \mid z_{0:k-1}, u_{0:k}\right)} \tag{1}$$

Where:

- **Likelihood** $= p\left(z_k \mid x_k, m\right)$ which gives the possibility of making an observation z_k when the vehicle position is x_k and the set of landmarks m are known. Called in the literature the **observation_model** [87].

- **Prior** $= p\left(x_k, m \mid z_{0:k-1}, u_{0:k}\right)$

$$\begin{aligned} &= \int p\left(x_k \mid x_{k-1}, u_k\right) \\ &\times p\left(x_{k-1}, m \mid z_{0:k-1}, u_{0:k-1}\right) dx_{k-1} \\ &= \int p\left(x_k \mid x_{k-1}, u_k\right) \cdot b(x_{k-1}, m) dx_{k-1} \end{aligned}$$

  This calculation of the prior called the **Prediction_Update** which represents the best possibility of the state x_k. The term $p\left(x_k \mid x_{k-1}, u_k\right)$ is called **motion_model** [14] [87] [16].

- **Marginal_likelihood** $= p\left(z_k \mid z_{0:k-1}, u_{0:k}\right)$ is a normalization term which depends only on the measurement and it can be calculated as given below [87] :

$$\begin{aligned} &p\left(z_k \mid z_{0:k-1}, u_{0:k}\right) \\ &= \int p\left(z_k \mid x_k, m\right) p\left(x_k, m \mid z_{0:k-1}, u_{0:k}\right) dx_k \end{aligned}$$

At this moment, let us replace what we got in the calculation of the prior in the equation (1) and explore that the **Marginal_likelihood** is only dependent on the z_k to deduce this formulation [14] [16]:

$$\begin{aligned} p\left(x_k, m \mid z_{0:k}, u_{0:k}\right) &= \frac{p\left(z_k \mid x_k, m\right) p\left(x_k, m \mid z_{0:k-1}, u_{0:k}\right)}{p\left(z_k \mid z_{0:k-1}, u_{0:k}\right)} \\ &\overset{\text{proportional}}{\propto} p\left(z_k \mid x_k, m\right) p\left(x_k, m \mid z_{0:k-1}, u_{0:k}\right) \\ &\overset{\text{prior}}{=} p\left(z_k \mid x_k, m\right) \int p\left(x_k \mid x_{k-1}, u_k\right) \cdot b(x_{k-1}, m) \end{aligned}$$

The problem of SLAM, as we have noticed above, can be solved by following the two steps: **Measurement_Update** and **Prediction_Update**. So any solution to this problem should be a great representation of the **observation_model** and the **motion_model**, which will give us the best estimation of the two steps aforementioned. The above explanation was about the probabilistic representation of the SLAM problem [16]. Now, le us change the focus to optimization and formalize the problem. Indeed, optimization is involved in resolving SLAM problems by minimizing a cost function which depends on the pose x_k and the map m with constraints. In the case of Graph SLAM, the cost function can be [14] [99]:

$$Fun(x) = \sum_{ij} e_{ij}(x_i, x_j)^T \mathbf{\Omega}_{ij} e_{ij}(x_i, x_j)$$

$$x^* = \underset{x}{\operatorname{argmin}}\ Fun(x)$$

where X is the set of vehicles poses, m is the set of landmarks that consist the map, $e_{ij}\left(x_i, x_j\right) = z_{ij} - z^*_{ij}\left(x_i, x_j\right)$ is the error function which calculate the distance between the prediction and real observation in the node i, j , Ω_{ij} is the information matrix.

2.4 Localization

Localization is a crucial task within the context of any autonomous system development. This task is indispensable such that to tell the vehicle where it is at each moment in time. Without this information, the vehicles can not avoid collision correctly and can not drive in the correct line. At this level, we have seen in (section2) how to extract relevant features from sensor measurements, and we have investigated different approaches to find the best representation of the environment (section3), which means that we have prepared the material to localize the vehicle. In the remainder of this section, we will depict what is new on localization approaches.

2.4.1 Dead Reckoning

Dead Reckoning (DR) is one of the oldest methods that determines the position of an object based-on three known pieces of information, including the courses that have been traveled, the distance covered (which can be calculated by using the speed of the object in the trajectory and the time spent within), lastly a reference point, which is to know the departure point and the estimated drift if there is. This method does not use any celestial

reference, and it is helpful in marine navigation, air navigation, or animal navigation [100], [101]. So the coordinates can be found mathematically by:

$$x_i = x_0 + \sum_{i=0}^{k=1} d_i \times \cos\theta_i$$
$$y_i = y_0 + \sum_{i=0}^{k=1} d_i \times \sin\theta_i$$

where (x_0, y_0) is an initial position and d_i, θ_i are, respectively, the shortest way and the angle between the current and last position.

2.4.2 Triangulation

Triangulation strives to estimate vehicles' position based on geometrical properties. The principal idea is that if we have two known points, we can deduce the coordinate of the third point, which is similar to GPS methods. For an accurate estimation of vehicles' positions, we should use at least three satellites. To solve this estimation, the procedure requires two steps as detailed below [100], [102].

- **Distance estimation**:

This is approximately the distance between the transmitter (satellites) and the receiver (vehicle). In the literature, many methods have been implemented to estimate this distance. One of them is based on the Time of Arrival (ToA), which is the time that takes a signal to arrive at the receiver, and it is based on the signal propagation speed. So, the distance can be estimated (a distance equal to speed * time). One of the problems here is that the time should be synchronized between the sender and the receiver, which is difficult. An extension of this method is the Time Difference of Arrival (TDoA), which calculates the difference of time between two signals with different propagation speeds, from which we can deduce the position. There are other methods to do that, including Received Signal Strength Indicator (RSSI), Hop-based, Signal attenuation, Interferometry [100], [102].

- **Position estimation**:

The position of the vehicles can be found based on the geometric properties and based on the distance estimated above. The calculation complexity depends on the dimensionality

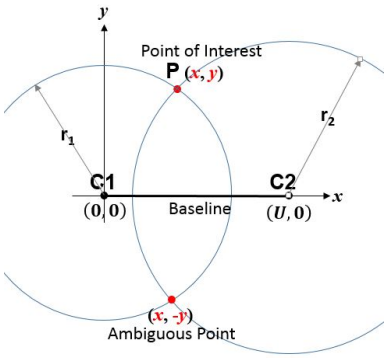


Figure 2.3: Cartesian 2D example scenario. C_1 and C_1 are centers of circles (satellites). P is point of interest (vehicle) with coordinates (x, y) and r_1, r_2 are the distances from the center of circle, respectively from C_1 and C_1, to the desired point [103]

sought (2D or 3D) and the type of coordinates (Cartesian or spherical). Let us take a small example in 2D Cartesian coordinate system; see figure.4. [103]
Based on Pythagoras's theorem:

$$r_1^2 = x^2 + y^2$$
$$r_2^2 = (U - x)^2 + y^2$$

Thus

$$x = \frac{r_1^2 - r_2^2 + U^2}{2U}$$
$$y = \pm\sqrt{r_1^2 - x^2}$$

We conclude that the remaining coordinates (x, y) can be found based on the distance to the satellites.

2.4.3 Motion sensors

Three principal components, namely acceleration, gyroscope, magnetometer, and motion sensor are used to detect object motions. Motions can be a translation over the axis x, y, or z, calculated by integrating the acceleration and computing the difference between the current and last positions. Motion can also be a rotation, which can be calculated by the gyroscope that provides the orientation using the roll, pitch, and yaw [100]. Much research has been proposed to use the motion sensors like IMU and wheel odometry to localize vehicles, but exploring machine learning algorithms in this field has shown its performance. Various architecture have been proposed. Input Delay Neural Network (IDNN) was used by Noureldin et al. [104] to learn error patterns in GPS outages. A

comparative study performed in [105], where they attempted to find the best guess of the INS during GPS outage, by testing different methods including Radial Basis Function Neural Network (RBFNN), propagation neural network, Higher-Order Neural Networks (HONN), Full Counter Propagation Neural network (Full CPN), back- propagation neural network, Adaptive Resonance Theory- Counter Propagation Neural network (ART-CPN), and the IDNN. Recent work by Dai et al. [106] used Recurrent Neural Networks (RNN) to learn INS drift error, which is more robust and performs well given that the RNN is intended to solve time series problems. Onyekpe et al. [107] suggested that a slight change in the diameter of the tire or the pressure can lead to a miss-displacement in the odometry. Also, LSTM-based methods have been performed to learn the uncertainties in the wheel speed measurement that appear while the vehicle displacement is more potent than the INS solutions. The same authors extended their work to obtain more accuracy and robustness against different GPS outages. The Wheel Odometry Neural Network (WhONet) framework justifies the use of the RNN network by examining various networks, including IDNN, GRU, RNN, and LSTM [108].

2.4.4 Matching Data

It is the task of matching data recorded from a sensor like signals, point cloud, and images with the real-word data reference to extract the autonomous driving location. Some preliminaries are presented below to enable further discussions in this subsection. The Rigid transform matrix is in this form:

$$\left(\begin{array}{ccc|c} & & & t_x \\ & \mathbf{R} & & t_y \\ & & & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

where R is the Rotation matrix, which is an orthogonal matrix with a determinant of ± 1 (*i.e.* $RR^T = R^T R = I_3$) and $T = (t_x, t_y, t_z)$ the translation vector. In the rest of this subsection we will present some approaches that are based on the matching process.

2.4.4.1 Fingerprint

This method enables the localization of a vehicle based on matching techniques. Indeed, the process is to create a database based on a reference car equipped with different sensors to record various environmental changes. After that, a" query" car will travel the environment and register its fingerprint, which will be compared with the database and pick the position that has the best match [100] [109].

2.4.4.2 Point cloud matching

3D point cloud registration provides a highly accurate source of information that enables it to be employed in various domains, including surface reconstruction, 3D object recognition, and, lastly, localization and mapping in self-driving cars, which is our purpose. To do so, scan registration can present sufficient information for vehicle localization by aligning multiple scans in a unique coordinate system and matching similar parts from the scans. In literature, several methods have been proposed to solve the scan matching. The Iterative Closest Point (ICP) is one of the earliest, proposed in 1992 by Besl and Mckay [110]. Its precision enables it to be very usable simultaneously in the positioning and mapping tasks. The idea behind this is to find a rotation matrix R and a translation vector T that will align the query scan with the reference scan iteratively. In fact, the aim is to minimize the distance between the points a_i from the target scan and their neighbor one b_i from the query scan [111]:

$$\arg\min_{R,T} \left\{ \frac{1}{M} \sum_{j=1}^{M} \|a_j - (Rb_j + t)\|_2 \right\}$$

Note that, in each iteration, we stock the new value of the query scan $b_i{}' = Rb_j + T$.
However, this method suffers from some serious limitations and assumptions that can affect the convergence of the algorithm, including the good initialization of the algorithm, e.g. one surface being a subset of another, high computational cost, etc. [112]. As consequence of these issues, many types of methods have been proposed to improve the ICP algorithm. The article [112] divides these extensions into five stages, depending mainly on the selection of the sampling and the feature metric used like point-to-point or point-to-plan methods, which effectively reduce the number of iterations. In recent years, the Normal Distribution Transformation (NDT) [113] emerged to solve the problem of measurement error and cover the problem of the no-correspondence relationship. Indeed,

the idea is to break down the set of points' cloud into a 3D Voxels representation and assign for each voxel a probability distribution. So, even if we have a millimeter of error far from our real measurements, the NDT algorithm takes place to match far points based on the probability distribution [114]. As a result, some researches have combined the NDT process with the standard ICP algorithm, like in [115]. Another interesting approach, the Robust Point Matching (RPM), has been performed to overcome the problem of the initialization (RPM) and change the traditional correspondence into a $0, 1$ matrix. Let m_i be the i_{th} point in M and s_j be the j_{th} point in S, so we define the correspondence matrix μ by:

$$\mu_{ij} = \begin{cases} 1 & \text{if point } m_i \text{ corresponds to point } s_j \\ 0 & \text{otherwise} \end{cases}$$

Thus, according to this method, the problem is to find the Affine transformation t and the match matrix μ that consists the best fit. The decomposition can give the transformation:

$$T(m) = \mathbf{A}m + \mathbf{T}$$

t is a translation vector, A is a matrix that depends on some parameter to be estimated based on a cost function [111] [116]. An extension of this method is the Thin plate spline robust point matching (TPS-RPM) algorithm, which augments the capability to handle the non-rigid registration [117].

Gaussian mixture models (GMM) family takes its place in this field as evidenced in literature. GMMReg [118] is one of them, where they try to represent the two points cloud into a GMM model to be robust to noise and outliers. Instead of aligning points, which is prone to error, they align distributions. e.g taking the intersection of these two distributions. Then, a standard minimization of the euclidean distance of the two new points cloud was performed to refine the transformation. DeepGMR [119] utilised the strategy of using deep learning methods to learn the correspondence between the GMM component and the transformation. Other methods based on deep learning can be found, like relativeNet [120], PointNetLK [121], Deep closest point (DCP) [122]. A complete survey on point cloud registration methods can be found in [123].

2.4.4.3 Image matching

Images contain plenty of information that can be explored to do many things, like object tracking, segmentation, object detection, and so forth. Due to that, the vision-based lo-

calization is supported by geospatial information. It is now possible to estimate the poses of vehicles by checking the movement of a point (pixel or features) from one image to another. In the literature, vision-based localization is divided into local methods (or indirect or feature-based) and global methods (or direct or appearance or featureless) [124]. Local ones consider just a part of the local region that characterizes the images in the database and the query image. The global method, or the direct method, uses the movement of the intensity of each pixel without using a pre-processing step, such as feature extraction.

Based on [125], image-based localization can be divided into three groups. Primarily, the video surveillance based-methods track the vehicles based on cameras mounted on road infrastructure, and they determine their position based on cameras calibration after detection. Secondly, they tend to search the similarity between the query image and a pre-built database. So the process is:

similarity research : After performing the feature extraction (section2), we receive a massive amount of data features that should be reduced to search quickly the similarity images. Typically, we need to profit from these features as much as possible without losing the generalization. Quantization is one solution for that. It is the process of creating an indexed which facilitate the research by query image. It adopt the concept of text research where they express each image by a vector of features, and a dictionary is built based on a large set of features which are extracted from visual documents, then applying a clustering method to reduce the size of the dictionary, each cluster centroid called visual word. Hence we can calculate the frequency of a specific visual word in the visual document which is called bag of feature (BOF) [126]. Another method called k nearest visual words was used in [127].

The similarity can be obtained by calculating the L2 norm between the identified features. However, this method did not work well in the case of many descriptors. which means we need to find a robust algorithm that will search similarity efficiently. Researches like [128] [129] turn this problem into an SVM classification task. [130] presents a new architecture to associate the similarity features called Multi-Task Learning (MTL). To improve the similarity process, we add another step of ranking, which is the task of underpinning the results of the retrieved data by classifying them into a list of candidates. For further information, the reader is referred to [8].

The last group is the visual odometry (VO) based methods, where the idea is to track the movement between two consecutive frames by matching the overlapping area. This

method is widely used in the robotics field.

Let M_1, M_2 two image matrix, respectively, of image 1 and 2. The problem is to find a vector $t = [u, v]$ where:

$$\begin{bmatrix} p_{m_1} \\ p_{n_1} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_{m_2} \\ p_{n_2} \end{bmatrix}$$

(p_{m_1}, p_{n_1}) and (p_{m_2}, p_{n_2}) points, respectively, from image 1 and 2. The position of the vehicle can be calculated by minimizing the photo-metric error. The advantage of this method is that it can be performed even in a low-texture environment. However, it suffers from a high computational cost.

In the literature, we found some cost functions that we can serve to find the best transformation vector t. Such as minimizing the sum of squared difference (SSD):

$$\mathrm{SSD}(t) = \sum_{p_m, p_n} \left(M_1(p_m, p_n) - M_2(p_m + u, p_n + v)\right)^2,$$

Or minimizing the sum of absolute differences (SAD):

$$\mathrm{SSD}(t) = \sum_{p_m, p_n} \left|M_1(p_m, p_n) - M_2(p_m + u, p_n + v)\right|,$$

Alternatively, maximizing the sum of cross-correlation coefficients.

$$CC(t) = -\sum_{p_m, p_n} M_1(p_m, p_n) M_2(p_m + u, p_n + v)$$

In the case of changing in the environment condition (i.e illumination), we must normalize our metric. Example with cross correlation.

$$A = M_1(p_m, p_n) - \bar{M_1}$$

$$B = M_2(p_m + u, p_n + v) - \bar{M_2}$$

$$NCC(t) = \frac{-\sum_{p_m, p_n} A * B}{\sqrt{\sum_{p_m, p_n} (A)^2 \sum_{p_m, p_n} (B)^2}}$$

where $\bar{M_1}$, $\bar{M_2}$ are the mean intensities in each image. More information can be found in [124]. Moreover, the survey [8] divides the set of direct methods into three approaches. First class is Direct vision-based localization (VBL) with prior, which are a group of methods that are built under the assumption of an existing prior (from GPS, magnetic compass). From the methods that hold this assumption, we found the article [131], where they use an initialization from the GPS and compass embedded in a smart-phone to refine

the global GPS coordinates, a coarse GPS initialization was given in [132] to perform a particle filter localization, also [133] used an indirect method as pre-processing step to refine estimation pose. The second class of methods is features (from images) to points (point cloud) matching, which find coherence between 2D image features and 3D points cloud, like in [134]. The third class is Pose regression approaches, which learn to regress the visual input data to the corresponding pose using mainly the regression forest method and CNN, like in [135], [136].

2.4.5 Optimization-Based approches

Optimization is the task of finding the best guest that can minimize or maximize a cost function concerning some constraints. These problems can exist in various domains, including operations research, economics, computer science, and engineering. In our case, we will use it to solve the localization and mapping problem, particularly the SLAM problem. Solving the SLAM problem is divided into front-end and back-end steps, from an optimization perspective. Front-end is the task of processing the data by extracting features, data association, checking loop closure, etc., in order to build a highly accurate map. The Back-end step is responsible for finding the best guest of the locations of the vehicles by minimizing (or maximizing) a cost function [138].

One question is raised here; what is the difference between the loop closure and the re-localization. **Loop closure** is performed to cover the problem of error of sensors' measurement, because a measurement taken from any sensor is not what it is in reality. Also, the vehicles are prone to having different issues like drifting, acceleration changes, and weather conditions, which can affect the reliability and credibility of these measurements. Loop closure presents an aid in detecting whether the vehicle can re-visit the same location. If this were the case, the loop will be closed. See the example in Figure 5.

Re-localization is a task performed if the system fails to detect its position in the map. Often, this problem appears when there is an inadequate matching process. Thus we 're-localize' through place recognition.

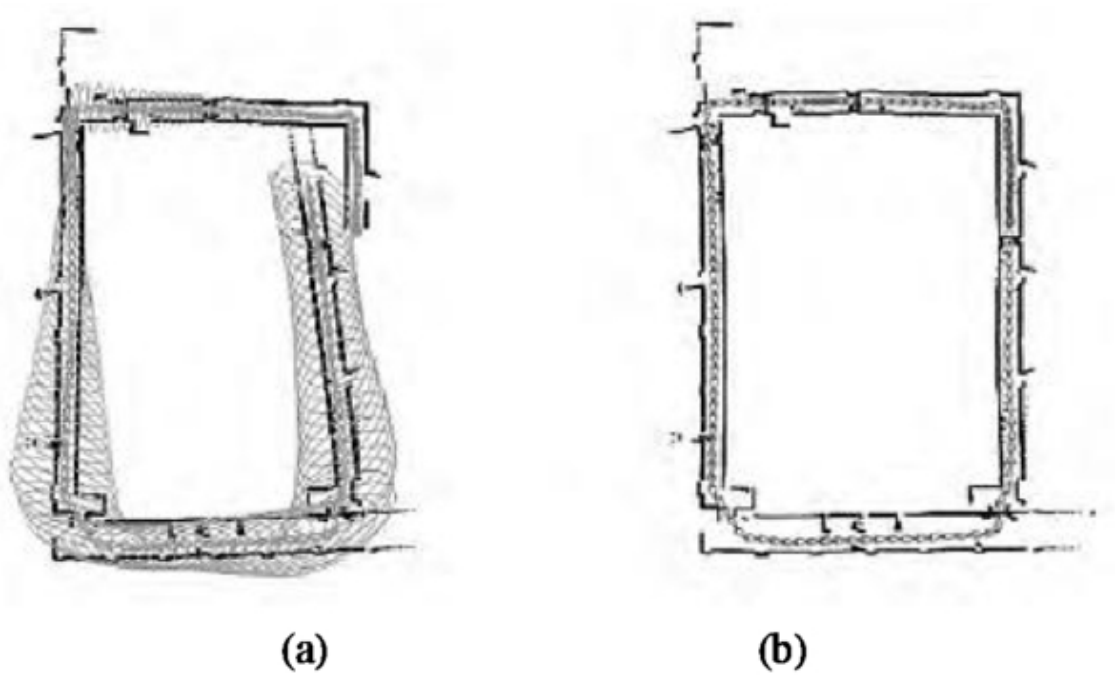


Figure 2.4: Example of a loop closure [137]. Image (a) shows a failure in loop closure. Image (b) shows a correct loop closure

2.4.5.1 Bundle adjustment based methods

One of the earliest methods that adopt the optimization techniques is the bundle adjustment. Bundle adjustment estimate simultaneously the camera pose parameter and the 3D point by minimizing a cost function. This method was used primarily in the 3D reconstruction. However, an adaptation was performed to apply this method to solve the SLAM problem. Due to its high performance, the Levenberg-Marquardt algorithm has been widely used to solve the optimization task [14] [139]. [139] divides the series of bundle adjustment methods into a group that tries to improve the efficiency of the bundle adjustment algorithms and a second group that aims to reduce the frequency of invocation of individual bundle adjustment. The term bundle adjustment refers to the light ray that gets out from the features and converges to the camera center 'bundle', which should be 'adjusted' according to the features and camera center. The core idea is the sum of errors between 2D observations and the 2D points extracted from the re-projection from the 3D object with the help of camera parameters. Mathematically, the bundle adjustment is a non-linear least square problem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \left(u_{ij} - \pi \left(C_j, X_i\right)\right)^2$$

u_{ij} is what we observe in the pixel at i_{th} and j_{th} node, $\pi(C_j, X_i)$ is the an operation that takes in input the 3D points $X = \{X_i\}$, and the camera parameter $C = \{C_i\}$ which can be divided into intrinsic and extrinsic parameters. The output is the Corresponding re-projected 2D coordinates. The process of this algorithm starts with the transformation from the world coordinates to the camera coordinates. Then, the 3D point coordinates are projected into a normalized image plane using triangular theory. After that, a distortion correction is applied. Finally, the remaining 2D point is deduced by projecting the corrected points into the pixel plan (intrinsic parameters) [139].
Attention-SLAM [140] is a newest method that uses a modified bundle adjustment (weight-BA) method in SLAM to reduce the trajectory error. Also, SalNavNet was proposed to extract relevant features to help, as much as possible, to keep the vehicle on the trajectory. The method was tested in a dataset collected by the same authors named Euroc [141]. The test shows that the proposed method outperforms the ORB-SLAM method [142]. BALM [143] is a BA-based framework for Lidar mapping. This method uses a new voxel architecture to adapt the use of BA in Lidar. After that, the new BA method was incorporated with LOAM [144] to refine the map. [145] attempted to imitate the human vision character by introducing a Salient Bundle Adjustment (SBA). DO-SLAM [146] was proposed by S. Lu et al. They tried to combine direct-based and features-based methods to create an accurate SLAM framework. Indeed, the direct method was used for camera tracking, and the features method was used to infer the keyframe poses. The motion-only bundle adjustment gave an initialization pose.

2.4.5.2 Graph based methods

Recently, the SLAM problem has been solved by a new and fresh intuitive approach, the graph-based approach. This method involves building a graph whose nodes are the vehicles poses, and a measurement acquired and its edges can be considered as constraints. These constraints are obtained by the collected measurement and the motion actions. Once the graph is created (front-end), solving a minimization problem that involves finding the optimal configuration of the poses is required (back-end). For the first time, this method has suffered from the high computational time, which made it inapplicable, but with the development in the field of linear algebra, this method takes its place in the state-of-art of SLAM problem solver [147] [148].
In regards to the mathematical approach, we are focusing on minimizing a specific cost

function which is explained in depth in subsection 3.2:

$$Fun(x) = \sum_{ij} e_{ij}(x_i, x_j)^T \mathbf{\Omega}_{ij} e_{ij}(x_i, x_j) \qquad (2)$$

$$x^* = \underset{x}{\operatorname{argmin}}\ Fun(x)$$

where

$$e_{ij}\left(x_i, x_j\right) = z_{ij} - z_{ij}^*\left(x_i, x_j\right)$$

Is the error distance between the real observation z_{ij} and the expected observation z_{ij}^*, for the reason of abbreviation we will note it as e_{ij}. We will depict a traditional nonlinear least-squares optimization solution for this problem. Lets begin with applying first-order Taylor expansion to approximate the error eij.

$$\begin{aligned} e_{ij}\left(\breve{x}_i + \Delta x_i, \breve{x}_j + \Delta x_j\right) &= e_{ij}(\breve{x} + \Delta x) \\ &\simeq e_{ij} + J_{ij}\Delta x \end{aligned}$$

Where $\breve{x}$ is a good initialization and J_{ij} is the jacobian of $e_{ij} = e_{ij}(\breve{x})$ computed on $\breve{x}$, Δx is an increment that should be estimated, because the optimal solution is equal to the sum of the initial pose and the increment.

$$x^* = \breve{x} + \Delta x$$

We deduce the formulation (at the node i, j) by replacing the new error estimated:

$$\begin{aligned} & Fun_{ij}(\breve{x} + \Delta x) \\ & = e_{ij}(\breve{x} + \Delta x)^T \Omega_{ij} e_{ij}(\breve{x} + \Delta x) \\ & \approx \left(e_{ij} + J_{ij}\Delta x\right)^T \Omega_{ij}\left(e_{ij} + J_{ij}\Delta x\right) \\ & = e_{ij}^T \Omega_{ij} e_{ij} + 2{e_{ij}}^T \Omega_{ij} J_{ij}\Delta x + \Delta x^T {J_{ij}}^T \Omega_{ij} J_{ij}\Delta x \\ & = c_{ij} + 2b_{ij}\Delta x + \Delta x H_{ij}\Delta x \end{aligned}$$

where

$$\begin{cases} c_{ij} = e_{ij}^T \Omega_{ij} e_{ij} \\ b_{ij} = e_{ij}^T \Omega_{ij} J_{ij} \\ H_{ij} = J_{ij}^T \Omega_{ij} J_{ij} \end{cases}$$

We generalize over all the nodes i, j. So we can rewrite the cost function:

$$\begin{aligned} Fun(\breve{x} + \Delta x) &= \sum_{ij} F_{ij}(\breve{x} + \Delta x) \\ &\simeq \sum_{ij} c_{ij} + 2b_{ij}\mathbf{\Delta} x + \mathbf{\Delta} x^T H_{ij}\Delta x \\ &= c + 2b^T\Delta x + \Delta x^T H \Delta x \qquad (3) \end{aligned}$$

where

$$\begin{cases} c = \sum_{ij} c_{ij} = \sum_{ij} Fun_{ij}(\breve{x}) = Fun(\breve{x}) \\ b = \sum_{ij} b_{ij} \\ H = \sum_{ij} H_{ij} \end{cases} \quad (4)$$

The result is a quadratic form with a linear shift. So the minimum (after setting the gradient of Fun to 0) is:

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$$

Which is a linear system that can be resolved with QR decomposition, Cholesky factorization, Levenberg Marquardt, etc. This theory is working with the assumption of an Euclidean space of x, which is not the case most of the time for the SLAM problem. To solve that, a common approach is to solve the problem in a local Euclidean space called manifold [147] [149].

Recently EyeSAM [150] introduced a novel graph-based approach based on the iSAM algorithm to help the surgeon under challenging eyes operations, especially in intraocular microsurgery. This paper aims to generate a map of vasculature points and their positions in the camera by exploring a series of inputs video frames. S. Jung , D. Choi [151] use an HG-SLAM: Hierarchical Graph-Based SLAM in a (UAV) Unmanned Aerial Vehicle with the aim of inspecting a Bridge. [152] used a sensor fusion method to generate nodes by using visual-inertial (VI) (for camera and IMU) and normal distribution transform (NDT) odometry (for Lidar), then, Generalized iterative closest point (G-ICP) matching is applied to define the constraint of the optimization problem. Finally, a Graph-based method (iSAM) was applied to refine the (UAV) trajectory. In [153] a robust ICP modification was proposed to solve the problem of the mismatched scanned image, then a Graph-based method was performed to deduce the robot path.

2.4.6 Probability-Based approches

In the probabilistic approach, solving the localization and mapping problem refers to finding a best estimation of:

$$p\left(x_k, m \mid z_{0:k}, u_{0:k}\right)$$

A probabilistic estimation was calculated based-on the Markov and Bayesian assumption in section 3.2:

$$\begin{aligned} & p\left(x_k, m \mid z_{0:k}, u_{0:k}\right) \\ & \overset{\text{proportional}}{\propto} p\left(z_k \mid x_k, m\right) \int p\left(x_k \mid x_{k-1}, u_k\right) \cdot b(x_{k-1}, m) \end{aligned}$$

where:

- $b(x_{k-1}, m)$: is the belief on x_{k-1}
- $p\left(z_k \mid x_k, m\right)$: is the observation model
- $p\left(x_k \mid x_{k-1}, u_k\right)$: is the motion model

Finding the best guess of the state x_k is by applying the **Prediction_Update** stage. Practically, it can be modeled mathematically by a non-linear or linear function (ϵ_{k-1} is an iid noise model) [87]:

$$x_k = f_k\left(x_{k-1}, u_k, \epsilon_{k-1}\right)$$

And the **Measurement_Update** stage, Which can be modeled by [87]:

$$z_k = h_k\left(x_k, u_k, n_k\right)$$

These two stages depend mainly in the good estimation of **motion_model** $p\left(x_k \mid x_{k-1}, u_k\right)$ and **observation_model** $p\left(z_k \mid x_k, m\right)$
motion_model gives an idea about how the state change over time given noise. **observation_model** which answer the question that if I have a state x_k and a control input u_k what will be the measurement z_k.

Note that these formulations are intended to solve the SLAM problem. However, we can solve the problem interchangeably, estimating the localization problem and deducing the map or vice versa. In the rest of this section, we will propose some methods to solve just the localization problem, whereas we can adapt these methods to solving the SLAM problem.

2.4.6.1 Parametric methods -Kalman filter family-

Kalman filter (KF) is a method that aims to predict the future state based on the previous state. Typically, KF estimates the model's parameters when it is delicate to estimate them

directly. KF attempts to reduce the error and noise that could affect the model's accuracy and provide fast and efficient data processing since it uses just the last previous state. Mathematically, The idea behind the KF is to find the $\hat{x}$ that will maximize the posterior $p\left(x_k, m \mid z_{0:k}, u_{0:k}\right)$ by using some analytical solution. In this case by a linear function where the Prediction_Update is [153][87]:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + \epsilon_{k-1} = f_k\left(x_{k-1}, u_k, \epsilon_{k-1}\right)$$

And the Measurement_Update:

$$z_k = H_k x_k + n_k = h_k\left(x_k, u_k, n_k\right)$$

So the prediction of the state, at the time t is [154]:

$$\hat{x}_k = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1}$$

$$P_k = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1}$$

After exploring the previous state by the prediction state and the co-variance matrix, a kalman gain was introduced to update these two last formulation [154]:

$$K_k = P_k\left(H_k P_k H_k^T + R_k\right)^{-1}$$

$$\hat{x}_k = \hat{x}_k + K_k\left(y_k - H_k\hat{x}_k\right)$$

$$P_k = \left(I - K_k H_k\right)P_k$$

where:

- F_k: is a matrix which represent the effect of the state x_{k-1} on the parameters at time t
- G_k: is a matrix that gives the effect pf the control input on the parameters
- H_k: is the observation matrix that the state parameters to the measurement domain
- P_k: is the co-variance matrix that describe the the correlation between parameters
- ϵ_k: is the motions noise
- n_k : is the measurement noise

KF assumes the linearity, and the noise should be Gaussian which is not the case in real-life systems. Consequently, many extensions inherited from the KF concept aim to fix these limitations. Beginning with the Extended Kalman Filter (EKF), which is, in fact, a linearization of a non-linear system by applying the Taylor expansion, then, directly KF is performed on the remaining system. However, (EKF) suffers from the high computational cost, especially the calculation of the jacobian matrix, which can affect the accuracy results. Using Unscented Kalman Filter (UKF). It is believed that an approximation of a probability distribution is more accessible than a heavy computation of the jacobian matrix. So the sampling strategies based on Sigma points was proposed. For each sigma sampling point, we apply a non-linear transformation (the Unscented Transform (UT)) which is more adopted and robust for the actual case. Robust Kalman Filters (RKF) also has been used for linear and non-linear systems. This method uses linear matrix inequality that provides an upper bound for the co-variance error matrix [153][154]. More extensions, Cubature Kalman Filter (CKF), ensemble Kalman filter (EnKF), adaptive Kalman filter (AKF), switching Kalman filter (SKF), fuzzy Kalman Filter, motor EKF (MEKF), and more can be found in [155]. Most of them are interested in reducing large number of variables, linearization, minimization of the sampling set.

2.4.6.2 Non-Parametric methods -Particle filter family-

Particle filter (PF) was Introduced in the 1990s to overtake the problem of non-linearity and the non-Gaussian systems. PF is a great solution for so many fields, including medical and economy. Lastly, networking [156], PFs have registered significant progress, and they register a large number of extensions that will be discussed later. This method comes with a robust idea that can tackle any system without any prior assumption of linearity or Gaussian system. Again, let us take a look at the mathematics of it. PF uses the sequential Monte carlo method in which to represent the posterior $p\left(x_k \mid z_{0:k}, u_{0:k}\right)$ by a set $\{w^i_{0:k}, x^i_{0:k}\}_{i=1}^{N_s}$ of N_s sample and weights:

$$p\left(x_{0:k} \mid z_{0:k}, u_{0:k}\right) \approx \sum_{i=1}^{N_S} w^i_k \delta\left(x_{0:k} - x^i_{0:k}\right)$$

Where $x^i_{0:k}$ is a set of possible realization of the trajectory, $w^i_{0:k}$ is a set of weights that give a degree of importance to the sample $x^i_{0:k}$ that verified:

$$\sum_{i=1}^{N_s} w^i_k = 1$$

The higher the weight, the more important is the sample. $\delta(a)$ is the Dirac function (0 everywhere except a).

This approximation of the integrals by a discrete method enables an approximate probability distributions without regarding if it is Gaussian or not. However, one question can be raised here which is how we can sample from an unknown distribution $p\left(x_{0:k} \mid z_{0:k}, u_{0:k}\right)$?. So the idea is to use importance sampling, which is a method that generates particles from some usual distribution to deduce the distribution in question. Projecting this idea in our case and letting $q\left(x_k \mid z_{0:k}, u_{0:k}\right)$ a known distribution in which we can draw samples and verified (for the sake of abbreviation we will give formulates without motion control measurement $u_{0:k}$)

$$q\left(x_k \mid z_{0:k}\right) = q\left(x_k \mid x_{k-1}^i, z_k\right) q\left(x_{k-1} \mid z_{0:k-1}\right) \tag{5}$$

So the new approximation is:

$$p\left(x_k \mid z_{0:k}\right) \approx \sum_{i=1}^{N_s} \tilde{w}_k^i \delta\left(x_k - \tilde{x}_k^i\right)$$

where;

$$\tilde{x}_k^i \sim q\left(x_k \mid z_{1:k}\right) (i = 1, 2, \ldots, N_s)$$

And

$$\tilde{w}_k^i \propto \frac{p\left(x_k^i \mid z_{1:k}\right)}{q\left(x_k^i \mid z_{1:k}\right)}$$

Practically, $\tilde{w}_k^i$ formulation should be a recursive one, i.e depending on its last step $\tilde{w}_{k-1}^i$ which is more suitable for calculation. (applying Bayesian theorem in the nominator and formulate (5) in the denominator) gives:

$$\begin{aligned} \tilde{w}_k^i &\propto \frac{p\left(x_k^i \mid z_{1:k}\right)}{q\left(x_k^i \mid z_{1:k}\right)} \\ &\propto \frac{p\left(z_k \mid x_k^i\right) p\left(x_k^i \mid z_{1:k-1}\right)}{q\left(x_k^i \mid x_{k-1}^i, z_k\right) q\left(x_{k-1}^i \mid z_{1:k-1}\right)} \\ &\propto \frac{p\left(z_k \mid x_k^i\right) p\left(x_k^i \mid z_{1:k-1}\right)}{q\left(x_k^i \mid x_{k-1}^i, z_k\right)} \end{aligned} \tag{6}$$

On the other hand (demonstration is in [156])

$$p\left(x_k \mid z_{1:k-1}\right) = \sum_{i=1}^{N_s} \tilde{w}_{k-1}^i p\left(x_k \mid \tilde{x}_{k-1}^i\right)$$

So

$$p\left(x_k^i \mid z_{1:k-1}\right) = \tilde{w}_{k-1}^i p\left(x_k^i \mid \tilde{x}_{k-1}^i\right)$$

By now we replace in (6) we got

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i \frac{p\left(z_k \mid x_k^i\right) p\left(x_k^i \mid \tilde{x}_{k-1}^i\right)}{q\left(x_k^i \mid x_{k-1}^i, z_k\right)}$$

Common choose of the importance density in the robot localization in general is [87] $p(x_k^i \mid x_{k-1}^i)$ so

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i p\left(z_k \mid x_k^i\right)$$

According to [87] PF is fronted by some serious challenge that could propose troubles in the convergence including:

- Degeneracy Problem: which appears in the case of a one weight particle being closed to 1 whereas, the other are close to be 0 due to the increase of the unconditional variance of the importance weights. This degenerecy can be measured by the effective sample size (ESS)

$$\mathrm{ESS} = \frac{N_s}{1 + \mathrm{var}\left(\tilde{w}_k^i\right)}$$

A big variance means a small ESS. Thus the degeneracy.

Hints: Re-sampling after the measurement update

- Sample Impoverishment: inherent from the degeneracy Problem, is where the particles collapse into a single point due to the low variance provided in the prediction update

Hints: Generate a sufficient amount of noise in the prediction update.

- Divergence: Caused by various things, including the inconsistent measurement, failures in the hardware, etc.

Hints: Prevent these causes by monitoring and update.

- Selecting the Importance Density: because it is the core of particles filter. Generating particles from a non-efficient Importance Density means that particles are out of range and will not converge to the state.

Hints: Choosing a great one.

- Execution: a large number of particles directly affecting the computational cost, a small number of particles affecting the convergence of the method.

Hints: check an optimal amount.

Due to these challenges, PF has been widely extended and developed to cover the discussed issues. An interesting one is the Unscented Particle Filter (UPF). An unscented transform is taken here to approximate the transformation of Gaussian variables [156]. Auxiliary Particle Filters (APF) is another extension that aims to highlight the particles that have a high likelihood by reusing the last measurement in the prediction step [87].

Extended Kalman Particle Filter (EKPF) is a mix of KF and PF to construct a robust importance density that can boost the experiments' performance. Rao–Blackwellized Particle Filter (RBPF) is a very important improvement where they divide the state $x_{0:k}$ into two chunk $x^1_{0:k}$ and $x^2_{0:k}$. Thus, the posterior $p\left(x_{1:k} \mid z_{1:k}\right) = p\left(x^1_{1:k'}, x^2_{1:k} \mid z_{1:k}\right)$ can be written as follow

$$p\left(x^1_{1:k'} x^2_{1:k} \mid z_{1:k}\right) = p\left(x^1_{1:k} \mid z_{1:k}\right) \cdot p\left(x^2_{1:k} \mid x^1_{1:k}, z_{1:k}\right)$$

The core advantage of this method is the reduction in time consumption. Another method is by reducing the dimensionality by the use of Adaptive Particle Filter (AdPF), which is based on varying the number of particles filter [87].

2.4.6.3 Localization with filter-based methods

PF and KF are widely used in the literature for localization vehicles like in [22] who used a KF to localize the vehicles based on the road marking features and Lidar reflectivity. [25] used double-layer KF to refine the location after extracting curbs features. [26] utilized a particle-based localization framework. [28] made use of a PF layer and a histogram filter layer based on the vertical features and ground features. In [29],[31] are also PF-based localization with some modification. SLAM problem, in the meanwhile, was very treated with these filters like in [157] an Invariant Kalman filter was proposed to solve the SLAM problem which is based on an Unscented KF and the theory of lie group (interesting). [158] is an improvement of the UKF-SLAM where a maximization method was designed to find the best parameters-collection for the model. A Rao–Blackwellized particle filter was used in [159] to solve the Monocular SLAM problem with a special weighting method. Q. Zhang, Y. li [160] attempted to combine a probabilistic architecture with graph theory. Indeed, the authors proposed a Bathymetric Particle Filter mixed with graph-based theory.

2.4.7 Cooperative localization

A more recent, and accurate approach is cooperative localization. According to [161], three key elements should be provided for successful autonomous vehicles, first, fusing information collected by the sensors. Second, integrating a high precision map. Third, providing technology to ensure communication which is the subject of interest of this section. The core idea is to create a communication between the vehicles with something

(lets call it X) in order to provide a fluid exchange of information, mainly their positions, speed, etc. This communication can be created using some devices, including WiFi, cellular, and UWB radio communications [18]. However, what does 'X', means, and what are the exciting communications that exist? [162], see Figure.6;

- **Vehicle-to-Vehicle (V2V)**: generate such a network between the vehicles where the information can be transmitted, this information (position, speed, orientation, etc.) can be used interchangeably between vehicles to perform the principles task like avoiding collision, changing lines.
- **Vehicle-to-Infrastructure (V2I)**: create communication between vehicles based on an intermediates infrastructure that is suitable to share information, from the node to any connected vehicles
- **Vehicle-to-people (V2P)**: Most generally uses the interaction of vehicles and other road users like walking people, people using mobile devices or bicycles or wheelchairs, etc.
- **Vehicle-to-Cloud (V2C)**: This is a technology that enables the vehicles to connect to the cloud-based on a cloud-based connected platform and via broadband cellular connectivity.
- **Vehicle-to-X (V2X)**: Offers the possibility of connecting the vehicles with everything, which is a great idea because providing information from road users' early gives enough time to execute, in a comfortable way, vehicles' task and checking the prioritization so that can increase the safety

with regards to V2V mode of communication. V2V requires a wireless signal to ensure communication between vehicles. In fact, V2V works in a Vehicular Ad-Hoc Network (VANET) environment, which is a temporary network environment where vehicles communicate with each other [18] [161]. V2V has so many applications, including emergency brake lights, pre-crash sensing, blind intersection warning, Forward collision, Left turn assist, Stop sign assist, Roadside service finder, etc. [161]. The history of information (like the position, speed, travel direction, acceleration) consists of the core messages that should be transmitted between vehicles to deal with the randomness of life road events. Most of the time, performing this communication requires the IEEE 802.11p protocol,

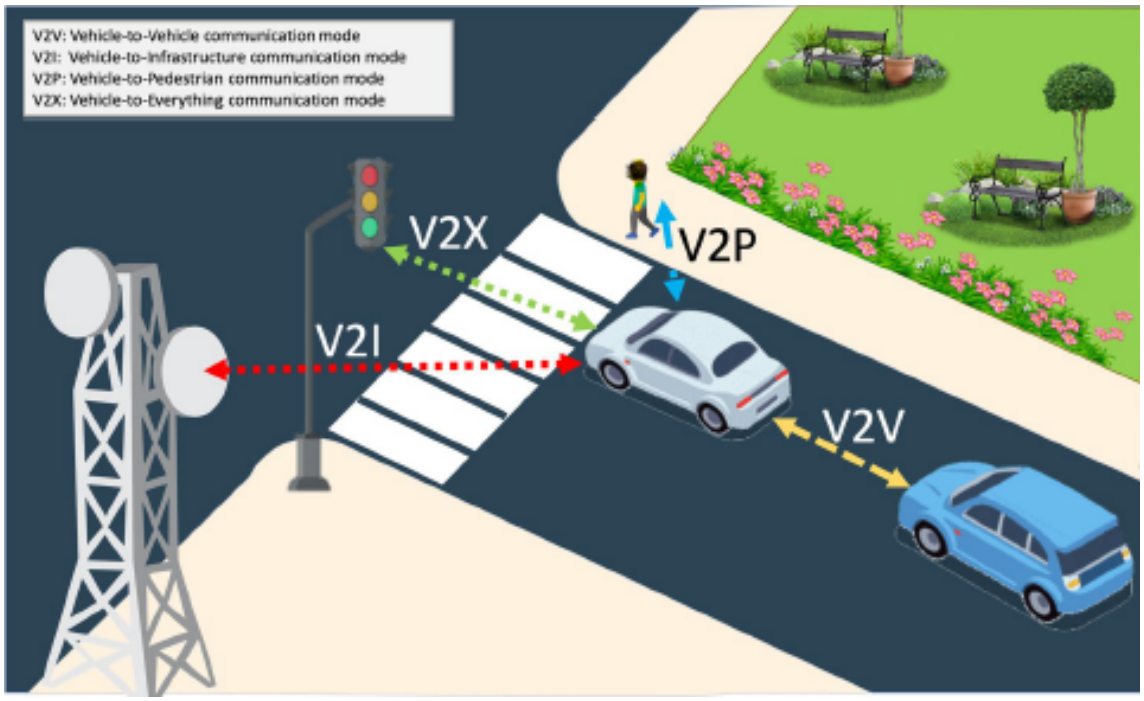


Figure 2.5: Illustration of the V2V-V2I-V2P-V2X [161]

which is an extension of the IEEE 802.11 protocol [18].

Probabilistic approaches were very used with the cooperative localization. [170] used features like inactive cars, traffic lights, or people who are linked with V2V communication to be fed as an input Kalman filter method based on GPS. Also, [171] proposed the Enhanced Using Enhanced Innovation-Based Adaptive Kalman Filter (EIAE-KF) architecture by fusing the innovation-based Adaptive Estimation Kalman Filter (IAEKF) with vehicular kinematic information. One of the most recent approaches [172] outperformed the GPSIDR system and Extended Kalman Filter (EKF), which uses information (i.e. location, speed, acceleration, etc.) to feed the Unscented Kalman Filter (UKF). A particle-based method was used in [173] based on the roadside unit (RSU), which achieved an error of 0.1 - 0.2m. A detailed survey was performed in [17].

2.5 Security in Localization

Integrating the revolution on telecommunication technologies in autonomous vehicles does bring about the problem of data security given the massive amount of data received from different sources (GPS, IMU, Cameras, Lidar, etc.) and transmitted (or received) via

different connection means (WiFi, Bluetooth, cellular connection, IEEE802.11p, WAVE). Attack threats, according to [163], could affect most of the sensors, and they can be performed remotely without using any attached component and without modifying parts of the AV. These attacks are not executed vainly, without a purpose or benefit. So, they should be interrupted from corrupting the autonomous system and stealing personal information, such as the location, destination, or other confidential information.

According to [163], most of the defense methods can be summarized into the Anomaly-based Intrusion Detection System (IDS) category, which can help detect unauthorized access data. Also, the abundance of information can be obtained by gathering data from different AVs, which can then be used to increase confidentiality. The final category is the Encryption method, which is intended to help sensors that lack authentication like Controller Area Network (CANs). In Table 5, we surveyed some interesting attacks that threaten different sensors of the AVs. Also, we present some key solutions to prevent these attacks.

2.6 Conclusions

In this survey, we have discussed and put forward the following terms: Features extraction, Mapping, and Localization processing in order to ensure safe localization and navigation.

Features represent the landmarks of the map or, more precisely, what we want from the vehicles to see in the environment (poles, curbs, buildings, intensity). The more relevant the features are, the more accurate the mapping, and therefore the more accurate vehicle localization will be. This survey presents a novel classification including the semantic, non-semantic, and deep learning approaches to facilitate the comprehension of these methods. Our investigation concluded that using non-semantic features is more suitable to solve localization and mapping tasks. We have added also that semantic features consume time to be extracted from the environment and can not exist in all the environment scenarios, and which can corrupt the matching process. Deep learning methods are excellent tools to represent the environment semantically or not depending on the approach used. Although the performance is high, the high computational consumption makes it a bit difficult to execute.

Moreover, the mapping stage, when a map is created based on the remaining features,

Table 2.3: A description of some of the attacks that threaten vehicle security systems and possible solutions.

Name	Devices	Access	Attack effect	Countermeasure
Malicious OBD devices	OBD	Physical	• Controlling some AV components • Injecting codes to ECUs	• Authenticity, Integrity • Reduce the time of execution [163]
CAN access through OBD	CAN	Physical	• Eavesdrop CAN messages from the ODB • Insert code in the ECUs	• Reinforce authenticity and confidentiality by proposing a robust methods of encryption [164] • Detecting attempts to transmit messages to the ECUs [165]
ECU access through CAN	ECU	Physical	• Change/add codes to the ECUs	• Same as the first two attacks
LiDAR spoofing	LiDAR	Remote	• ECUs make believe that there is an object close	• Signal filter, using machine learning [166]
LiDAR jamming	LiDAR	remote	• Denial-of-Service	• Changing wave frequency [167] • Integrating V2V communications[167] • Multiple LiDAR views [167]
Radar spoofing	Radar	Remote	• False distance estimation to the object	• [168] Propose (PyCAR) that detect malicious signals by transmitting a challenging signal that will be compared with certain threshold
Radar jamming	Radar	Remote	• Miss detection object	• Distinguish the correct signal from the counterfeit one [163]
GPS Spoofing	GPS	Remote	• False GPS location	• Controlling the GPS absolute, relative signal strength perform a time comparison • Use redundant information from multiple-GPS satellites [163]
Camera blinding	Camera	Remote	• Miss object detection	• Use multiple-cameras to increase the redundancy [169]
Adversarial Images	Images	Remote	• Provide incorrect prediction	• Increase the performance of • The pre-processing step after feeding to • The machine learning algorithms
Falsified information	V2V &V2I	Remote	• Threat the AVs operations and the flow traffics	• Reinforce the authentication schemes [163]

was examined as well. Different techniques have been investigated, including the pre-built and online ones, with the underlying mathematical intuitions. We found that the updates of maps are necessary to be done frequently to correct the changes in the environment. Moreover, choosing the right features to include in the map is questionable since we have to use a minimum of features for each scene without hurting the memory storage.

The last parameter that should be demystified is localization, where we have tried to collect and present, as much as possible, methods that aim to localize the vehicles according to different sensors. The approaches based on probability, optimization, or even cooperative localization are more robust and provide accurate results compared to the classical one. Each approach was described briefly and some recent related papers in the literature were investigated. Finally, we have reviewed some of the attack methods that can threaten the AV systems in general.

Part I

Non semantic features extraction method for localization and mapping purpose

CHAPTER 3

LOCALISATION AND MAPPING OF SELF-DRIVING VEHICLES BASED ON FUZZY K-MEANS CLUSTERING : A NON-SEMANTIC APPROACH

3.1 Introduction

Recent advancement in the development of Autonomous Vehicles (AVs) motivates the potential for enormous improvements in road safety. AVs offer a promotable solution to reduce the number of road related accidents and deaths. Critical to the achievement of this goal, is an effective AV localisation system for positioning the vehicle within its environment. The combination of position and orientation estimation techniques for localisation, and appropriate mapping techniques, are required for path planning for the safe navigation of the AVs.
A very common approach to the positioning of AVs is the use of a Global Navigation Satellite System (GNSS). The triangulation of at least three signals from the GNSS provides information to enable road localisation of AVs. The reliability of the GNSS however deteriorates due to visibility issues in tunnels, urban canyons, under bridges, dense tree canopies, etc.; multi-path reflections; and signal latency [174, 175, 176].

A common approach adopted for use during the absence of GNSS signals is the Inertial Navigation System (INS). The INS can be used to provide continuous position and orientation information based on the vehicles acceleration and orientation rate as provided by the accelerometer and gyroscope sensors respectively [176]. The accuracy of the

INS however deteriorates exponentially during the double integration of the acceleration measurement to position and the integration of the orientation rate measurement to orientation [177]. Similar to the accelerometer of the INS, wheel odometry has also been explored as a solution for position estimation in GNSS deprived environments [107, 108]. Using the measurements from a wheel encoder (which measures the speed of the vehicle's wheels), the linear velocity of the vehicle derived from the wheel's angular velocity can be integrated to determine the vehicle's position using one less integration step compared to the measurements from the accelerometer [108]. However, very much like the accelerometer, the wheel encoders measurements are plagued by errors which are amplified exponentially during its integration to position [108]. Several machine learning methods based on the Recurrent Neural Networks [176, 177, 107, 108, 178, 179, 106], have been proposed to learn the errors present within the wheel encoder, accelerometer, and gyroscope measurements to provide a more accurate navigation solution.

Camera and Light Detection and Ranging (LiDAR) are commonly used sensors for intelligent transportation, with the potential for lane accurate localisation. The technique called Registration [4], can be used to localise AVs by matching scans from LiDAR measurements or by matching keyframes from the camera. Typically, this approach tracks the motion between the keyframes in order to obtain the transformation matrix which gives the translation vector T and the rotation matrix R from the first keyframe to the second keyframe [180]. Consequently, the vehicle's position follow the transformation described in Equation 1.

$$X_t = R * X_{t-1} + T \tag{3.1}$$

Where X_t is the position of the vehicle at time t, and X_{t-1} is the position of the vehicle at time $t-1$.

Iterative Closest Point (ICP) is one of the methods that adopt this concept [188]. It matches lidar scans by minimizing the cost function:

$$\arg\min_{R,T} \left\{ \frac{1}{M} \sum_{j=1}^{M} \|a_j - (Rb_j + T)\|_2 \right\} \tag{3.2}$$

Where $\{a_j\}_j$, $\{b_j\}_j$ are the 3D coordinates points of two scans A and B, and M is the number of points inside the scan A (or B). The idea behind that is to find the translation and the rotation matrix that will accurately present the similarity between these two scans. Thus, the vehicles motion will also follow the same transformation as shown in Equation 1. [21]. Due to the measurement errors, ICP algorithm failed to achieve good accuracy in long-term matching especially for non-linear trajectories [**?**]. Conse-

quently, the authors of [113] have proposed Normal Distribution Transformation (NDT) to represent each group of 3D points formed on a voxel, by a probability distribution. The advantage of this representation is that it enables scans to be matched even with measurements errors or difficulties in the detection of similarity between them.

Nevertheless, the registration approach suffers from the high computational time and does not perform well in situations characterised by an insufficient amount of matching features or a dissimilarity between the scans [86].

In this research, we integrate another category of methods that combine inputs from various sensors to perform localisation. The Particle Filter (PF) is one of such approaches. The PF has several advantages supporting its use for localisation. These advantages includes the PF's ability to handle non-linear trajectories [86], track multiple objects that vary over time,[181] provide the capacity the handle occlusions and overlapping[181]. Moreover, it does not make prior assumptions such as the Gaussian distribution of the data in the Kalman filter approach [181]. PF was used in its early days for signal processing in order to estimate the states based on the observed variables [188]. After that, it was extended to solve some problems on dynamic systems including the localisation task, and it has achieved excellent results with a few centimeter level of errors [26], [27], [29], [31]. Practically, PF (also known as sequential Monte Carlo) makes the Markov assumption which is a concept that relates the state (or the position in our case) X_t with the previous state X_{t-1}. The algorithm starts by generating several thousands of particles that represent position candidates. Each particle is assigned a weight. A high weight value means more possibility that the particle is the vehicle's position.

PF offers the opportunity to explore the advantages of data from multiple sources. The motion sensors tell the particles the transformation (translation and rotation) from the last position to the current one. We then use the sensor's measurement to check whether the actual distances to the objects are similar to those we obtain from our measurements. Particles with small distances are more likely to be selected to represent the vehicle position and have high weight values. However, the huge amount of data we receive from the collectors complicates this process, forcing us to create a feature extraction layer to reduce the time and cost of this comparison. We propose for the first time the Fuzzy K-means clustering technique to extract relevant features and group similar data points,

which facilitate efficient search for similarities in the data associations.

The major contributions of this study are summarised as follows:

- Exploring the power of non-semantic characteristics to reduce map features using the Fuzzy k means algorithm. The remaining clusters are more representative especially, in the less texturised environment. Moreover, They are fast to extract and easy to use for data association tasks.
- A fast method to create a global map using the Gaussian Mixture Model (GMM) and simplifies the localisation task based on a modified particle filter.

3.2 Related work

Our survey of the state-of-art found three approaches that fuse information from different sensors: parametric filters like the Extended Kalman filter [181], non-parametric filters like a particle filter [182], and lastly least-square approaches like bundle adjustment or graph-based Simultaneous Localisation and Mapping (SLAM) [183], [184]. Usually, these methods explore measurements from the motion sensors to track the vehicle's movement i.e what is the translation and the rotation from frame $t-1$ to the frame t. They also check the data association in order to find how similar the scan at time t is to the global map. This information helps to perform place recognition and to localise the vehicle. However, the massive amount of data received from the sensors presents an issue computationally. Thus, employing a feature extraction method is tremendously important to reduce the dimensions of the data, computational time consumption, and computational energy cost; and for a clear representation of the map. There are two types of features: semantic and non-semantic features. Semantic features like poles, trees, buildings, sidewalks, etc., are widely used in the process of creating a lightweight feature map and localising the vehicle. The authors of [26] voxelised a set of points cloud provided by the LiDAR sensor and connected cells that contained minimum points. Each pole candidate has a certain number of connected cells (higher sense). Furthermore, a cylinder is fitted to the landmark to ensure reliability and to extract the pole's parameters. The localisation process was done using particle filter and achieved a good accuracy on the Kitti Dataset. Kummerle et al. [27] obtained an improvement in the localisation accuracy by detecting poles and walls. In [29], an algorithm was proposed to detect poles based on the intensity of the points

cloud in a voxel grid map. The algorithm was shown to detect landmarks like trees, street lamps, and telegraph poles, and identified their boundaries i.e beginning and ending of the pole's bodies. A probabilistic grid map was implemented in [31] to identify voxels with a higher value of the Beta probability distribution. The authors explored the idea of isolation (e.g poles are usually isolated from the surrounding) to propose a mathematical formulation that gives a score to pole candidates based on the intensity value inside and outside the pole. The closer the value is to 1, the higher the probability that it is a pole. Their approach also adopts a particle filter to perform the localisation task and achieved great accuracies on the Kitti [187] and the University of Michigan North Campus Long-Term vision (NCLT) dataset [190] compared to other methods they investigated. In the same regard, reference [22] used the probabilistic map to identify walls and buildings in its environment. A Kalman filter was then applied to localise the vehicle using the authors dataset. Generally, integrating these kinds of methods are computationally expensive. Moreover, these methods do not work in all environments i.e the environment should have a texture and contain the shape of the specified features. Moreso, these methods cannot be trusted [4].

An alternative to this approach is to use non-semantic features like extracting key points from image or LiDAR points. Recent works have adopted this approach like Oriented fast and Rotated Brief - SLAM (ORB-SLAM) family [183] [184] [142] have used the ORB features to extract camera's locations in each frame by matching features between key frames based on a local bundle adjustment, and created a map for the environment. A tracking stage was proposed to help the system in the process of relocalisation, place recognition, or even in matching frames. Those feature points are saved in a robust DataBase (DB) architecture called DBoW2 [185]. ORB-SLAMs frameworks have been tested in real-world data and they have achieved great accuracy in terms of tracking points, loop closure, and frame localisation. However, weather changes, lack of luminosity are major problems to the camera-based methods [86]. Hungar et al.[39] utilised the DAISY [191] descriptor to explain the 2D reflectivity map of the aligned points cloud. The remaining maps were fed into a gradient of intensities of eight radial orientations with Gaussian smoothing. This method compared the maps with twelve shapes i.e shapes are supposed to be relevant features. Some recent methods have exploited recent advancements in deep learning. Reference [23] proposed DeepICP as an approach for the novel generation of registration methods based on end-to-end learning. DeepICP overcomes the ICPs registration issue [186]. BirdNet as proposed in [63], uses transfer learning e.g VGG16 architecture to gather

important patterns from the Bird-eye's-view which is a 2D image established from the projection of the LiDAR measurements. They also applied the Fast-Region based Convolutional Neural Networks (Fast-RCNN) for object detection. This work was extended in [46] to propose BirdNet+. BirdNet+ has a direct end-to-end strategy applied on a 3D point cloud instead of using a projection pre-processing. Both methods were tested on the Kitti object detection data set benchmarks and achieved good performances. For further information about registration, please see [123]. The non-semantic approach is a good for creating a lightweight feature map, i.e., it consumes less time.

3.3 Methodology

3.3.1 System Architecture:

In this paper, a sensor fusion approach is adopted to exploit the motion sensor and LiDAR measurements, to enable an accurate representation of the physical environment. Due to its performance, a particle filter from the probability perspective was selected to perform the main localisation tasks and correct the measurement errors. Our system architecture consists of three principal stages: feature extraction, global mapping, and vehicle localisation as further illustrated in Figure. 1.

The massive amount of data points received from the LiDAR sensor was reduced by the feature extraction process. These extracted features have to satisfy some critical criteria such as being time-invariant i.e features should be stationary to make the data association between the local and global map not complex. Moreover, the algorithm must not spend too much time in the detection phase since the process of localisation is instantaneous. Finally, the method should be robust enough to detect features even in fewer texture environments.

3.3.2 Component explications:

In this section we explain in detail the workflow of our framework by describing the role of each components and the interactions between them.

- **Features extraction:** This stage contains several steps to be performed for each

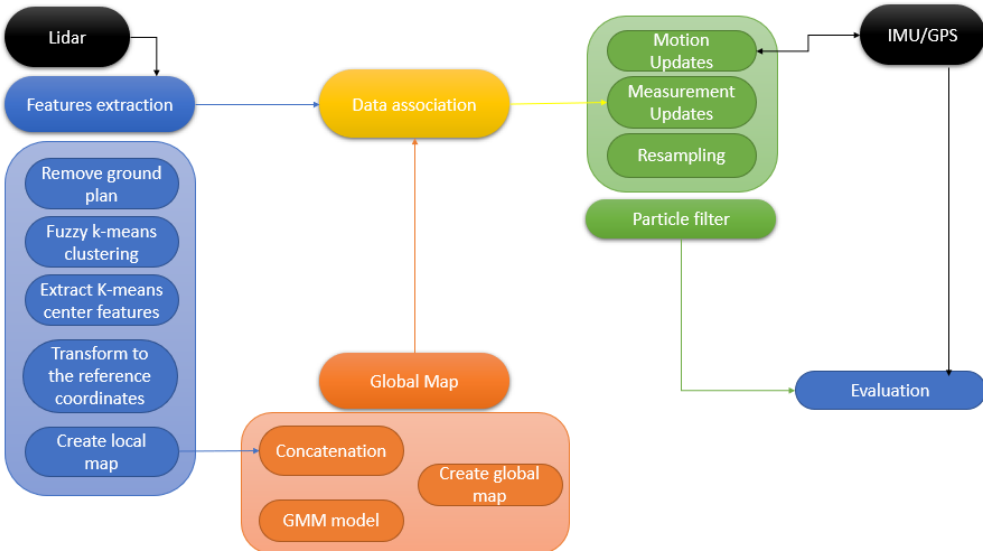


Figure 3.1: Showing the framework of the proposed approach, consisting of three main stages: the feature extraction (step 1), the global mapping (step 2), and the localisation stage (step 3)

scan received from the LiDAR. The first step involves removing the ground plan as it is somewhat uninformative and reduces the computational load. Furthermore, the Fuzzy k-means algorithm was applied to describe the environment. Fuzzy k-means was employed due to its speed in facilitating the extraction of the cluster centroid. A sample of the cluster center extracted by the Fuzzy k-mean is as presented in Figure. 2. Furthermore, the remaining points are transformed into a reference system to create the local map.

- **Create global map:** The global map was created by concatenating all the local feature maps. A Gaussian Mixture Model (GMM) was then applied to create several clusters for each group of points in the concatenated map. This idea is based on data association (detailed below) where each local feature map should be associated with a chunk from the global map. However, this association is prone to errors created by the detection of features from scan to scan as well as the concatenation. The aforementioned creates the need to employ a GMM model as a clustering solution for noise removal and to speed up the similarity search as shown in Figure. 3.

- **Data association:** is an indispensable task that indicates how far the scan at time t is from the real map. The more the similarity, the more the possibility that the vehicle is located there. The data association is divided into three categories: the short association which checks the motion between a few consecutive frames

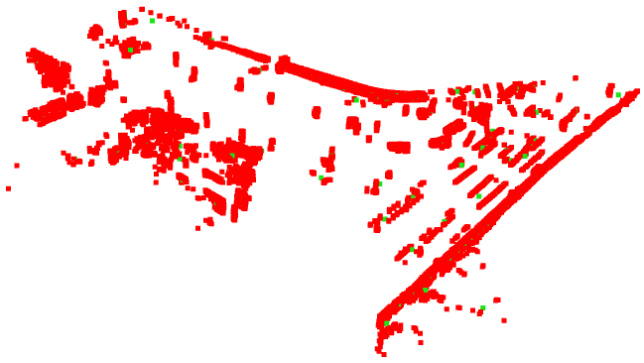
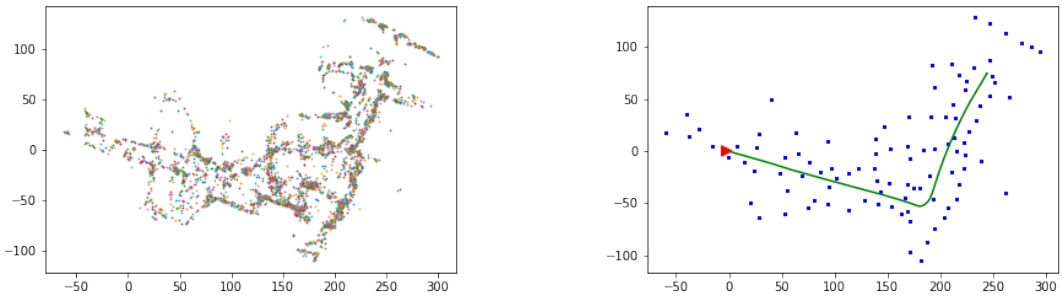


Figure 3.2: Showing a sample of cluster center extracted by Fuzzy k-mean, with the data presented in red, and the center of the cluster in green



(a) Concatenate map

(b) GMM-based map

Figure 3.3: Showing a sample of transformation of a concatenating map (above) into a GMM map (bottom). The number of neurons depends on the sequence.

[183]; secondly, mid-term association which identifies the relationship between the features point that are close to the camera and have a small accumulated drift; and finally, the long-term data association which is suitable for loop closure and place recognition since it provides long association threads.

- **Localisation with Particle filter:** In this stage, the position of the vehicle is updated based on the odometry information such as from the Inertial Measuring Unit (IMU), and the matched features between local and global maps received previously. A particle filter was chosen due to its performance to track multiple hypothesised poses and to handle data provided by the local and global map [86]. The idea is to generate a certain number of particles i.e examples of pose candidates. Each particle is updated with the IMU motion e.g translation and rotation matrix at every LiDAR scan t.

$$X^t = \{X^t_{Par_i}\}, \quad X^t_{Par_i} = R^t * X^{t-1}_{Par_i} + T^t \tag{3.3}$$

$X^t_{Par_i}$ is a 4 by 4 homogeneous matrix created after uniform generation of the x and y coordinates, and the *theta* orientation (roll) of the vehicle, $i \in [0, N]$, where N is the number of particles; and R^t and T^t are respectively, the rotation matrix and translation vector at scan t. Furthermore, an update was performed on each particle. Firstly, the data association is calculated between the local features map $\alpha^t = \{\alpha^t_k\}$, $k \in [0, M]$ and the global features map $\beta^t = \{\beta_k\}$, $k \in [0, M]$, where M is the number of feature points in the scan at time t, which is done through the KD-tree nearest neighbor algorithm. Secondly, the distance between each particle and the landmarks β which we call particle-landmarks-distance, is calculated. This provides information on how far the particles are from the real landmarks (suppose the global features map are the real landmarks). Finally, we evaluate the similarity of the particle-landmarks-distance[1] with the local features map which presents the vehicle measurements. The more the similarity, the more the probability that the particle coincides with the vehicle's location. This problem refers also to the weight update where each particle is assigned with a weight $w^t_{Par_i}$. The higher the weight, the more relevant the particle is. According to [31] these weights are updated by:

$$w^t_{Par_i} = \prod_{p \in [0,t]} p\left(\alpha^p \mid X^p_{Par_i}, \beta_{n(p)}\right) \tag{3.4}$$

Where

$$p\left(\alpha^p \mid X^p_{Par_i}, \beta_{n(p)}\right) := \mathcal{N}\left(\left\|X^p_{Par_i}\alpha^p - \beta_{n(p)}\right\|, \sigma\right) \tag{3.5}$$

σ is an isotropic position uncertainty depending on the reference landmarks. $\mathcal{N}$ is the normal distribution. $n(p)$ is an index of the global features associated with the k_{th} local feature (provided by KD-tree). Finally, a re-sampling method was performed in order to select particles with a high weight value to support faster computations and attainment of better accuracies especially if we have a small number of particles.

- **Evaluation:** In order to accurately evaluate our proposed approach, and compare it with other relevant published approaches, we have use different metrics that measure how accurate is the position is in comparison to the actual location of the vehicle (this includes 2D coordinates and the headings). Δ_{pos}, Δ_{lat}, Δ_{lon} and Δ_{ang} denote the mean absolute positional, latitudinal, longitudinal, and heading errors in metres, while $RMSE_{pos}, RMSE_{ang}$ represents the corresponding root mean squared errors

[1]It is the distance between the features map and the particles

of the position in metres and heading estimations in degrees.

$$\Delta_{lat} = \frac{\sum_{i=1}^{n} \left| X_i - X_i^{'} \right|}{n} \qquad \Delta_{lon} = \frac{\sum_{i=1}^{n} \left| Y_i - Y_i^{'} \right|}{n}$$

$$\Delta_{pos} = \frac{\sum_{i=1}^{n} \left|\left| pos_i - pos_i^{'} \right|\right|_2}{n} \qquad \Delta_{ang} = \frac{\sum_{i=1}^{n} \left| \theta_i - \theta_i^{'} \right|}{n}$$

$$RMSE_{pos} = \sqrt{\frac{\sum_{i=1}^{n} \left|\left| pos_i - pos_i^{'} \right|\right|_2^2}{n}}$$

$$RMSE_{ang} = \sqrt{\frac{\sum_{i=1}^{n} \left| \theta_i - \theta_i^{'} \right|^2}{n}}$$

Where n is the key frames number, $pos_i = [X_i, Y_i]$ is the predicted state, $pos_i^{'} = [X_i^{'}, Y_i^{'}]$ is the actual state, θ is the predicted angle, $\theta_{'}$ actual state

3.4 Results

3.4.1 Experiments

We performed the experiment on a computer with an i7 8th generation processor and 20 GB Ram. However, a GPU is recommended for better results in less computational time. We chose the Kitti dataset benchmarks to test our proposed method across different sequences characterising driving in cities, residential area, campuses, pedestrians, etc. The advantage of using these data is that it provides a wide range of sensor measurement information including, points cloud from the LiDAR, images from stereo or mono cameras, inertial measurements from the IMU, and a high-precision GNSS. Furthermore, Kitti enables the evaluation of the proposed method over various environmental changes [187]. We applied our feature extraction method with 10 core clusters for the k-means algorithm to create the global map (or ground truth) based on the high-precision navigation system. Two criteria were considered when choosing the number of clusters: speed and representativeness. A small cluster means high speed and less representativity and vise versa. We matched each local map with the global map to begin the localisation process using a particle filter. The number of particles differs from one sequence to another $[1000, 2000]$. Also, the range of initialisation is included in the interval $[-i, i]$ where $i \in [0, 1]$. The

Table 3.1: Localisation errors for various sequences categories.

Category	Seq	no of frames	t_{Feat} (s)	t_{Loc} (s)	Δ_{pos} (m)	Δ_{lat} (m)	Δ_{lon} (m)	Δ_{ang} (°)	$RMSE_{pos}$ (m)	$RMSE_{ang}$ (°)
City	0001	108	47s	38s	0.07	0.01	0.06	0.02	0.08	0.02
	0014	314	02:19s	57s	0.55	0.32	0.43	0.11	0.66	0.13
	0009	443	03:05s	01:21s	0.12	0.05	0.08	0.04	0.14	0.05
	0057	361	02:15s	01:05s	0.004	0.001	0.004	0.001	0.012	0.003
Residential	0020	86	41s	14s	0.017	0.002	0.016	0.004	0.019	0.004
	0035	131	55s	24s	0.024	0.007	0.021	0.036	0.029	0.064
Road	0052	78	34s	13s	0.01	0.0003	0.01	0.00033	0.01	0.0004
	0027	188	01:28s	33s	0.28	0.003	0.28	0.005	0.40	0.005
Campus	0034	49	22s	8s	0.002	0.001	0.001	0.005	0.002	0.005
	0035	35	19s	6s	0.0005	0.0002	0.0004	0.0009	0.0006	0.0007
Person	0053	68	32s	11s	0.0002	2.40e-5	0.0002	0.0005	0.0002	0.0007
	0054	45	22s	7s	0.0002	6.941e-5	0.0001	0.001	0.0002	0.001

t_{Feat} **:**	Is the cost of time to calculate the features extraction
t_{Loc} **:**	Is the cost of time to calculate the Localization

result of the localisation algorithm is a $4 * 4$ information (transformation matrix) containing the rotation and translation matrix done between two consecutive keyframes. The method was tested in all the categories provided by the kitti dataset which is in total 1902 scans from 4 series of sequences.

As the field of localisation and mapping research moves from observational studies to directly implementation in real-world scenarios, there is a growing need for methods that speed up the positioning process as we have proposed in this article. However, some leak points appear when using particle filter algorithm especially in the detection of the number of particles and the identification of threshold values.

Table 3.2: Performance comparison of the proposed technique. The results of Weng et al. also Kümmerle et al. are not straightforwardly similar and are expressed for qualitative analysis only.

Methods	Δ_{pos} (m)	$RMSE_{pos}$ (m)	Δ_{lat} (m)	σlat (m)	Δ_{lon} (m)	σ_{lon} (m)	Δ_{ang} (°)	σang (°)	$RMSE_{ang}(rads)$
Kümmerle et al.[27]	0.12	—	0.07	—	0.08	—	0.33	—	—
Weng et al.[29]	—	—	—	0.082	—	0.164	—	0.329	—
Sefati et al.[26]	—	0.24	—	—	—	—	—	—	0.68
A. Schaefer et al.[31]	**0.096**	**0.111**	0.061	0.075	**0.060**	0.067	0.133	0.188	0.214
Ours	0.12	0.141	**0.059**	0.09	0.08	0.05	**0.043**	0.078	**0.057**

3.4.2 Discussion

The results in Table 1 show that the proposed method is able to perform the localisation and mapping tasks rapidly and achieve excellent accuracy even in large frame sequences. Our findings on the usability of non-semantic features to represent the environments extends the work of Hungar et .al proposed approach called GRAIL [39]. Hungar et .al compared the characteristics of the GRAIL results with twelve forms [2] that are considered insufficient to cover all possible changes in the environment. GRAIL's method obtained an average absolute position error of $1.11 + / - 0.41(m)$, and $0.10 + / - 0.19$ (°) on the rotational error over 2433 scans, which is not provided by the authors to perform an exact comparison.

In our case, we took a 1902 scans, and we obtained an average absolute error of 1.079 (m) (calculated from Table 1)[3] and 0.22 (°) in the rotational errors. Fuzzy k-means speed up the data association between the local features and the corresponding features on the global map significantly. Our proposed method was also compared with that from Kümmerle et al. [27], Weng et al. [29], and Sefati et al. [26], and Schaefer[31]. According to the Table 2, our method outperforms the others on the heading error. However, Schaefer's method [31] outperforms our approach on the absolute positioning error because the data association is more accurate as shown in [31]. Nevertheless, conducting non-

[2]These forms represent the possible shape of the vehicle's trajectory.

[3]We calculated the average errors for all of the sequences

semantic localisation features is more advantageous as competitive accuracy is achieved in lesser time, is more representative, and works even in less texturised environments.

3.5 Conclusion

In this paper, we presented a localisation and mapping method for a self-driving vehicles which is based on extracting non-semantic features. Fuzzy K-means was used to find patterns inside the LiDAR scans. The remaining cluster centroids were used to create the local map features. Furthermore, the global map was created based on a GMM (Gaussian Mixture Model), which facilitates the data association. The localisation process was performed based using a particle filter. Our proposed approach achieved competitive results in terms of accuracy, but does so in significantly lesser time compared to state-of-art methods evaluated on the Kitti dataset benchmarks.

Part II

vehicle localization: A stochastic approach

CHAPTER 4

FAST AND ACCURATE LOCALIZATION AND MAPPING METHOD FOR SELF-DRIVING VEHICLES BASED ON A MODIFIED CLUSTERING PARTICLE FILTER

4.1 Introduction

With a progression of successful testing and developing speculation. Autonomous vehicles (AVs) are a step toward revolutionary technology that promotes road safety. Many scientists and specialists agree that autonomous vehicles are ready to move beyond the experimental phase [4]. Global organizations such as Tesla and Waymo have invested in autonomous vehicle research, with some preliminary large-scale projects around the world. AVs assist in controlling and managing the traffics by the Traffic Flow Prediction (TFP) which exerts a favourable impact on solving those issues [206]. AVs aim to reduce injuries and fatalities on the road by eliminating human error, but this is difficult to achieve because the AVs must know the area in which they are working [4].

AVs require robust localization and mapping to ensure the consistency and reliability of the tasks. A centimeter error of localization can perturb several AVs tasks including path planning, avoiding obstacles, lane following, and time of overtaking [207].

A Global Navigation Satellite System (GNSS) is used to locate the vehicle among three or more satellite signals. However, Global Positioning System (GPS) can have reliability problems due to visibility in tunnels, under bridges, in urban canyons and dense tree

canopies, multipath reflections and signal latency. As a result, on most roads, we will not be able to obtain sufficient information to locate the AVs vehicle. This also means that more sophisticated techniques are required to achieve sufficient localisation accuracy in difficult terrains [175],[176], [209].

The Inertial Navigation System (INS) is a key component of most guidance, navigation and control systems, particularly those which must operate in the absence of GNSS signals. In this context, an Inertial Measurement Unit (IMU) is the most important component of the system. The IMU consists of accelerometers and heading gyros whose outputs are integrated by a suitable algorithm to provide information on the position and orientation of the vehicle. However, this system suffers from cumulative errors caused by the double integration of accelerometers and heading gyroscopes. Like INS, wheel-odometry has been a reliable method for measuring the speed of production vehicles for many years due to its easy integration, high reliability and low cost. However, wheel-odometry is prone to errors, especially when the sensors are biased, i.e. when the average is not zero (causing a drift effect) or when the variance varies over time (causing a diffusion effect). [18], [7]. Several machine learning algorithms were proposed to correct the measurement information provided by the sensors with the GPS measurements [108].

Light Detection and Ranging (Lidar) and camera sensors can separately find the transformation (translation and rotation) that took place in each segment of the trajectory based on the so-called registration. Registration involves two main steps, namely feature extraction and feature matching. Features are typically detected from laser scans as 3D points, or as 2D points from images. Feature matching then determines the similarities between the extracted features using similarity measures (e.g. Euclidean distances for points, chi-square test for histograms). After determining the transformation, the vehicle position can be deduced easily by this formulation Equation 1.

$$X_t = R * X_{t-1} + T \tag{4.1}$$

X_t is the position of the vehicle at time t, T is the translation vector, and R is the rotation matrix.

Iterative Closest Point (ICP) registration is one of the most popular and successful approaches to local pose estimation [21]. It is the process of registering two point clouds by finding the transformations that best align them. The ICP algorithm proposes an optimization that determines the degree of alignment of the last/current scan and calculates

the rigid registration that minimizes the sum of squared Euclidean distances between two sets of corresponding 3D points; see Equation 2.

$$\arg\min_{R,T} \left\{ \frac{1}{M} \sum_{j=1}^{M} \|a_j - (Rb_j + T)\|_2 \right\} \tag{4.2}$$

$\{a_j\}_j$, $\{b_j\}_j$ are the 3D coordinates points of two scans A and B.

Despite its advantages, the ICP is sensitive to outliers and only works well when the two point clouds overlap [196]. It can be slow on large 3D scans because the transformation has to be calculated by iterating over each pair of points [4]. Due to measurement errors, the ICP algorithm also fails to achieve good long-term matching accuracy, especially for non-linear trajectories [196]. To solve this problem, the authors of [113] proposed the Normal Distribution Transform (NDT), which encodes each voxel with a probability distribution instead of a single value. The proposed approach addresses two problems in the traditional 3D registration framework. First, it discretised the set of points cloud into voxels using the NDT feature representation. Second, the authors match the statistical (i.e mean and variance) properties between two distributions (voxels) as a matching feature and find the correspondence based on the standard distance metrics of probability theory.

The surveyed methods either cost a lot of time, or they lack reliability, robustness, or accuracy. In contrast to all these methods that are based on a single sensor. In this work, we are mainly interested in sensors fusion, i.e., examining more than one source of information to localize the vehicle. The particle filter (PF) [182] is one of the methods that offer the possibility of processing multiple sources, i.e., measurement sensors such as cameras, radar or lidar, and motion sensors such as IMU or wheel odometry. Measurement sensors provide a description image of the environment, including the shape of the object and the distance between the reference vehicle and the objects. Motion sensors provide the translation and rotation performed between a time segment. Therefore, the particle filter, and more generally the Bayesian filter, offers the advantage of using the translation and rotation provided by the IMU and correcting the error by examining the difference between the real distance of the vehicle to objects and the one we have obtained from the measurement sensors. A particle filter is a form of Bayesian filtering. It has found applications in areas such as robotics, computer vision, and signal processing [86]. The particle filter extends the Kalman filter by including additional states for particles that

are not associated with a particular measurement, which provides the flexibility to work with nonlinear trajectories and also to work without assumptions about the probability distribution, such as Gaussianity in the Kalman filter family [86]. The PF algorithm iteratively performs two steps: updating the motion and updating the measurements. After uniformly generating a number of particles, the motion update calculates the translation and rotation found from the motion sensors for all the particles. Next, the distance between the global map features and the particles is calculated in order to be compared with the information from the local features. The higher the similarity, the more likely the position state. The position is calculated by taking the average of the closest particles or by directly taking their maximum.

In this paper, we propose a modified clustering particle filter, an extension that speeds up the position computation process by making the matching process fast in the update measurement phase, i.e., instead of matching all particles with the feature map, we use only a few prototype particles that can be efficiently distinguished. We addressed the particle selection problem with two approaches: clustering-based and sigma point-based methods, each of which is examined in detail. To facilitate understanding of the workflow of our method, we have exposed an architecture consisting of mapping and localization. The mapping phase is responsible for extracting features from the Lidar measurements (especially fuzzy c-means clustering and noise filtering with DBSCAN) and creating a global map of the environment. The localization phase is responsible for the input of the IMU and the results of the mapping phase to determine the corrected position.

The remainder of this paper is organized as follows: In the second part, we present related work on localization methods. In the third section, we provide a clear description of the architecture of our framework and the methodologies used to solve the localization task. The fourth section analyzes the performance of the method on two different datasets: Kitti [187] and Pandaset [195]. The last section is a conclusion.

4.2 Related work

In our state of the art, we have found different methods for locating AVs, mainly divided into three approaches: Registration (scan matching), optimization, and probability-based. Registration methods are used to find the best transformation (translation, rotation) between the last and current position of the vehicle. The most interesting work that adopted

this approach was the LiDAR odometry and mapping (LOAM) [144] which used scan-to-scan and scan-to-map matching with edges and flat plane feature points. However, scan-to-scan matching consumes more time and energy. LOAM_Velodyne [201] is a modified version of the LOAM [144] characterized by real-time speed. Another extension of the LOAM [144] method was proposed to reduce the cost of time in scan-to-scan matching. Lightweight and ground-optimized LOAM (LeGO-LOAM) [202] improved the features extraction process by removing noisy points and optimizing ground and non-ground points. A downsampling technique was proposed in A-LOAM [203] in order to reduce the complexity of the scan-to-scan matching. However, neither LeGO-LOAM nor A-LOAM arrived to reduce efficiently the high computational time. Deep learning-based methods take place in LO-Net [204] to reduce the error of the matching process by using the end-to-learning. An extension of LO-Net [204] was given in the same article [204] named LO-Net-M which is a LO-Net underpinned by a mapping stage which gives better results compared to LO-Net. SGLO [205] have registered good results in term of accuracy and fastness using only sparse geometric line and plane features with line-to-line and plane-to-plane data associations in the scan-to-map matching. However, the accuracy of the method depends mainly on the initialization. Recently, there have been many improvements in the scan matching approach. One way to overcome the problems of ICP is to use end-to-end learning as in DeepICP [186], which is an interesting improvement that provides a good registration rate. SDRSAC [113] and DS++ [192] have proposed a method to transform the optimization problem (Equation 2.) into a semidefinite optimization problem by introducing a symmetric matrix A describing the matching potential between the points of point pair X and $Y = |X|^2$. The methods transform the classical optimization problem into $max_{XY}AY$ with respect to some constraints that depend mainly on the normalization of the data. This technique offers the possibility to estimate the NP-hard global solution in (Equation 2.). The transformation can be derived according to the correspondence in [193]. Gaussian mixture models (GMM) also have their place here to solve the log problem in (Equation 2.) by modeling the input data as a maximum likelihood. The advantages of the GMM-based method are its robustness to noise and outliers [123]. The main idea in it is to develop an optimization strategy to optimize the transformation matrix by maximizing the likelihood [119]. DeepGMR [143] is one of the methods that applied this concept by teaching a deep learning model the similarity between GMM components and points. However, this technique suffers from a number of problems including noisy data and outliers captured from 3D lidar measurement, also some problems may appear due to unbalanced density, partial overlap or scale variation

of points between scans that deeply affect the matching process [119].

The second approach to address the localization and mapping problem are least squares (or optimization) techniques that aim to minimize (maximize) a cost function while respecting certain constraints. In this area, bundle adjustment is one of the first methods that allows to simultaneously find the camera position parameter (intrinsic or extrinsic) $\{C_j\}$ and the corresponding 3D points $\{X_i\}$. The method solves this problem :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \left(u_{ij} - \pi \left(C_j, X_i\right)\right)^2 \tag{4.3}$$

u_{ij} is the observation of pixels at node i_{th} and j_{th}, $\pi \left(C_j, X_i\right)$ is the projection operator that projects the 3D coordinate points into 2D coordinates. In general, bundle adjustment (BA) searches for attachments to find the best parameters (C_j, X_i) that yield the 2D point (pixel) closest to the actual pixel u_{ij}. The position can be derived from the extrinsic matrix. BALM [143] from the recent methods that used bundle adjustment adapted for the 3D Lidar measurement. The results were concatenated with the LOAM [144] method to deduce the map of the environment. Moreover, BA was used to solve the SLAM problem like in the ORB-SLAM family [7], [184], [142] which used ORB features to locate cameras and create a map of the environment. BA method act as a helper technique for locale matching. A tracking phase was proposed to aid the system in the process of relocation, site recognition, or even frame matching. frame matching. These feature points are stored in a robust database (DB) architecture called DBoW2 [185]. ORB-SLAM frameworks have been tested on real-world data and have achieved high accuracy in terms of tracking points, loop closure, and frame localization. However, changes in weather and lack of brightness are the main problems for camera-based methods.

The third option for locating AVs in environments is with the probabilistic perspective. The authors [26] of have used particle filtering to examine the translation and rotation provided by the IMU and find higher candidate poles by looking at the difference in between the features extraction. The authors voxelized the fit point cloud and vertically connected the cells that exceeded a fixed threshold. In addition, a cylinder was applied to all poles to eliminate noise and avoid object overlap. The method achieved good accuracy on the Kitti[187] data set. Kummerle et al. [27] detected poles and walls according to their characteristics and distinctiveness in the environment. Localization was performed using a particle filter, and they obtained a more interesting result. The particle filter was also used in [29] with an interesting algorithm for feature extraction. The authors have investigated the isolation and distinction of pole landmarks in the detection process

from the point cloud. The method obtained good results on the Kitti dataset [**?**]. [31] is an interesting work that used probabilistic map features to speed up the process of location determination and meanwhile construct a map of the environment. the location determination was performed based on a modified particle filter. The modification is a small perturbation (multiplication with a generated noise) applied to the predicted state which gives a higher generalization. The method has been tested on the Kitti [**?**] and NCLT dataset and has shown excellent accuracy, especially on the long-term trajectory. A real-time Monte Carlo localization (RT_MCL) was proposed in [207] which a trade-off between handling pose estimation accuracy and real-time performance was addressed. RT_MCL extracted features as landmarks after fusing Lidar and radar measurements using the unscented Kalman filter (UKF) and used a tailored particle filter to derive the vehicle pose. Data association between the global map (generated offline from the 3D lidar) and the local feature map was performed using ICP matching. The method showed good results in the CARLA driving simulator [208]. However, the method may have some distortion in real scenarios, such as an environment without poles and landmarks. In this paper, we extend the work done in [43] on non-semantic feature map reduction using k-means, global map generation using GMM and localization using particle filters by adding a DBSCAN filter after performing fuzzy c-means feature extraction to remove noisy points and striking lines from the environment. Moreover, we present the modified clustering particle filter which consists of choosing some relevant particles for position computation. We propose two methods to choose those particles and we analyze their potential using several tests in real scenarios.

4.3 Methodology

4.3.1 System Architecture:

This section describes the architecture followed to solve the problem of localization and mapping in self-driving vehicles by exploring and manipulating Lidar and IMU measurements. The overall system architecture is illustrated in Figure.1, which includes two main components: mapping and localization. The mapping phase is responsible for creating a global map of the environment by processing the Lidar scans (or scenes, or measurements). It consists of two main steps: feature extraction and global mapping. Feature extraction is the process of manipulating the Lidar scans to extract relevant features, remove noise,

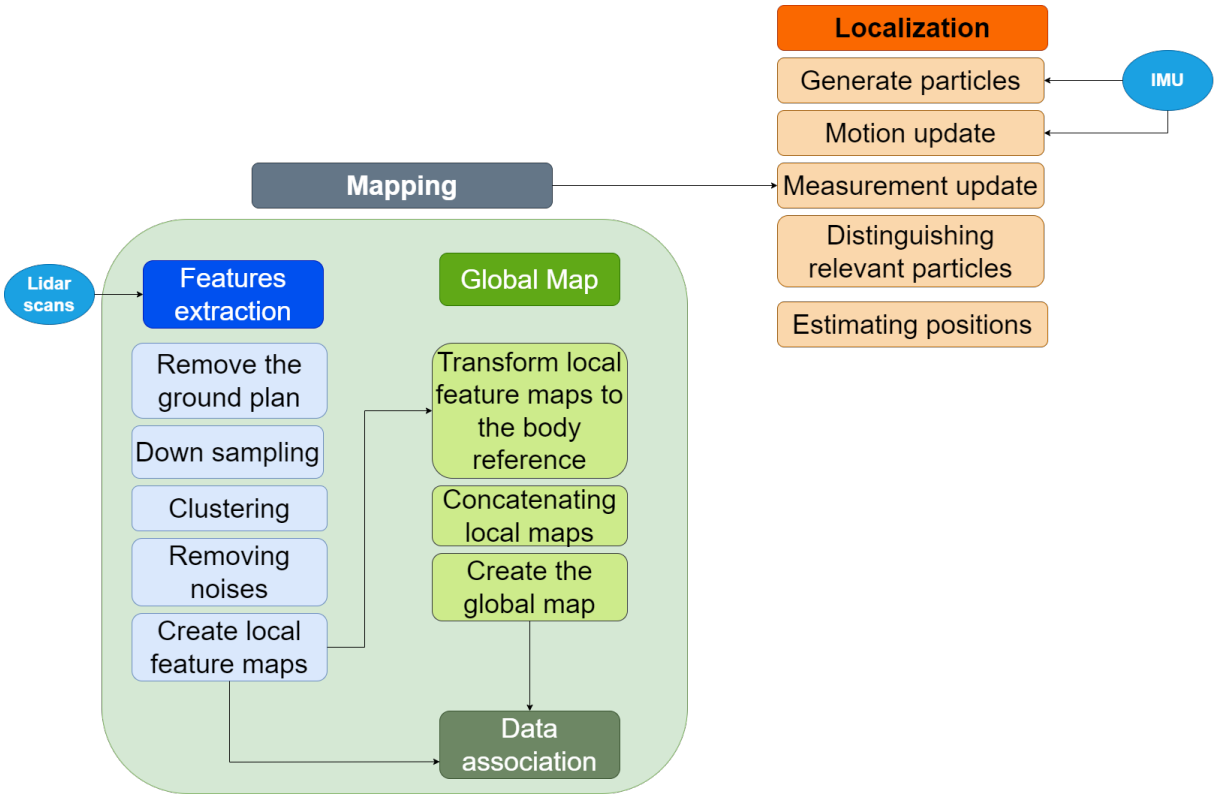


Figure 4.1: Presentation of the framework used in this work, which consists of two main components, mapping and localization. Mapping is responsible for extracting the features and creating the global map. Localization is responsible for calculating the positions

and create a local feature map for each scan. The global mapping step is set up to create a global map of the environment by converting all local feature maps into a body reference coordinate system and then merging them all. The localization phase then calculates the position of the vehicle based on the vehicle motion provided by the IMU and based on the results of the mapping step. Each of the steps (and stages, components) is explained in more detail below.

4.3.2 Mapping

To create a map of the environment, we use Lidar scans to represent objects in the environment with 3D coordinates, where the accuracy and robustness of these scans to changes in the environment are important. However, Lidar scans contain a huge amount of data points that contain noise and irrelevant information that needs to be removed to create a lightweight map and speed up the localization process. Below, we demonstrate our relevant feature extraction techniques to help create the global map.

- **Feature extraction:**

Acts as a pre-processing step, which receives data from the 3D lidar point cloud and transforms it into usable inputs, as the huge number of point data can especially affect performance and consume huge time for execution, which increases the need to reduce it without losing relevant information. We processed each lidar scan data with a series of filters; ground plane removal, downsampling of points, clustering, and removing noises. Let us define a scan of 3D Lidar points by:

$$P = [p_0, p_1, ..., p_N] \tag{4.4}$$

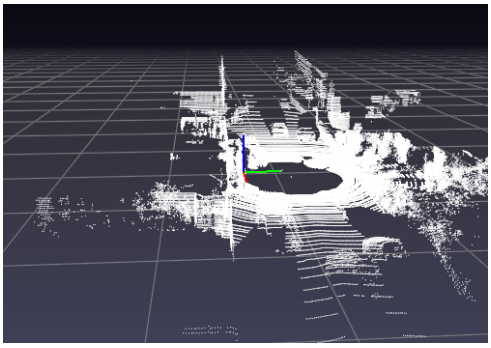
Such that

$$p_i = [x_i, y_i, z_i] \tag{4.5}$$

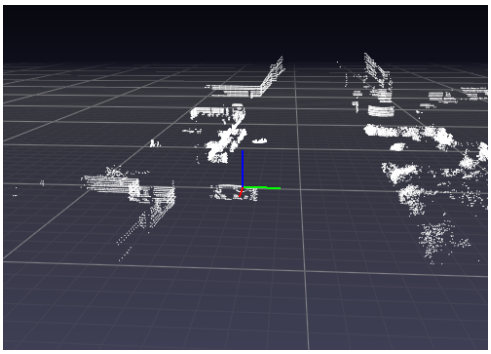
Each scan P is processed as follows. First, the 3D Lidar point scans provide an overview of the environment which certainly contains noise and information that we will not examine in our architecture, such as the ground plan. Since the ground plane did not show any hard distinguishable features that could help the matching process we **remove the ground plane** to reduce time. Our algorithm for removing the ground plane is take into consideration the points whose z-axis is above 0.1 of all points within the scan P. A **down sampling** method was applied to reduce the cost of feature extraction. The points are grouped into voxels (3D cube which is like pixels in images). Then, each occupied voxel generates exactly one point by averaging all the points in it. Finally, we applied a **clustering** phase to the remaining points. We tested several clustering methods to find the best one that meets the criteria of speed and representativeness. According to Table.1, fuzzy c-means [194] has recorded the lowest log time, which means that it takes less time to generate cluster functions. Given its ability to handle stochastic systems, it will be our choice in this work. To remove the noisy points and preserve the local feature map, we used the Density-Based Spatial Clustering of Applications with Noise (Dbscan) algorithm. We tracked features that only formed lines, where each line must contain a minimum number of points and the distance between them must be small. (An example of the process was illustrated in Figure.2)

- **Create global Map:**

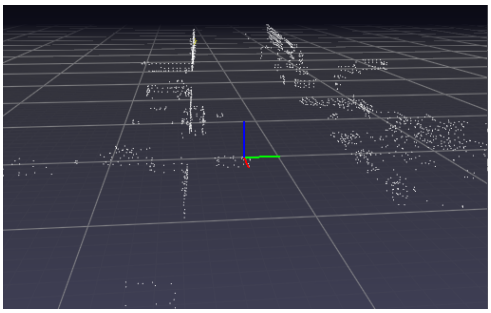
It is a very important step before the main task of localization. It is the process of creating a map of the environment based on the data collected by the sensors (lidar scans). In our case, we transform the local feature maps resulting from the feature extraction workflow into a **reference coordinate** system that unifies the viewing angle and places each



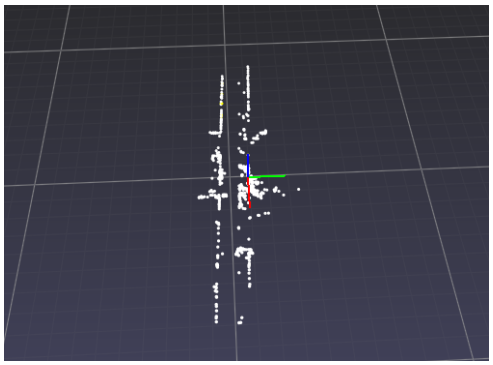
(a) Lidar 3D points scan



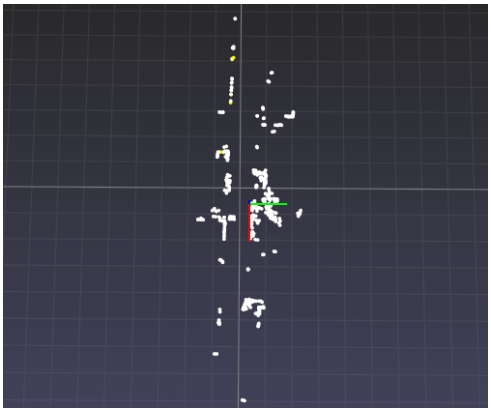
(b) Removing the ground plan Scene



(c) Downsampling process



(d) Features extraction



(e) Removing noises

Figure 4.2: Example of the features extraction workflow. Here we took K-means as a clustering algorithm (for image clarity it is recommended to zoom in the image (d))

Table 4.1: Time consumption (mm: ss) to execute the features extraction workflow with the specified clustering methods tested in the sequences 0001 from the Kitti data set [187]

Clustering method	**Time cost**
Growing Neural Gas	01:24
KMeans	01:38
fuzzy c-means	**00:43**
Hierarchical clustering	01:07
Gaussian mixture model	00:49
Self Organizing maps	02:06
Agglomerative Clustering	01:18
Particle Swarm Optimized Clustering	00:52

scan on its corresponding timestamp (see Figure.3a). furthermore, We concatenate the transformed scan in order to create the global map of the trajectory and the environment (see Figure.3b).

Let we note:

$$k \in [0, M_f] \qquad M = \{m_k\} \tag{4.6}$$

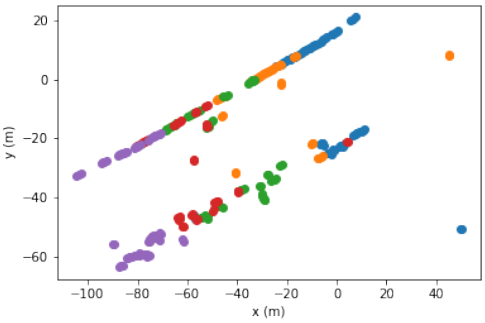
M_f number of feature points in the map, m_k the feature point at position k, M contain features of the global map.
and let also define:

$$k \in [0, N_f] \qquad F^t = \{f_k^t\} \tag{4.7}$$

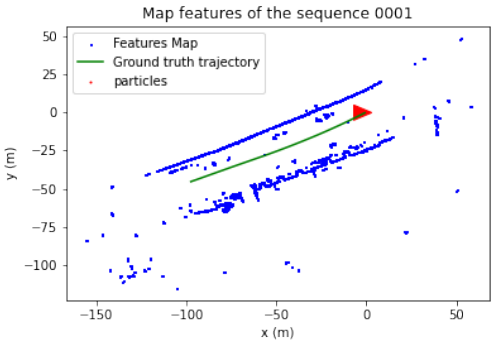
N_f number of feature points in the lidar scan t, f_k^t the feature point at position k in the scan t, F^t list of features at scan t which contain features of the local map at scan t.

- **Data association:**

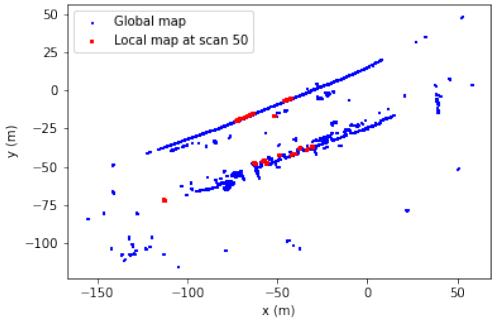
It is the task of finding a part of the global map that matches the local map at scan t; that is, we search for similar points of the local map F^t (local map at scan t) with the corresponding part of the global map $M_{n(t)}$ Such that $n_(t)$ is the index of the matching global feature map. Here, a Kd-tree algorithm was used to find the closest points to a local future map. An example of the process is given in Figure.3c



(a) Example of transforming and concatenating 5 scans from the Kitti dataset, sequence 0001. Each scan (red, blue, green points) occupied its suitable place that corresponds to each timestamp.



(b) The results of the mapping phase after concatenation and transformation. The blue dots are the feature map, the red triangle is the particle initialization and the green line is the ground truth.



(c) Example of data association of the scan 50 of the sequence 0001 of the Kitti dataset with global map

Figure 4.3: Illustration of the process of creating the global map of the environment

4.3.3 Localization:

After executing the mapping stage. The map is well prepared for the localization phase. Here present our contribution of the modified clustering particle filter that takes the IMU inputs (translation and rotation) and the results of the mapping stage (global and local maps). For simplicity, it is important to understand what the particle filter is and why we use it, which makes it easier to explain paper contributions in depth. Particle filters are used to correct the IMU sensor error (translation and rotation) by **generating** a series of particles $X = \{X_{Par_i}\}$ around an initial state of the GPS system representing the candidate positions (Figure. Figure. 4a blue dots). Each candidate is associated with a weight w_{Par_i} which is uniformly initialized. After the AV moves to the next state, the IMU computes the transformation inside it, which is usually error-prone. Each particle followed this transformation (**motion updates** see Figure.Figure. 4b)

$$X^t = \{X^t_{Par_i}\}, \quad X^t_{Par_i} = R^t * X^{t-1}_{Par_i} + T^t \tag{4.8}$$

$X^t_{Par_i}$ is 4 by 4 homogeneous matrix created after the generation of the x,y coordinates and the θ orientation uniformly, $i \in [0, N]$ Such that N is the number of particles, R^t and T^t are, respectively, the rotation and translation matrices at scan t.

Moreover, the second stage in the particle filtering algorithm is the **measurement update**, which aims to calculate the weights $\{w_{Par_i}\}$. These weights play the role of a judge, such that a higher weight value means that the associated particle is close to the actual position of the state. According to [31], these weights can be updated by:

$$w^t_{Par_i} = \prod_{p \in [0,t]} p\left(F^p \mid X^p, M_{n(p)}\right) \tag{4.9}$$

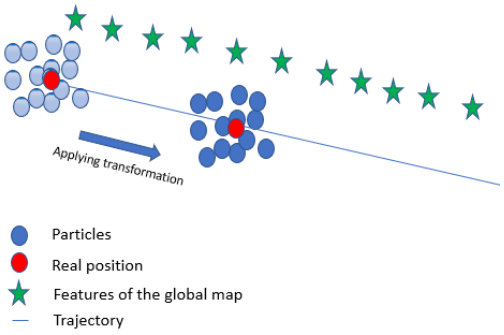
Such that

$$p\left(F^p \mid X^p, M_{n(p)}\right) := \mathcal{N}\left(\left\|X^p F^p - M_{n(p)}\right\|, \sigma\right) \tag{4.10}$$

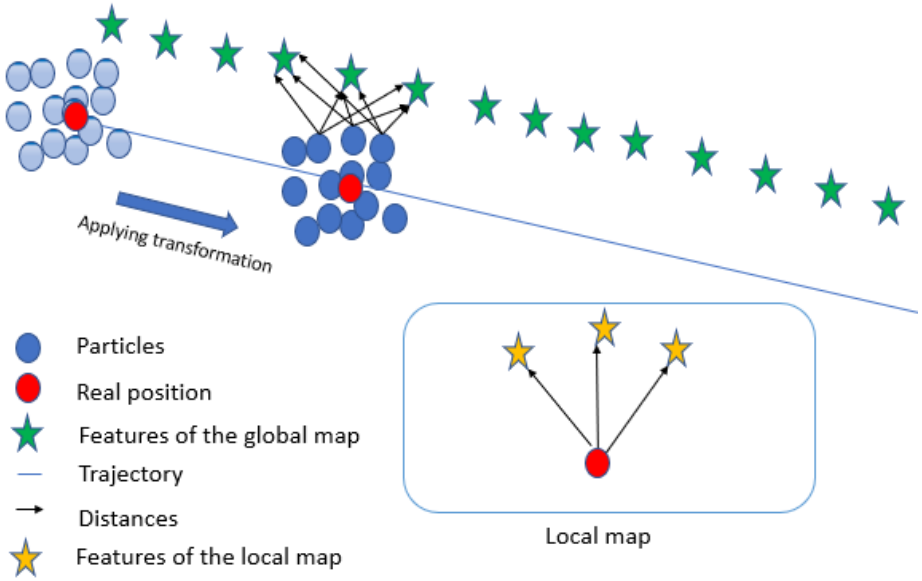
The product in $X^p F^p$ gives the distances between the particles and features in scan p, σ is an isotropic position uncertainty depending on the reference landmarks. $n(k)$ is an index of the global features associated with the k_{th} local feature (provided by kd-tree), and $\mathcal{N}$ is the normal distribution. weights are like a probability of contribution of corresponding the particle to calculate the real position. The idea is to compare the



(a) Generate particles



(b) Applying the IMU transformation



(c) Compare the distances of the particles with the futures of the global map (features in green) and the real distance

Figure 4.4: Explication of the workflow of the particle filter (the images are not real. They are generated to explain in depth the process.)

normal curve of particle distances to features on a global map with what we have on the corresponding local map in each scan. The distances to features in the local map are the actual distances. Consequently, we compare them with those of the particles. The closer the distances are, the greater the contribution to the position calculation. The correspondence between the local and global maps is established by the data association step (see. Figure. 4c for an illustration)

The state estimation at scanning t

$$pos^t \approx \sum_{i=1}^{N} w_{Par_i}^t X_{Par_i}^t \tag{4.11}$$

Such that

$$\sum_{i=1}^{N} w_{Par_i}^t = 1 \tag{4.12}$$

N is the number of particles.

The accuracy of the method depends mainly on the number of particles, the higher the number of particles, the higher the probability of convergence to the real state. However, increasing the number of particles causes additional energy costs and increases time consumption, which is not recommended for self-driving vehicles [197].

To speed up this process while maintaining accuracy, we proposed to choose efficiently some relevant particles. This step should be done directly after the **motion update** step in order to reduce the computational time of calculating weights and calculating positions also. We suggest two approaches to choose those points which will be thoroughly analysed in the next section.

- **Using clustering approach:**

Clustering is a well-known approach to store information in a smaller number of clustering centers. This reduces the time cost and preserves the particle information.

Let us define.

$$C^t = \{C_i^t\} \tag{4.13}$$

$i \in [0, N_c]$, N_c clusters number, C_i^t is the center cluster i at scan t

Consequently,

$$w_{C_i}^t = \prod_{p \in [0,t]} p\left(F^p \mid C^p, M_{n(p)}\right) \tag{4.14}$$

Such that

$$p\left(F^p \mid C^p, M_{n(p)}\right) := \mathcal{N}\left(\left\|C^p F^p - M_{n(p)}\right\|, \sigma\right) + \epsilon \tag{4.15}$$

The state estimation at scan t, $\mathcal{N}$ is the normal distribution

$$pos_C^t \approx \sum_{i=1}^{N_c} w_{C_i}^t C_i^t \tag{4.16}$$

Such that

$$\sum_{i=1}^{N_c} w_{C_i}^t = 1 \tag{4.17}$$

The K-means algorithm was chosen to perform this task regarding its convergence capability and its ability to incorporate clusters of different shapes.

- **Using sigma points approach:**

After the **motion update** step, we calculate the mean and covariance of the particles, x and P, respectively.

In order to get those sigma points, we need to find the minimum of these formulas:

$$\min_{\langle \mathcal{X}, w \rangle} c(\langle \mathcal{X}, w \rangle, r, P(\mathbf{x})) \quad \text{subject to } \xi(\langle \mathcal{X}, \mathbf{w} \rangle, \mathbf{r}, \mathrm{p}(\mathbf{x})) = 0 \tag{4.18}$$

χ are the sigma points, ω are the weights, r is the number of sigma points which is equal to $2L + 1$ and L is the dimension of x. In this paper, we chose three different methods: Van der Merwe 2004 dissertation [197], Julier and Jeffery K. Uhlmanns [198], and the simplex method of Phillippe M., and Dominique C. [199] and we merge their results to investigate the benefit of all these methods.

Without diving into the theory of each of them, we present their method to distinguish the sigma points.

According to Julier et al. the points can be generated by using these formulas:

$$\begin{aligned} \chi_J[0] &= \quad x \\ \chi_J[i] &= x + \left(\sqrt{(L+\kappa)P}\right)_i \quad i = 1, \ldots, L \\ \chi_J[i] &= x - \left(\sqrt{(L+\kappa)P}\right)_i \quad i = L+1, \ldots, 2L \end{aligned} \tag{4.19}$$

χ_J is the sigma points generated, and κ is a scaling factor that can reduce high order errors. Note that we need to find the matricial root square weather using the Cholesky method or other. However, regarding its fastness, we choose the Cholesky decomposition.

According to Van der Merwes the points can be generated by using these formulas:

$$\begin{aligned} \chi_M[0] &= x \\ \chi_M[i] &= x + \left(\sqrt{(L+\lambda)P}\right)_i \quad i = 1, \ldots, L \\ \chi_M[i] &= x - \left(\sqrt{(L+\lambda)P}\right)_i \quad i = L+1, \ldots, 2L \end{aligned} \tag{4.20}$$

$\lambda = \alpha^2(L + \kappa) - L$ is a scaling parameter.α determines the spread of the sigma points around x, and κ is a scaling factor that can reduce high order errors.

According to Phillippe M. et al the points can be generated by using these formulas:

$$\begin{aligned} &\chi_P[i] = x - I^{(i)} \quad i = 1, \ldots, L+1 \\ &I^{(i)} = \sqrt{P}\bar{I}_r^{(i)} \end{aligned} \tag{4.21}$$

Such that the vectors $I_r^{(i)}$ are the columns of the matrix noted $\left[\tilde{I}_r^*\right]$ recursively defined by

$$\begin{cases} \left[\bar{I}_1^*\right] = \left(-\frac{1}{\sqrt{2\alpha}}\frac{1}{\sqrt{2\alpha}}\right), \quad \alpha = \frac{p}{p+1} \\ \left[\bar{I}_d^*\right] = \begin{pmatrix} & & & 0 \\ & \left[\bar{I}_{d-1}^{(i)}\right] & & \vdots \\ \frac{1}{\sqrt{\alpha d(d+1)}} & \cdots & \frac{1}{\sqrt{\alpha d(d+1)}} & \frac{-d}{\sqrt{\alpha d(d+1)}} \end{pmatrix}, 2 \leq d \leq p \end{cases} \tag{4.22}$$

Finally, we concatenate our sigma points.

$$\chi^t = \{[\chi_J, \chi_M, \chi_P]^t\} = \{\chi_i^t\}$$

$i \in [0, N_\chi]$, N_χ sigma points number, χ_i^t sigma point of the element i at scan t
Consequently,

$$w_{\chi_i}^t = \prod_{p \in [0,t]} p\left(F^p \mid \chi^p, M_{n(p)}\right) \tag{4.23}$$

Such that

$$p\left(F^p \mid \chi^p, M_{n(p)}\right) := \mathcal{N}\left(\left\|\chi^p F^p - M_{n(p)}\right\|, \sigma\right) + \epsilon \tag{4.24}$$

The state estimation at scan t, $\mathcal{N}$ is the normal distribution

$$pos_\chi^t \approx \sum_{i=1}^{N_\chi} w_{\chi_i}^t \chi_i^t \tag{4.25}$$

Such that

$$\sum_{i=1}^{N_\chi} w_{\chi_i}^t = 1 \tag{4.26}$$

4.3.4 Evaluation

Our metrics included six error measurement Δ_{pos}, Δ_{lat}, Δ_{lon} and Δ_{ang} denote the mean absolute positional, latitudinal, longitudinal, and heading errors, while $RMSE_{pos}, RMSE_{ang}$ represents the corresponding root mean squared errors.

$$\Delta_{lat} = \frac{\sum_{i=1}^{n}\left|X_i - X_i^{'}\right|}{n} \qquad \Delta_{lon} = \frac{\sum_{i=1}^{n}\left|Y_i - Y_i^{'}\right|}{n}$$

$$\Delta_{pos} = \frac{\sum_{i=1}^{n} \left|\left|pos_i - pos_i^{'}\right|\right|_2}{n} \qquad \Delta_{ang} = \frac{\sum_{i=1}^{n} \left|\theta_i - \theta_i^{'}\right|}{n}$$

$$RMSE_{pos} = \sqrt{\frac{\sum_{i=1}^{n} \left|\left|pos_i - pos_i^{'}\right|\right|_2^2}{n}} \qquad RMSE_{ang} = \sqrt{\frac{\sum_{i=1}^{n} \left|\theta_i - \theta_i^{'}\right|^2}{n}}$$

Such that n is the key frames number, $pos_i = [X_i, Y_i]$ is the predicted state, $pos_i^{'} = [X_i^{'}, Y_i^{'}]$ is the actual state, θ is the predicted angle, $\theta_{'}$ actual state

4.4 Results

In this section, we analyze the performance of our method for locating the vehicle on the trajectory in the short, medium, and long term, and investigate which method is more accurate and efficient among the two presented in our methodology; i.e., clustering-based and sigma-point based methods.

4.4.1 Experiments

We have tested our method in two benchmark data sets; Kitti [187], and Pandaset [195]. Kitti data contain several sequences driven around the mid-size city of Karlsruhe, The vehicle was equipped with two high-resolution colour and grey-scale video cameras. Accurate ground truth is provided by a Velodyne laser scanner and a GPS location system. We have chosen to test our method in different categories provided in the data set, including city, residential, Road, Campus, and Person. This advantageous data set enables the test of various environmental changes and investigate the performance of the vehicles in short and long-term localization trajectories. In addition, we examined the performance of our method on a second data set that included a complex showcase of urban driving scenarios, including steep hills, construction, heavy traffic and pedestrians, and a variety of weather and light conditions in the morning, and afternoon, dusk, and evening. Pandaset vehicle was equipped also with IMU/GPS information and Lidar, cameras measurement. The tests were conducted in an i7 8th generation pc with 20 GB Ram. However, a GPU is recommended for this task. We have fixed some configuration in the features extraction process, including the voxel size $(0.5 * 0.5 * 0.5)$ per voxel, the number of cluster of 100 clusters (This choice depend mainly on the execution cost and the representation of features in the environment) and we keep as defaults other fuzzy c-means parameters. For

the DBSCAN algorithm, we have fixed 2 points as a minimum sample to fit the lines and 1 m unit as the maximum distance between two samples for one to be considered as in the neighbourhood of the other which is very important parameter to choose accurately. In the localization stage, we have fixed a 10 cluster for the localization based on k-means clustering and we generate 100 particles to run the localization tests for both approaches (clustering or sigma points). The justification of this choice will be discussed in the next section. In the sigma localization method, we set kappa=0 which gives the standard unscented filter, $L = 3$, $\alpha = .00001$. We have run the Mapping stage for one time to create the local and global maps.

4.4.2 Parameters discussion

We have analyzed the appropriate configuration to obtain better results in terms of accuracy and time cost. Table 2 shows that, for a fixed number of 2000 particles, 10 cluster centers recorded better results and represent a practical parameter that serves for accuracy, energy and time cost. Furthermore, Table 3 shows that, for a cluster of 10 fixed centers, we obtained that 100 particles are sufficient to obtain good accuracy in a reasonable time These results justify the choice of our configuration (10-center cluster and 100 particles) and constitute our main contribution to reduce time consumption. However, some perturbation of the accuracy occurs due to the initialization of the particle filter or the reduced number of center clusters.

Table 4.2: Testing the performance of the method with different numbers of cluster particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs.

Number of cluster	10	50	100
RMSE_{pos} (m)	0.127	0.1479	0.552
Time cost (s)	01:25	03:53	06:42

4.4.3 Short-term localization discussion

The first type of experiment was conducted to investigate the performance of our two models (i.e., based on clustering techniques, or based on sigma points) and to test the extent to which each is able to accurately locate the vehicle without wasting too much

Table 4.3: Testing the performance of the method with different numbers of particles. The performance was based on five different sequences of different categories with a total of 389 frames, and we recorded the sum of the accuracies and time costs.

Number of Particles	100	1000	2000
RMSE_{pos} (m)	0.206	0.3704	0.1802
Time cost (s)	00:23	00:55	02:38

time and energy. Therefore. We have distinguished a number of short Kitti and Pandaset dataset sequences that contain several environmental changes and different trajectory shapes. The results in Table 4, Table 5 show the accuracy of the two models in locating the vehicle within the chosen sequences and the time consumed in each. It was shown that our models are able to accurately determine the positions in a short computation time; **41 s** to locate the vehicle in 544 frames (105 s across all sequences) with **0.24 m** average absolute positioning error, demonstrating the effectiveness of using clustering techniques to reduce the time required to calculate the positions. Meanwhile, the sigma point based localization used **5 s** to locate the vehicle in 544 frames (105 s across all sequences) with **0.16 m** in the average absolute positioning error which is super fast and accurate, especially for real-time applications. Based on the mean and covariance of the particles after the motion update, the sigma points method manages to perfectly choose the "winning" points that efficiently contribute to the calculation of the positions. According to the analysis performed above, the localization based on sigma points outperforms the localization based on clustering both in terms of accuracy and speed. Therefore, it will be our choice in the remainder of this review.

Table 6 confirm our statement; a mean absolute positioning error of 0,18 m was obtained across all sequences in the pandaset [195] reference data set over 400 frames in a 5(s) of execution time which demonstrates again the robustness and efficiency of our method to localize the vehicle in different environmental scenarios.

We manually tested Scheafer's method on sequence 0001 (city category) of the Kitti data set and obtained 0.06 m absolute pose error and 0.1 *degree* absolute rotation error. The run took 19 s, which was more time and energy than what we obtained in Table 7. 0.054 m absolute pose error, 0.01 *degree* rotation error in 1 s of execution. We also see in Table 7 that we have such excellent rotation and translation errors that we outperform all competing methods, which is due to the robust technique that distinguishes

Table 4.4: Accuracy and time consumption of the Localization based on the clustering in the Kitti dataset from different categories sequences.

Category	Seq	frames no	t_{Feat} (s)	$t_{loc}(s)$	$\Delta_{pos}(m)$	$\Delta_{lat}(m)$	$\Delta_{lon}(m)$	$\Delta_{ang}(^\circ)$	$RMSE_{pos}(m)$	$RMSE_{ang}(^\circ)$
City	0001	108	00:02	00:06	0.08	0.01	0.07	0.009	0.11	0.01
Residence	0035	131	00:04	00:10	0.06	0.01	0.05	0.04	0.08	0.06
Road	0027	188	00:07	00:16	0.10	0.031	0.09	0.005	0.005	0.0037
Campus	0034	49	00:03	00:04	0.0063	0.003	0.004	0.005	0.008	0.005
Person	0053	68	00:02	00:05	0.0009	0.0007	0.0005	0.002	0.001	0.002
SUM		544	00:18	00:41	0.2472	0.0547	0.2145	**0.061**	0.204	**0.0807**

Table 4.5: Accuracy and time consumption of the Localization based on the sigma points in the Kitti dataset from different categories sequences

Category	Seq	frames no	t_{Feat} (s)	$t_{loc}(s)$	$\Delta_{pos}(m)$	$\Delta_{lat}(m)$	$\Delta_{lon}(m)$	$\Delta_{ang}(^\circ)$	$RMSE_{pos}(m)$	$RMSE_{ang}(^\circ)$
City	0001	108	00:02	00:01	0.054	0.013	0.05	0.01	0.067	0.011
Residence	0035	131	00:04	00:01	0.01	0.004	0.015	0.048	0.025	0.088
Road	0027	188	00:07	00:01	0.084	0.019	0.07	0.005	0.098	0.006
Campus	0034	49	00:03	00:01	0.0064	0.004	0.005	0.005	0.0097	0.005
Person	0053	68	00:02	00:01	0.00087	0.0007	0.0004	0.0012	0.001	0.0015
SUM		544	00:18	00:05	**0.15527**	0.0407	**0.1404**	0.0692	**0.2007**	0.1115

Table 4.6: Accuracy and time consumption of the Localization based on the sigma points in the Pandaset dataset from various sequences

Seq	frames no	t_{Feat} (s)	t_{Loc} (s)	$\Delta_{pos}(m)$	$\Delta_{lat}(m)$	$\Delta_{lon}(m)$	$RMSE_{pos}(m)$
100	80	00:03	00:01	0.269	0.056	0.252	0.331
109	80	00:03	00:01	0.207	0.041	0.195	0.267
117	80	00:04	00:01	0.237	0.043	0.227	0.295
139	80	00:02	00:01	0.0825	0.048	0.063	0.107
158	80	00:03	00:01	0.149	0.05	0.134	0.193

t_{Feat} **:**	Is the cost of time to calculate the features extraction of all the sequences
t_{Loc} **:**	Is the cost of time to execute the Localization process

Table 4.7: Evaluation of the accuracy of the strategies of Sefati et al. and A. Schaefer and of our methodology. The results of Weng et al. and Kümmerle et al. are not directly comparable and are presented for qualitative analysis only.

Methods	$\Delta_{pos}(m)$	$RMSE_{pos}$	$\Delta_{lat}(m)$	$\sigma_{lat}(m)$	$\Delta_{lon}(m)$	$\sigma_{lon}(m)$	$\Delta_{ang}(^\circ)$	$\sigma ang(^\circ)$	$RMSE_{ang}(^\circ)$
Kümmerle et al.[27]	0.12	—	0.07	—	0.08	—	0.33	—	—
Weng et al.[29]	—	—	—	0.082	—	0.164	—	0.329	—
Sefati et al.[26]	—	0.24	—	—	—	—	—	—	0.68
A. Schaefer et al.[31]	0.096	0.111	0.061	0.075	**0.060**	0.067	0.133	0.188	0.214
Charroud. A et al. [43]	0.12	0.141	**0.059**	0.09	0.08	0.05	**0.043**	0.078	**0.057**
Ours	**0.077**	**0.085**	**0.0195**	**0.025**	0.071	**0.059**	**0.04**	**0.04**	**0.05**

the sigma points from the particles based on the mean and covariance. Scheafer et al. also record a small absolute pose error with respect to the data association followed in that work. Extracting pole features, as done in Scheafer et al. also significantly helps to achieve these results. However, poles, such as trees, or even the use of walls, do not exist in all environmental settings, which especially affects the localization process. Non-semantic feature extraction guarantees the existence of a certain number of features that will represent the data, which ensures the continuity of the data association workflow. Moreover, this extended work outperforms the one of Charroud. A et al. [43] in term of accuracy and robustness where they have used Gaussian Mixture Models (GMM) to find the global map which affects the data association process. Even the fastness of the process, GMM with a fixed number of clusters misleads the matching process. In contrast, we have concatenated all the features received from the local map to ensure the reliability of the matching process.

4.4.4 Long-term localization discussion

We tested our method on medium and long duration sequences from the Kitti odometry dataset (different from the short sequences in Table 5 and Table 7). The method was evaluated on sequence 00 and sequences 05-10 over 17554 frames and 29 minutes of running. According to Table 8, our method produced excellent results; 0.69 m in average absolute positioning error across all sequences. We compared our method with more advanced registration methods, including LOAM_Velodyne [200] LOAM_Velodyne [201], LeGO- LOAM [202], A- LOAM [203], SGLO [205], LO-Net [204], LO-Net-M [204], and ours. The metric used by these methods is called t_{rel}; the relative translation drift averaged over trajectories from 100 to 800 m. In our case, however, we took the average absolute error over all trajectories. Since the approach used to locate the vehicle differs between us and the registration methods. The analysis was performed approximately to provide a general inspection of the performance of the methods. Table 8 shows that our method outperforms the other methods in 4 sequences scenario, including trajectories of long duration. Moreover, we obtained competitive results in other sequences. Our method records a positioning error of 1.16 m in sequence 08, a long trajectory of 08 min :38 s driving. Again, our method proves its speed by performing the localization task in only 22s. On another side, our method was capable to generate accurately the global map of the environment for these long term trajectories which showed the performance of our

Table 4.8: Accuracy evaluation of our method with state-of-art registration methods

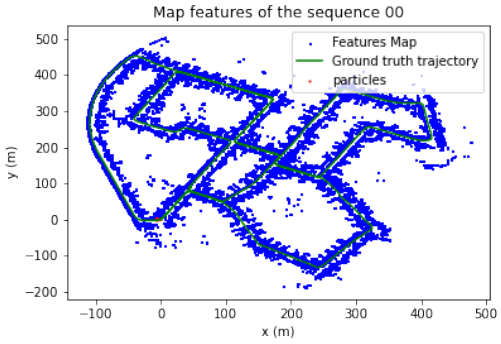
Seq	LOAM [200]	LOAM Velodyne [201]	LeGO- LOAM [202]	A- LOAM [203]	SGLO [205]	LO-Net [204]	LO-Net-M [204]	Ours
00	0.78	3.410	1.347	0.791	**0.775**	1.47	0.78	0.95
05	0.57	1.318	0.887	**0.504**	0.596	1.04	0.62	0.73
06	0.65	1.007	0.87	0.615	0.483	0.71	0.55	**0.297**
07	0.63	1.256	0.77	0.453	0.515	1.70	0.56	**0.391**
08	1.12	2.159	1.337	1.107	**1.059**	2.12	1.08	1.16
09	0.77	1.437	1.282	0.739	0.733	1.37	0.77	**0.71**
10	0.79	1.912	1.675	1.006	0.977	1.80	0.92	**0.65**

features extraction workflow.

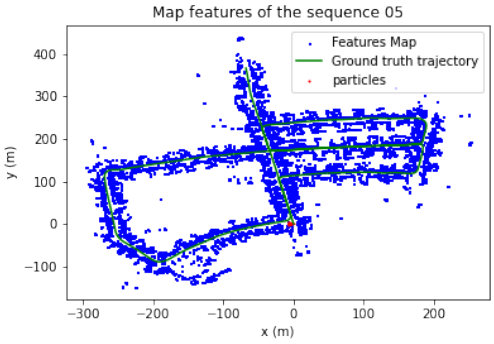
4.5 Conclusion

In this paper, we proposed a multi-source localization and mapping method for self-driving vehicles based on non-semantic feature extraction. The fuzzy c-means method was used to find patterns in LiDAR scans. The cluster centers were filtered by a DBSCAN algorithm to remove the noises and were used to create the local map features. In addition, the global map was created by merging local map features and transforming them into a reference coordinate system which facilitates data association. The localization process was carried out using a particle filter with modifications which is a technique to distinguish some relevant features from the generated particles in order to speed up the calculation process and track the trajectory positions efficiently. We have proposed two approaches to distinguish those points; clustering-based and sigma points based approaches. sigma points has registered the best accuracy in a small calculation time which was our choice for the next experiments and comparisons.

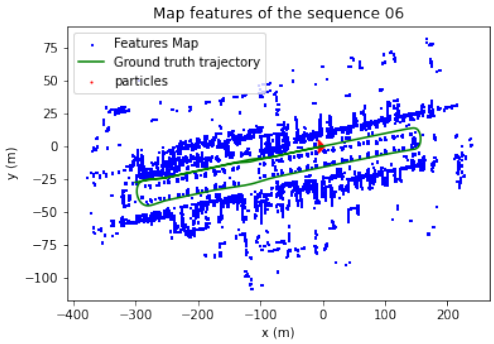
Our proposed approach (sigma points) achieved competitive results in terms of accuracy. but does so in significantly less time compared to the state-of-the-art methods



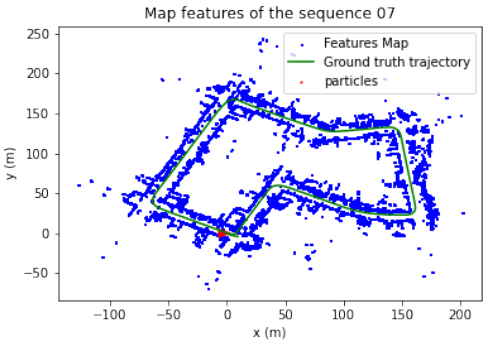
(a) Seq00



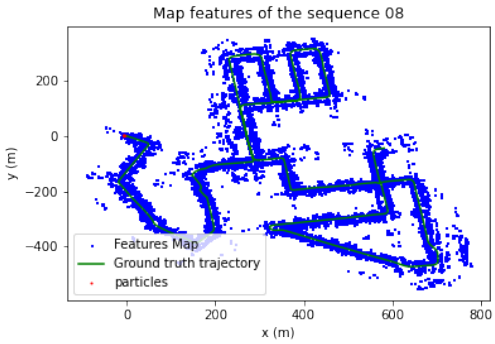
(b) Seq05



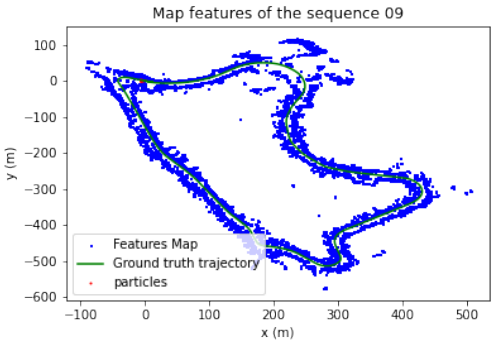
(c) Seq06



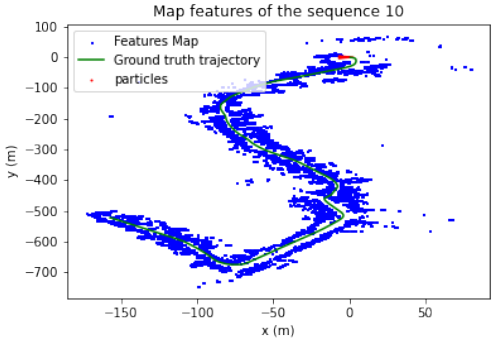
(d) Seq07



(e) Seq08



(f) Seq09



(g) Seq10

Figure 4.5: Global maps

evaluated on the Kitti and pandaset benchmark dataset; 0.16 m in the average absolute positioning error of all the short-term sequences of the Kitti dataset in 5 s of execution. Also, we obtained 0.18 m in the average absolute positioning error of all the short-term sequences of the Pandaset dataset. We have tested extensively our method in long-term sequences and we obtained 0.69 m of the average absolute positioning error of all sequences, which demonstrates the robustness, reliability, and efficiency of our method. Meanwhile, we have compared our method with recent state-of-the-art works. Indeed, our method outperforms all of them in short-term sequences, especially, the Schaefer et al. method [31] where we registered 0.077 m, while 0.096 m is registered by Schaefer's method. Furthermore, we have obtained competent results in the long-term sequences.

However, some accuracy deviations occurred due to poor initialization of the particle filter or poor choice of cluster and number of particles. Therefore, our future work will mainly focus on finding the best initialization and investigating the parameters to find the best configuration based on optimal control or using machine learning.

CHAPTER 5

RAPID LOCALIZATION AND MAPPING METHOD BASED ON ADAPTIVE PARTICLE FILTERS

5.1 Introduction

The recent advances in the development of Autonomous Vehicles (AVs) offer great potential for improving road safety. AVs offer a market-ready solution to reduce road-related crashes and fatalities [4]. Achieving these goals requires an effective autonomous vehicle localization system that can locate the vehicle in its environment. The combination of positioning and orientation techniques for localization and appropriate mapping methods are required for trajectory planning and safe navigation of AVs [18].

The classical approach for locating AVs is to use a Global Navigation Satellite System (GNSS). Triangulation of at least three GNSS signals provides information for locating AVs on the road. However, GNSS reliability is compromised by visibility problems in tunnels, canyons, under bridges, and dense tree canopies, as well as by multipath reflections and signal delay [174, 175, 176]. A common approach employed during the absence of GNSS signals is to use the Inertial Navigation Systems (INS), which can provide continuous position and attitude information based on the vehicle's acceleration and heading speed provided by the accelerometer and the gyroscope, respectively [176]. However, the accuracy of the INS deteriorates exponentially during the double integration of the acceleration measurements to position and of the azimuthal velocity measurements to azimuth [177].

Similar to the accelerometers of the INS, wheel odometry has been investigated as a solution for position determination in GNSS-deprived environments [107, 108]. By using measurements from wheel encoders (which measure the wheel speed of the vehicle) and integrating the linear velocity of the vehicle derived from the wheel speed, it is possible to determine the position of the vehicle with one less integration step compared to the accelerometer measurement [108]. However, like the INS, wheel encoder measurements suffer from errors that are exponentially amplified when integrated into position [108].

To provide a more accurate position and orientation estimation, needed for a safer and more effective navigation solution in GNSS-deprived environments, several machine learning methods based on recurrent neural networks [176, 177, 107, 108, 178, 179, 106, 211] have been proposed to learn the errors present in the wheel encoder, accelerometer, and gyroscope measurements. However, Black Box type of models, as usually encountered in deep learning, can pose several challenges, as they require large amounts of data to be properly trained in order to provide reliable prediction results, in addition to their decisions not being transparent to human users.

Particle Filters (PF) have also been shown to be capable of accurately modelling measurement errors in similar types of applications. PFs have several advantages that can be used for localization. For example, PFs can handle nonlinear trajectories [86], can track multiple objects varying in time [181], and can handle occlusion and overlap [181]. Moreover, the PFs approach does not require prior assumptions, such as a Gaussian distribution of the data, such as the Kalman filter [182]. In its early days, PF was used in signal processing to estimate states based on observed variables [188]. Since then, PF has been adapted to solve localization problems [26, 27, 29, 31, 180]. In practice, PF (also called sequential Monte Carlo) uses the Markov assumption based on a concept that relates the state (in this case, the position) X_t to the previous state X_{t-1}. The algorithm begins by generating thousands of particles representing candidate positions. A weight is assigned to each particle. The higher the weight's value, the higher the probability that the particle is in the vehicle's position. PFs offer the opportunity to explore the advantages of data from multiple sources.

Nevertheless, the above-described methods are not capable of providing lane-level accuracy using the information provided only by the INS and/or wheel encoder sensors. Camera and Light Detection and Ranging (LiDAR) are sensors commonly combined with the INS to provide lane-level localization. They are widely used in intelligent vehicles to

provide an accurate representation of the vehicle's environment, i.e., the objects around the vehicle and the distances between them. A technique called "registration" [4] can be used to locate vehicles by aligning LiDAR scans and images from the camera. Typically, this method tracks the movement between keyframes, producing a transformation matrix that gives a translation vector T and a rotation matrix R from the first keyframe to the second keyframe [180]. Thus, the position of the vehicle follows the transformation described below in Equation (1).

$$X_t = RX_{t-1} + T \tag{5.1}$$

where X_t is the position of the vehicle at time t, and X_{t-1} is the position of the vehicle at time $t-1$.

The Iterative Closest Point (ICP) is one method that applies this registration concept [4, 189]. It matches LiDAR scans by minimizing the cost function

$$\arg\min_{R,T} \left\{ \frac{1}{M} \sum_{j=1}^{M} ||a_j - (Rb_j + T)||_2^2 \right\} \tag{5.2}$$

where $(a_j)_j$, $(b_j)_j$ are the 3D coordinate points of the two LiDAR scans A and B, respectively, and M is the number of points inside A (or B). The idea is to find a matrix of translations and rotations that accurately represents the similarities between these two scans. Thus, the vehicle's motion will also follow the same transformation as shown in Equation (5.1) [21]. However, due to measurement errors, the ICP algorithm is not able to provide good accuracy, especially in the long-term matching of nonlinear trajectories [196]. Therefore, the authors of [113] proposed the Normal Distribution Transform (NDT), which represents each group of 3D points included in a voxel (a 3D cube) by a probability distribution. The advantage of this representation is its ability to provide better results despite measurement errors and difficulties in detecting similarities between scans. However, these 'registration' methods are usually computationally time-consuming and do not work well when the number of feature points matched is not sufficient [123], which complicates the process of finding the optimal parameters R and T needed to minimize the cost function in Equation (5.2). Developing a highly accurate and real-time efficient registration method is still an active research area in autonomous driving [43].

In this study, we propose a fuzzy K-means clustering technique to represent the environment efficiently, which facilitates the process of finding similar parts between the scan of the local map t and the corresponding parts on the global map. The global map contains information about the object in the trajectory. After generating these maps (local

and global), an adaptive particle filter is used to localize the vehicle along the trajectory. The Adaptive Particle Filter (APF) is a particle filtering method supported by a resampling technique that allows the particles to follow approximately the actual positions of the trajectory and maintain accuracy. The APF uses the created local and global map to distinguish the best candidate particles for position calculation. In this paper, we extend the work conducted in [43] on non-semantic feature map reduction using K-means, global map generation using GMM, and localization using particle filters by adding a resampling technique to the particle filter (i.e., employing the adaptive filter) to obtain a more accurate localization solution. In addition, we further investigated different parameter settings of the particle filter and the effect of the resampling extension on the performance of our proposed localization and mapping technique.

5.2 Related Work

Three approaches to fuse information from different sensors were found to be the most popular. These include parametric filters, such as extended Kalman filters; nonparametric filters, such as particle filters; and finally, least squares approaches, such as beam adaptation and graph-based Simultaneous Localization and Mapping (SLAM). In general, these methods use measurements from motion sensors to track vehicle motion, i.e., translation and rotation between two consecutive keyframes, and explore measurement sensors, i.e., LiDAR scans or images, as criteria to accurately approximate positions. Map features represent objects from the vehicle's environment, or at least part of them, by their coordinates, which reduce the calculation time of the matching process and provide a clear representation of the map. There are two types of features: semantic and non-semantic features. Semantic features such as poles, trees, buildings, and sidewalks are widely used in feature mapping and vehicle tracking. The authors of [26] detected poles as feature landmarks by converting point clouds provided by the LiDARs into voxels, and the cells with an acceptable number of point clouds were connected vertically with a similar voxel. Each candidate pole has a certain number of connected cells (vertical sense). The landmarks were also fitted with cylinders to ensure reliability and extract pole parameters. A particle filter was used to perform the localization process. The method showed good accuracy on the Kitti dataset. Better localization accuracy was also achieved by poles and wall detection by Kummerl et al. [27]. In [29], the particle filter was used to localize the vehicle based on an algorithm that detects poles based on the intensity of the group

of points in a voxel grid map. The algorithm was found to be able to detect landmarks such as trees, streetlights, and telegraph poles and determine their boundaries. Moreover, it detects the starting and ending points of the poles. A probabilistic grid map was implemented in [31] to identify voxels with high beta probability distribution values. The authors investigated the idea of isolation (i.e., poles are usually isolated from the surrounding space) and proposed a mathematical formulation that assigns a score to a pole candidate based on the intensity values inside and outside the pole. The closer the value is to 1, the more likely it is a pole. Their approach also used a particle filter to perform the localization task and compared its performance to other published methods. An accumulative positioning error of 7.67 m was registered in the University of Michigan North Campus Long-Term vision (NCLT) dataset [190] over 147.3 km of trajectory. Moreover, 0.096 m of the positioning error was registered in sequence 0009 of the Kitti dataset [187]. In article [22], the same probability maps were also used to identify walls and buildings in the vicinity. A Kalman filter was then applied to locate vehicles using the author's dataset. In general, the integration of such semantic feature-based methods is computationally intensive. Moreso, these methods do not work efficiently in all environments, i.e., the environment must have texture and contain the desired features. Based on clustering methods, our proposed approach detects non-semantic features which exist in any environment, even with fewer objects' textures.

Recent works have used this approach, such as the family of Oriented fast and Rotating Beams—SLAM (ORB-SLAM) [183, 184, 142], which extracted the camera position in each frame by matching the ORB features between keyframes based on local bundle adjustment, and the ORB features were used to generate maps of the environment. A tracking step was proposed to assist the system in re-locating, location awareness, or matching frames. These feature points are stored in a robust DataBase (DB) architecture called DBoW2 [185]. The ORB-SLAM system was tested on real data and has achieved high accuracy in point tracking, contour locking, and frame localization. However, weather fluctuations and insufficient brightness are the main problems with camera-based methods [4]. Hungar et al. [39] used DAISY [191] descriptors to illustrate 2D reflection maps of aligned point clouds. The remaining maps were fed an intensity gradient in eight radial directions with Gaussian smoothing. This method compares the maps with 12 shapes. In other words, the shapes are considered relevant features. Some recent methods take advantage of recent advances in deep learning. DeepICP [186] was proposed as an approach for a new generation of registration methods based on end-to-end learning.

DeepICP solves the problems of ICP registration. BirdNet [63] collects important patterns from a bird's eye view, which are 2D images of projected LiDAR measurements. Transfer learning, such as the VGG16 architecture, is used for this purpose. Fast-Region-based Convolutional Neural Networks (Fast-RCNN) are also used for object detection. As an extension of this work, BirdNet+ [64] is proposed, which applies an end-to-end strategy directly to 3D point clouds instead of using projection pre-processing. Both methods have been tested on the Kitti object detection benchmark dataset and have shown good results. More information about the registration can be found in [123]. The authors of [210] proposed a method to solve localization and mapping based on the use of a clustering-modified particle filter that selects the best candidate positions using sigma point selection techniques that have the same concept as the unexposed Kalman filter. These points speed up the localization process and also improve accuracy to achieve excellent results. This method also uses non-semantic features to perform the measurement update step in the particle filter. Non-semantic methods are suitable for creating lightweight feature maps, i.e., feature maps can be created in less time because most of these methods do not use complicated calculation formulas. In addition, they can represent any object inside the environment or at least part of them, which makes them robust against any change in the environment.

5.3 Methodology

In this section, we present the proposed methodology for solving the problem of localization and mapping in GNSS deprived environment. In Figure 5.1, a general overview of the architecture of the proposed approach is shown, where an adaptive particle filter is employed to correct the error provided by the motion and measurement sensor (IMU and LiDAR, respectively) and find the best position estimates. Our method uses two kinds of information sources (inputs): LiDAR and IMU.

The issue is that the IMU unit is responsible for providing information about the vehicle's movement (translation and rotation). Based on that, the vehicle's position is estimated by dead reckoning (based on initial position information provided by the GPS). Mathematically, the dead reckoning calculation can be given by:

$$\nu_t = r \cdot \omega_t + r \cdot \epsilon_t \tag{5.3}$$

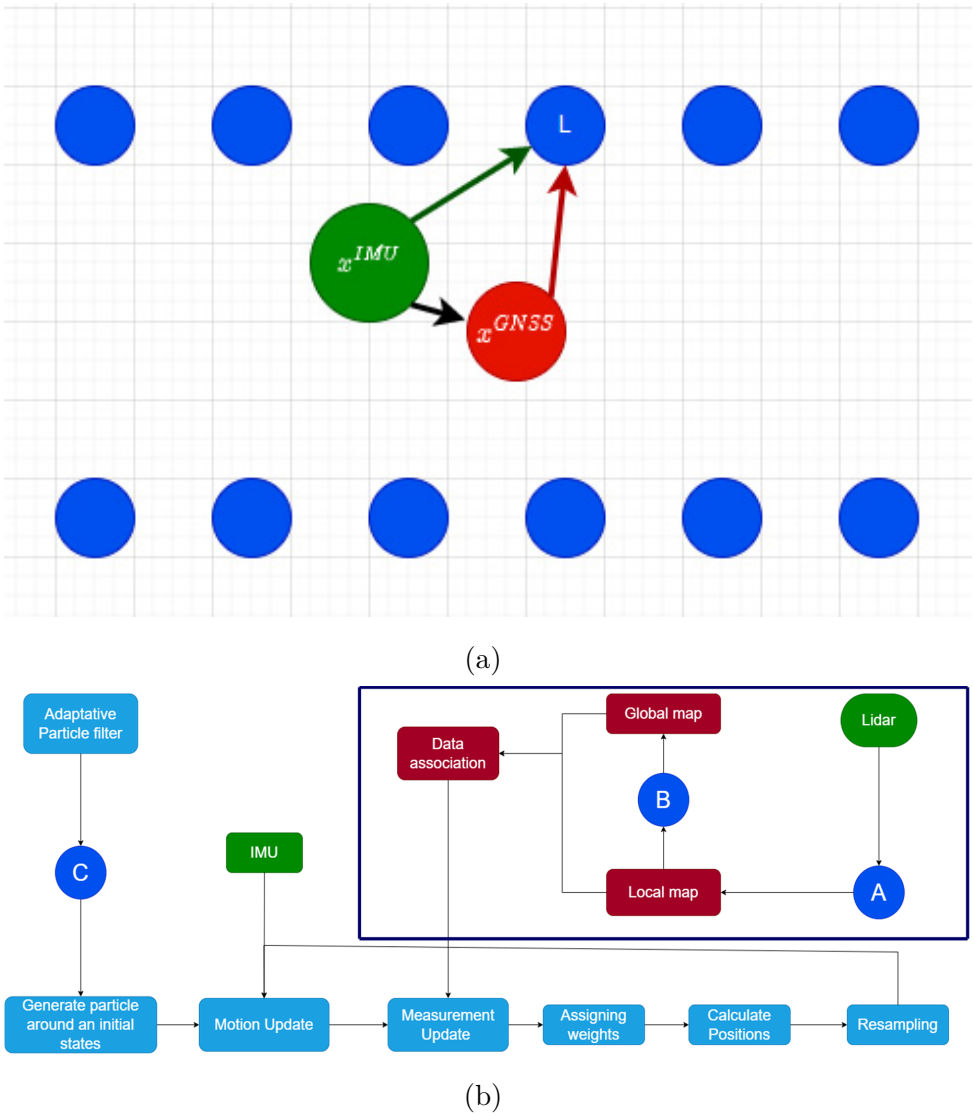


Figure 5.1: Workflow architecture describing our proposed method. **(a)** Describing the vehicle's true position (in red), IMU's position (in blue) and objects in the vehicle's environment as picked up by the LiDAR (in blue). **(b)** The structure of the proposed approach consists of two main steps: data processing features extraction (A) + global mapping (B), described within the dark blue rectangle, which is a framework for LiDAR measurements processing and creating local and global maps. Secondly, localization (C) using adaptive particle filters.

where ν_t is the velocity of the vehicle at time t, r is the radius of the tire, ω_t is the angular velocity at time t, and ϵ_t is the uncertainty estimation. To calculate the displacement of the vehicle, we need to integrate Equation 1. We get:

$$d = \int_{t-1}^{t} (r \cdot \omega)_t + C \tag{5.4}$$

where $C = \int_{t-1}^{t} \epsilon_t$ is an uncertainty of integration. Thus, we can calculate the vehicle's position by this formula: (Note that we will need an initial position from the GNSS)

$$(x_t)^{\text{IMU}} = (x_{t-1})^{\text{IMU}} + d = (x_{t-1})^{\text{IMU}} + \int_{t-1}^{t} (r \cdot \omega)_t + C \tag{5.5}$$

However, the noise present within the IMU's measurements leads to an exponential error growth, which is cascaded over time during the continuous position estimation process (green dot in Figure 5.1(a)) which should be shifted to the real vehicle position (black arrow to red dot in Figure 5.1(a)). Mathematically, we need to minimize this error:

$$\text{Err} \approx (x_t)^{\text{IMU}} - (x_t)^{\text{GNSS}} \tag{5.6}$$

where $(x_t)^{\text{GNSS}}$ is the real position to estimate (red dot in Figure 5.1(a)) provided by the GNSS.

The approach to minimize this error: In order to overcome this issue, we proposed Adaptive Particle filter which is probabilistic method that takes as an input; the IMU's information $(x_t)^{IMU}$ (containing errors) and a second source of information; the LiDAR's measurement, which captures objects within the vehicle's environment and the distances of these objects from the vehicle in the form of 3D points. Using the information of the resolved global position of the object L as illustrated in Figure 1a and the distance between the object L and the vehicle as derived from the LiDAR's measurement, the IMU's positional resolving can be corrected.

Note that the points in LiDAR scans are static and need to need to be converted to the body reference system for use. At each timestamp t, the datasets provide a way to transform the LiDAR points to the body system i.e providing the rotation and translation in form of a homogeneous transformation matrix.

$$(Trans)^t = \begin{bmatrix} & R^t & & T^t \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix will have the form:

$$R^t = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And

$$T^t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Now, to transform the LiDAR scan at time t, we multiply each point from the scan by the homogeneous transformation matrix. We obtain:

$$P' = (Trans)^t * P$$

where P is a point from the scan at time t, $P = [p_1, p_2, p_3, 1]$, P' is the transformed points to the reference body system. We repeat the process for each point in the scan and also for all the scans captured by LiDAR measurement. Note that we have added a fourth coordinates just to make the multiplication possible. We can remove it directly after performing the multiplication and that will not affect the coordinates since the results of the fourth coordinates at all times will be equal to 1. Consequently, to reduce the dimensionality from 4 to 3 we need to divide the first three coordinates on the fourth coordinates but since the fourth coordinates equal to 1 will not change nothing.

Our architecture method consists of two main steps: data processing (the dark blue rectangle in Figure 5.1 and the localization step. Data processing (feature extraction + mapping) is responsible for extracting relevant features from LiDAR's measurement and creating local and global maps. In addition, it ensures the fluidity of the matching search between the local and global maps. In the localization phase, the estimation of the vehicle positions is performed using the APF extension. The features extracted, and the IMU information, will feed the input of the APF. Next, some mathematics operations and probabilistic concepts will be applied over the received input (explained in section 5.3.2). Finally, the method will provide the estimated positions.

5.3.1 Features Extraction and Local Maps Creation

The features extraction step, which corresponds to step (A) in Figure 5.1, was performed to treat the massive amount of LiDAR data points in order to speed up the localization

task and create a lightweight feature map. In this stage, each LiDAR scan, as shown in Figure 5.3, is processed as follows. Firstly, we removed the ground plan by excluding points on the z-axis of the scan less than 0.1 m, as shown in Figure 5.3, as the ground plan does not contain distinguishable features that could help in the matching process and adds to the computational complexity of the approach. Secondly, we applied a Fuzzy K-means clustering to the remaining points and extracted central clusters representing our local feature map, as shown in Figure 5.3. In contrast to semantic features, our features are capable of representing less textured environments, such as poles, trees, roads, and curves, without knowing which cluster corresponds to semantic shape features, e.g., poles or trees. We chose to use Fuzzy K-means due to its ability to obtain clusters of different shapes and sizes and reduce the information of a group of points to a central cluster, which is very important for fast interaction within the framework. Moreover, in Table 5.1, we demonstrate its speed by testing different clustering methods on sequence 0001 from the Kitti data set [187]. A flowchart of the following process is provided in Figure 5.4, underpinned by an illustration in Figure 5.3.

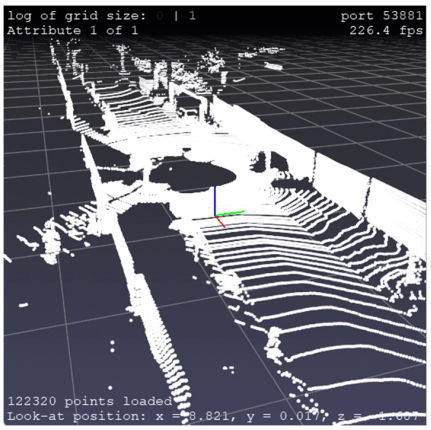
Table 5.1: Time consumed in executing the feature extraction workflow with the specified clustering methods tested on the sequence 0001 from the Kitti data set [187].

Clustering Method	Time Cost (mm:ss)
Growing neural gas	01:24
KMeans	01:38
Fuzzy K-means	00:43
Hierarchical clustering	01:07
Gaussian mixture model	00:49
Self-organizing maps	02:06
Agglomerative clustering	01:18
Particle swarm optimized clustering	00:52

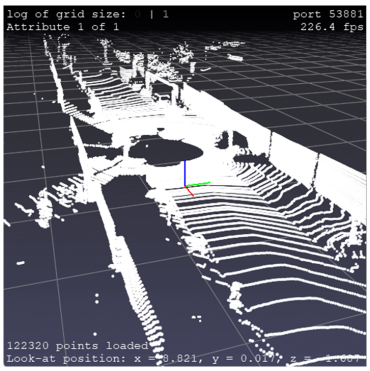
Thirdly, we transform all the scans into the vehicle reference coordinate system [31].

5.3.2 Localization

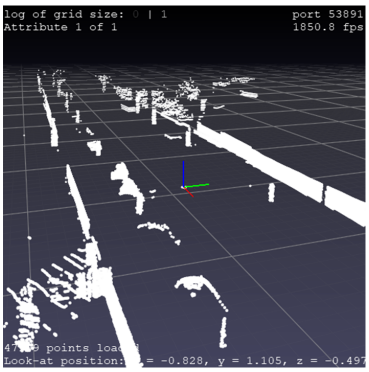
The last stage is the localization part (step (C) in Figure 5.1), where we explore the results of the pre-processing steps (features extraction and mapping) to correct the accumulated IMU errors based on an adaptative particle filter described below:



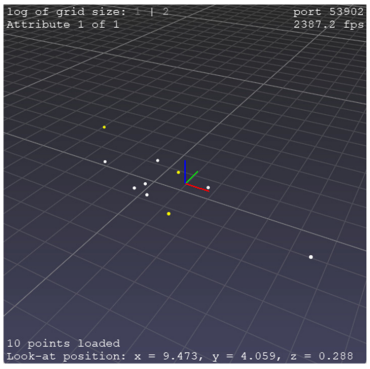
(a) 3D LiDAR points scan.



(b) 3D LiDAR points scan.



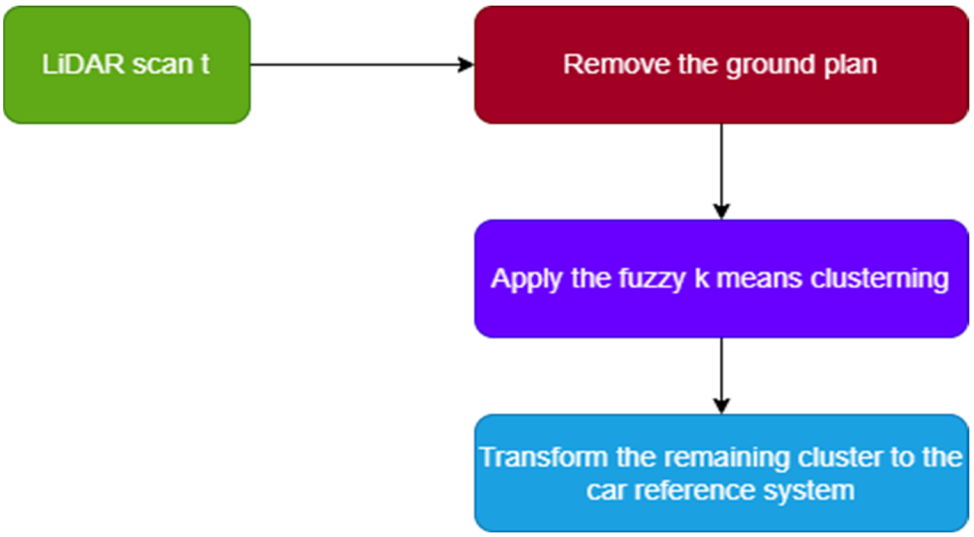
(c) Removing the ground plan Scene.



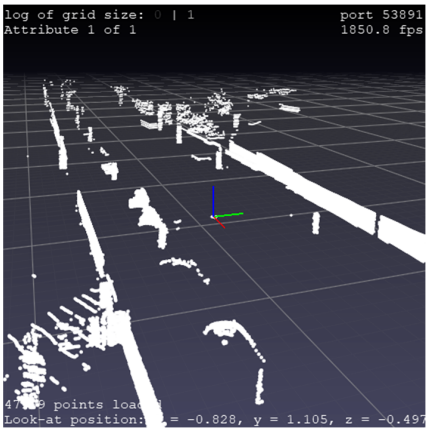
(d) Features extraction process (Zoomed in).

Figure 5.2: Illustration of the feature extraction.

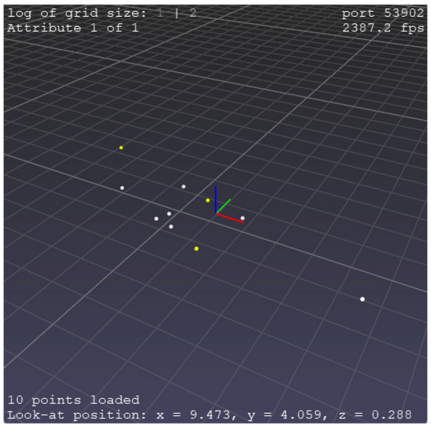
- **Particles generation**: The filter generates hundreds of particles around an initial state provided by the GNSS noted by $X = X_{(par_i)_{i\in[0,N]}}$ a set of particles where N is the number of particles (see Figure 5.5a). Each particle is represented as a 4×4 homogeneous coordinate matrix resulting after uniformly generating the x, y coordinates and θ orientation, which is given by:



(a) Flowchart of the feature extraction from LiDAR measurement.



(b) Removing the ground plan Scene.



(c) Features extraction process (Zoomed in).

Figure 5.3: Illustration of the feature extraction.

$$X_{(par_i)} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & x_i \\ \sin(\theta_i) & \cos(\theta_i) & 0 & y_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

This particular representation of the homogeneous coordinate matrix since there is no rotation about the x and y axes when the vehicle moves (i.e., the roll and pitch angle are neglected), and the value of the z coordinates is neglected because the vehicle is assumed to be driven on a 2D surface. The main advantage of this representation is the simplification of the matrix calculation (i.e., multiplication). Each particle represents a possible target state and is associated with a probability

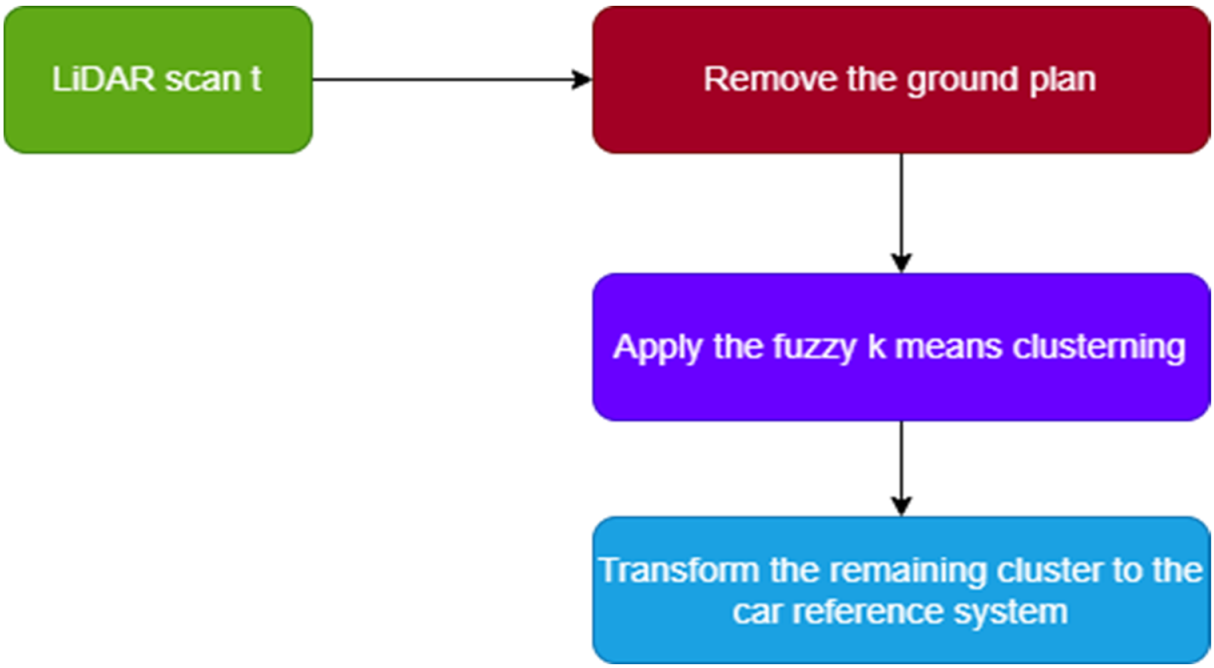


Figure 5.4: Flowchart of the feature extraction from LiDAR measurement.

value, called weight $\omega_{(par_i)}$, which is uniformly initialized. The largest of these values belongs to the particle corresponding to the true position of the target.

- **Motion update**: In the second step of the localization stage using IMU and PF, the motion update moves the particles from scan $t-1$ (or timestamp $t-1$) to scan t (or timestamp t) by using the homogeneous coordinate matrix properties, which perform the transformation with this formulation:

$$X^t = \{X^t_{(par_i)}\}, \quad X^t_{(par_i)} = G^t * X^{t-1}_{(par_i)} \tag{5.8}$$

G^t is a 4×4 homogeneous matrix of this form:

$$G^t = \begin{bmatrix} R^t & T^t \\ 0 & 1 \end{bmatrix} \tag{5.9}$$

R^t (3×3 matrix) and T^t (3×1 vector) are the rotation and translation matrices provided by the IMU sensor at scan t, respectively, as shown in Figure 5.5b.

- **Measurement update**: The third stage is the measurement update, which examines the information from the measurement sensors, especially the LiDAR sensors, since the information from the camera is sensitive to changes in light and weather conditions. The aim is to determine how similar the transformed (We calculate the

distances of each particle with all the features gathered in scan t) particle $X^t_{(Par)i}$ provided by the IMU at scan t, is to the real state, i.e., how much it contributes to the determination of the real state. In order to find that, a weight (or percentage) $\omega_{(par)_i}$ is assigned to each particle and updated at each scan, which gives the probability of the particle's contribution along the trajectory, as shown in Equation 5.12. According to [31], these weights can be updated by:

$$w^t_{(Par)i} = \prod p \in [0,t] P(F^p | X^p_{(Par)i}, Mn(p)) \tag{5.10}$$

where

$$P(F^p | X^p_{(Par)i}, Mn(p)) := N(||X^p_{(Par)i} F^p - Mn(p)||, \sigma) \tag{5.11}$$

The product of X^p and F^p gives the distances between the particles and features at scan p. σ is an isotropic position uncertainty depending on the reference of features. $n(p)$ is an index of the global features associated with the p_{th} local feature (provided by kd-tree), which means that $M_{n(p)}$ is the corresponding of F^p in the global map M. The idea behind this weight modification is that we can compare the normal curve of distance particle-to-feature (The transformed particles provided by the IMU at scan t) and what we have on the real map (global map). The closer the distributions, the higher the value the weights are (see Figure 5.5c).

The state estimation at scan t can be calculated by:

$$pos^t \approx \sum_{i=1}^{N} w^t_{(Par)i} X^t_{(Par)_i} \tag{5.12}$$

where

$$\sum_{i=1}^{N} w^t_{(Par)_i} = 1 \tag{5.13}$$

and N is the number of particles.

- **Resampling**: To obtain a fast and accurate method, a resampling step was carried out to reduce the uncertainty of the particles and to adjust their distribution. By checking the corresponding weights, the particles were ranked from important to unimportant. The particles were resampled, focusing on the 10 most important particles, which enabled the particles to track the real position at each scan and

obtain intuitions about the direction of the positions. We calculated the mean and covariance of the selected particles. Then, we regenerated the particles based on the multinormal distribution (see Figure 5.5d).

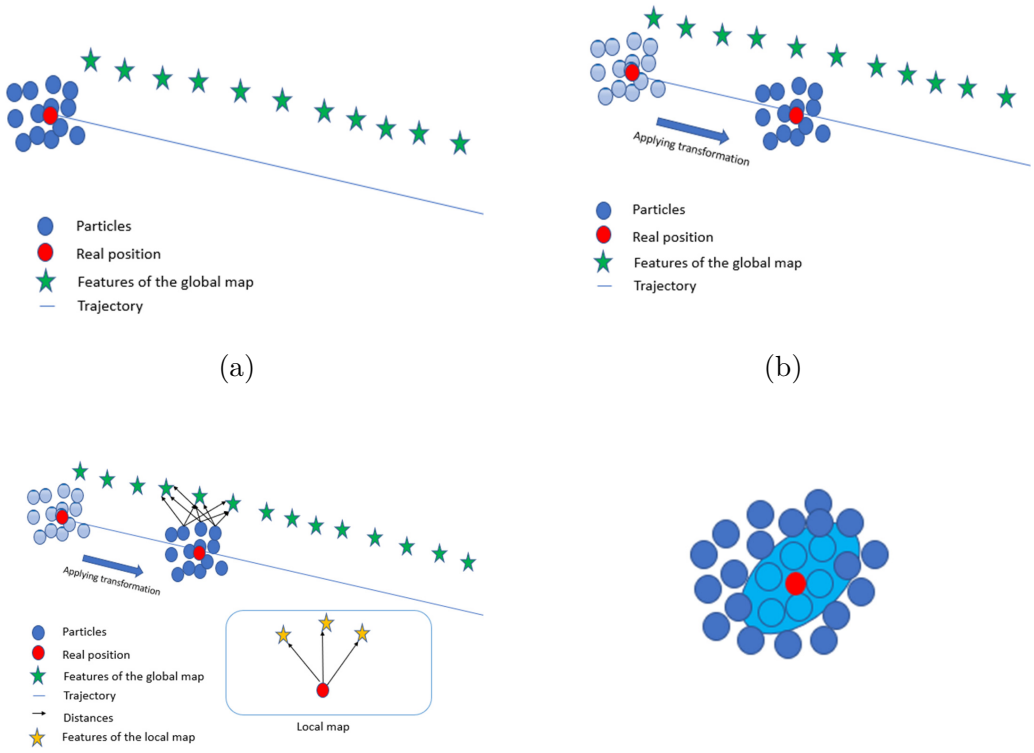


Figure 5.5: Explanation of the workflow of the particle filter (the images are not real; they are generated to explain the process in depth). (a) Generate particles. (b) Applying IMU transformation. (c) Compare the distances of the particles with the futures of the global map (features in green) and the real distance. (d) Resampling stage, where we regenerate particle around the closest one (particle in-cyan color)

- **Evaluation**: Our metric evaluation included six error measurements Δ_{pos}, Δ_{lat}, Δ_{lon}, and Δ_{ang}, denoting the mean absolute positional, latitudinal, longitudinal, and heading errors, respectively, while $RMSE_{pos}$ and $RMSE_{ang}$ represent the corresponding Root Mean Squared Errors of the position and heading estimation.

$$\Delta_{lat} = \frac{\sum_{i=1}^{n} |x_i - x'_i|}{n} \qquad \Delta_{lon} = \frac{\sum_{i=1}^{n} |y_i - y'_i|}{n}$$
$$\Delta_{pos} = \frac{\sum_{i=1}^{n} |pos_i - pos'_i|\, 2}{n} \qquad \Delta ang = \frac{\sum_{i=1}^{n} |\theta_i - \theta'_i|}{n}$$
$$RMSE_{pos} = \sqrt{\frac{\sum_{i=1}^{n} |pos_i - pos'_i|\, 2^2}{n}} \qquad RMSEang = \sqrt{\frac{\sum_{i=1}^{n} |\theta_i - \theta'_i|^2}{n}}$$

where n is the scan number, $pos_i = [x_i, y_i]$ is the predicted state, $pos'_i = [x'_i, y'_i]$ is the actual state, x_i and x'_i are the predicted and actual lateral, y_i and y'_i are the predicted and actual longitudinal. θ_i is the predicted angle, and θ'_i is the actual angle.

We used the longitudinal and latitudinal mean absolute error Δ_{lat} and Δ_{lon}, respectively, to examine the causes of error fluctuation according to the x-axis and y-axis, such as hard braking, vehicle rotation, or vehicle acceleration. We used the mean absolute position and root mean square positioning error Δ_{pos}, $RMSE_{pos}$ to study the overall localization error and the effect of the vehicle rotation error Δ_{ang}, $RMSE_{ang}$ on that error.

5.4 Results

5.4.1 Experiments

A computer with an 8th generation i7 processor and 20 GB of RAM was used in our experiments. We chose the Kitti dataset to test the proposed method on several sequences characterizing urban, residential, campus, and pedestrian traffic. The advantage of using this dataset is that it provides measurement information from a variety of sensors, including LiDAR point clouds, stereo or mono camera images, IMU inertial measurements, and high- accuracy GNSS data. Kitti also enables the evaluation of proposed approaches in various environments. We tested our method on a second dataset, the Panda dataset [195], which contains different driving scenarios, including steep slopes, construction sites, heavy traffic, and pedestrians, as well as a variety of weather and lighting conditions, such as in the morning, afternoon, evening and night. The dataset is equipped with various sensors, such as IMU/GPS information and Lidar, for camera measurements. A 10-core feature extraction method in the K-means algorithm was used to create a global map (or

a ground plan) based on a high-accuracy navigation system. Two criteria were considered in selecting the number of clusters: speed and representativeness. Smaller clusters are faster but less representative, and vice versa. Each local map was matched with a global map, and the localization process was started using particle filters. The number of particles was around 100 points. The initialization region was also in the range $[0, 1]$. The output of the localization algorithm is a 4Ö4 information (transformation) matrix containing the matrices of rotation and translation performed between two consecutive keyframes. The method was tested on the entire class of Kitti datasets, which contained a total of 987 scans from four sequences. The codes used in this experiment can be found in [212].

For the purpose of this experiment, the global maps were created priorly using the LiDAR scans. After creating the local maps and converting them into a reference system., we merge them in an order where each time frame corresponds to a single local map, which creates a global map of the trajectory. Creating a global map (step (B) in Figure 5.1) is an essential step before executing any localization method. As mentioned before, it acts as a guide for the vehicle, i.e., it provides distances to the objects. This kind of map is a low-content information one, meaning that it represents objects (or part of them) by providing only their coordinates (see the illustration in Figure 5.6b). The creation of the global map with this technique speeds up the process of data association. Data association is the task of finding a similar part of the local map in the global map by using the kd-tree searching algorithm. Example of this process can be seen in Figure 5.6b.

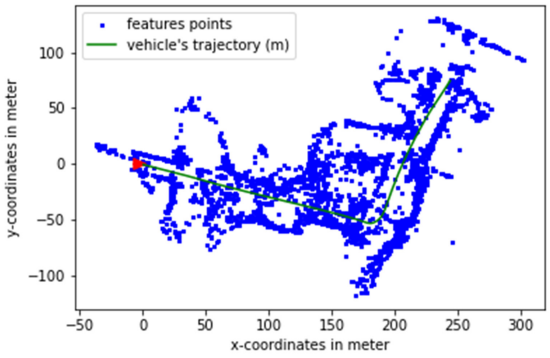
Let us note:

$$k \in [0, M_f], M = m_k \quad (3) \tag{5.14}$$

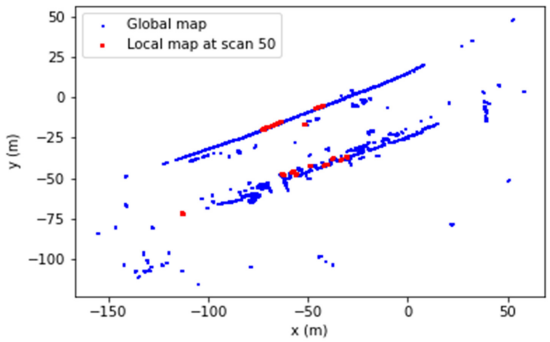
where M_f is the number of feature points in the global map, m_k is the feature point at position k, and M is the list of features in the global map. We also define:

$$k \in [0, N_f], F^t = f_k^t \quad (4) \tag{5.15}$$

with N_f the number of feature points in the LiDAR scan t, f_k^t the feature point at position k in the scan t, and F^t the list of features at scan t. It is important to note that the use of LiDAR scans in our experiments doesn't represent map matching in real-world scenarios.



(a)



(b)

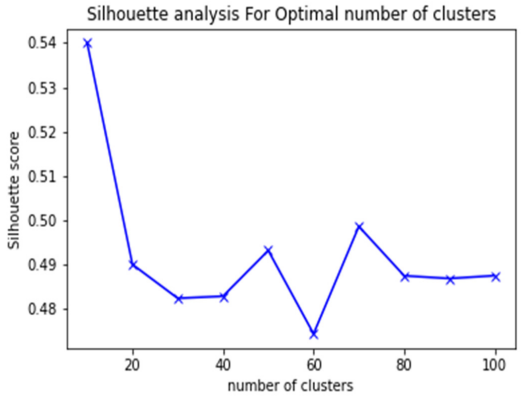
Figure 5.6: Explanation of the workflow of the particle. Figure (a) shows the results of the mapping phase after concatenation and transformation, where the blue dots represent the global features map, the red triangle is the particle initialization, and the green line is the ground truth. Figure (b) shows an example of data association of scan 50 of the sequence 0001 of the Kitti dataset.

5.5 Discussion

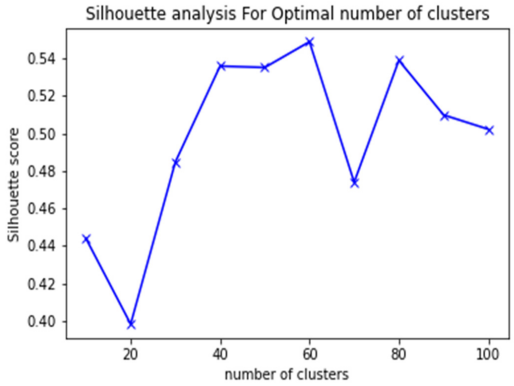
5.5.1 Parameters discussion:

We have investigated the parameters of a configuration of our features extraction framework in terms of fastness and accuracy in order to find the optimal parameters that give the best results. The most important parameter that should be investigated in the features extraction workflow is the number of clusters in the fuzzy K-means algorithm. We calculated the silhouette values for different numbers of clusters in different sequences of the Kitti dataset. We observe in Figure 5.7 that the average silhouette value for all

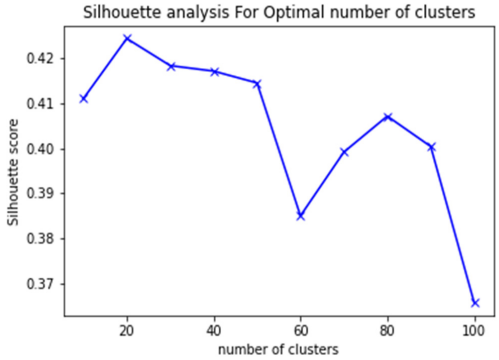
sequences is 0.44 with a cluster number of 10, 0.43 with a cluster number of 20, 0.46 with a cluster number of 30, 0.47 with a cluster number of 40, 0.46 with a cluster number of 50, 0.46 with a cluster number of 60, 0.45 with a cluster number of 70, 0.47 with a cluster number of 80, 0.45 with a cluster number of 90, and 0.44 with a cluster number of 100. We saw that the silhouette value of all scenarios is in the range of 0.43–0.47, which justifies the choice of a 10-core clustering center, as it consumes less time and regularly has a good silhouette value.



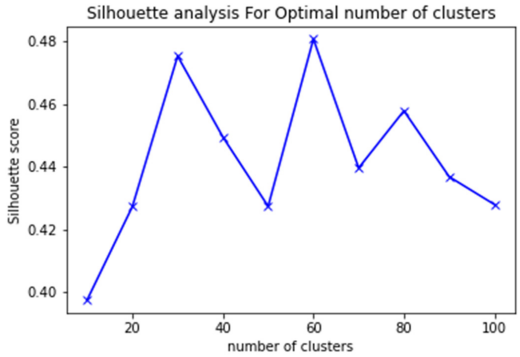
(a) Silhouette values in sequence 0001.



(b) Silhouette values in sequence 0034.



(c) Silhouette values in sequence 0027.



(d) Silhouette values in sequence 0053.

Figure 5.7: Silhouette values registered by fuzzy K-means algorithm for different cluster numbers in sequences 0001, 0034, 0027, and 0053 from the Kitti dataset. (a) Silhouette values in sequence 0001. (b) Silhouette values in sequence 0034. (c) Silhouette values in sequence 0027. (d) Silhouette values in sequence 0053.

Our feature extraction process, which is illustrated in Figure 5.3, effectively reduces the time and cost of calculation, where each scan of size 1.8 MB (on average) was processed in 0.4 s (on average) and reduced into 790 KB (on average), which demonstrates the lightweight of our global and local maps. Moreover, the process helps the data association stage to be fast, facilitating investigations on similarity issues.

5.5.2 Discussion on Kitti and Pandaset accuracies:

The results in Table 5.2 show that the method can reliably locate the vehicle in various sequences and categories, such as weather and seasonal changes. In addition, the method takes 7 s to locate the vehicle within 5 sequences and 544 frames, which is fast enough to operate in a real-time scenario; the IMU sensor takes 52 s to record information from 544 frames, and our method quickly processes the incoming data with an execution time of 7 s. The manual testing of the Schaeffer et al. [31] method on the sequence 0001 (City category) of the Kitti dataset showed an absolute position error of 0.06 m and an absolute rotation error of $(0.1)^{\check{r}}$ in 19 s. In our case, we obtained 0.07 m for the absolute position error and $(0.013)^{\check{r}}$ for the rotation error in a run time of 6 s, as presented in Table 5.2, which demonstrates the fastness of our method. According to Table 5.2, we registered 0.25 m of the average positioning error of all the sequences in only 7 s of execution, which demonstrate the capability of our method to localize the vehicle with a minimum of 100 particles. The obtained results are due to the robust resampling technique that followed accurately the trajectory.

Table 5.2: Kitti localization error obtained in different categories and the time consumption in each sequence. tFeat is the cost of time to calculate the features extraction. tloc is the cost of time to calculate the localization.

Category	Seq	frames no	tFeat (s)	tloc (s)	Δpos (m)	Δlat (m)	Δlon (m)	Δang (°)	RMSEpos (m)	RMSEang (°)
City	0001	108	00:34	00:01	0.07	0.01	0.06	0.012	0.08	0.014
Residence	0035	131	00:43	00:02	0.038	0.019	0.02	0.09	0.053	0.18
Road	0027	188	01:08	00:02	0.15	0.03	0.14	0.01	0.17	0.006
Campus	0034	49	00:16	00:01	0.0009	0.0006	0.0005	0.0014	0.001	0.001
Person	0053	68	00:23	00:01	0.0001	0.0001	0.0001	0.001	0.0007	0.0002

In another part, we tested the accuracy of our method against four methods that we found relevant in our state-of-art survey: Kümmerle et al. [27], Weng et al. [29], Sefati et al. [26], and Schaefer et al. [31]. All of them have used semantic features to represent LiDAR scans, such as poles, walls, trees, etc. The comparison of Sefati et al. [26] and Schaefer et al. [31] approaches with our proposed method was feasible, as they used the same error metrics, and their implementation was provided in their paper. However, it was hard to compare our method with Weng et al. [29] and Kümmerle et al. [27] method unless qualitatively, as it was provided by Schaefer et al. [31] in the article. Table 5.3 shows that our method can produce competitive results. In fact, our method outperformed all others in terms of mean absolute angular error and mean square angular

error. In addition, we obtained competitive results for mean absolute positioning error and mean squared positioning error, which outperformed all compared methods except Schaefer et al. [31] (with our results 0.09 m and 0.11 m, respectively) and Schaefer et al. [31] (0.11 m and 0.12 m, respectively, for our method).

These results are justified by the robust data association in the work of Schaefer et al. [31]. However, working with semantic features can affect the localization process, especially when these features are not present in the environment.

We extensively tested our method on a second dataset, Pandaset, on 5 sequences containing 80 frames each. Our method recorded an average absolute positioning error of 0.16 m in 5 s runtime, demonstrating the performance of our method when dealing with different environmental scenarios (see Table 5.4).

Table 5.3: Performance comparison of the proposed technique. The results of Weng et al. [29] and Kümmerle et al. [27] are not straightforwardly similar and are expressed for qualitative analysis only.

Methods	Δpos (m)	RMSEpos (m)	Δlat (m)	σ_{lat} (m)	Δlon (m)	σ_{lon} (m)	Δang (°)	σ_{ang} (°)	RMSEang (°)
Kümmerle et al. [27]	0.12	—	0.07	—	0.08	—	0.33	—	—
Weng et al. [29]	—	—	—	0.082	—	0.164	—	0.329	—
Sefati et al. [26]	—	0.24	—	—	—	—	—	—	0.68
A. Schaefer et al. [31]	0.096	0.111	0.061	0.075	0.060	0.067	0.133	0.188	0.214
Charroud. A et al. [210]	0.12	0.141	0.059	0.09	0.08	0.05	0.043	0.078	0.057
Ours	0.101	0.12	0.064	0.035	0.06	0.087	0.043	0.075	0.075

We studied in depth the results of sequence 0009. In Figure 5.8a, 5.8c, we mapped the position and angular error variations in each scan, allowing us to make some relevant observations. We found that acceleration mainly affects the accuracy of positioning. From scan 0 to 20, the period in which the vehicle takes off, and from scan 110 to 170, the period in which the vehicle slows down when taking a turn on the road (Figure 5.8c). In both periods, the error varies from 0 m to 0.25 m and back to 0 m, which means that the particles need time to adapt to the new acceleration of the vehicle.

The same is observed in the case of drift or hard braking, like in Figure 5.8c, 5.8d and Figure 5.9a, 5.9b sequences 0035 and 0027. Therefore, a good positioning model should take into account the velocity to anticipate the vehicle's motion. Furthermore, we observe that the positioning error is systematically 0.1 m from scan 170 to 250 when the vehicle is turning. However, the angular error recorded high values from 0 to 0.4, indicating the need to determine the turning intensity. Additionally, the same issue was registered in

Table 5.4: Accuracy evaluation of our method on the Pandaset dataset.

Seq	Frames No.	t_{Feat} (s)	t_{loc} (s)	Δ_{pos} (m)	Δ_{lat} (m)	Δ_{lon} (m)	RMSE$_{pos}$ (m)
100	80	00:01	00:01	0.18	0.07	0.16	0.21
109	80	00:01	00:01	0.22	0.07	0.20	0.23
117	80	00:03	00:01	0.16	0.06	0.13	0.19
139	80	00:01	00:01	0.22	0.03	0.21	0.29
158	80	00:01	00:01	0.05	0.03	0.03	0.06

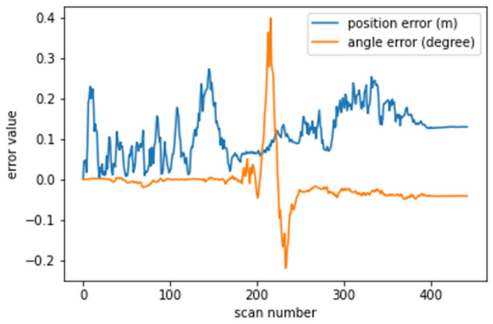
sequence 0034 in Figure 5.8 and sequence 0053 in Figure 5.9.

As research in the field of localization and mapping shifts from observational studies to direct applications in real-world scenarios, the need for methods that speed up the localization process, such as the method we propose in this paper, is increasing. However, there are some drawbacks to the use of particle filtering algorithms, particularly in setting the particle number and threshold identification.

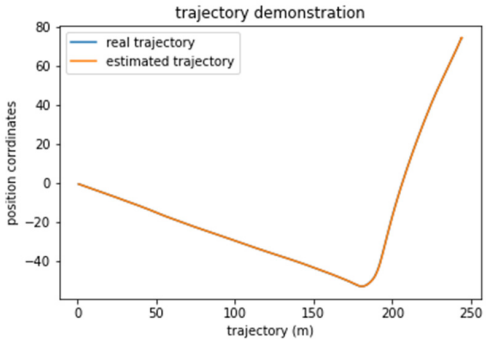
5.6 Conclusions and Future Work

This paper presents a method for autonomous vehicle positioning and mapping based on non-semantic feature extraction. A fuzzy K-means clustering was used to extract features from LiDAR scans. The cluster centroids features were used to create local map features. Furthermore, an adaptive particle filter was used in the localization process, which included a resampling stage after finishing the measurement updates in order to increase the reliability of the position estimation and reduce the time and energy cost. The resampling method selects the closest 10 particles to the real position by checking their weights (percentage) and regenerating particles around them using a multinormal distribution.

The proposed method provides competitive accuracy results in significantly less time compared to the state-of-the-art methods evaluated on the Kitti database. We obtained an error of 0.25 m in the mean position error of all sequences in the Kitti dataset in a run time of 7 s. Furthermore, we obtained 0.15 m in the mean position error of all sequences in the Pandaset dataset in only 5 s of execution, which demonstrates the potential of our resampling contribution to speed up the localization process and obtain state-of-the-art results. The proposed method provides competitive accuracy results in a significantly



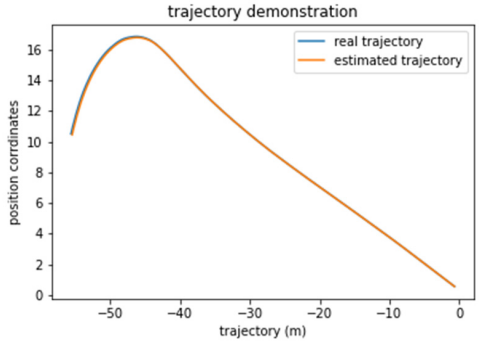
(a) positional and angular error in seq 0009.



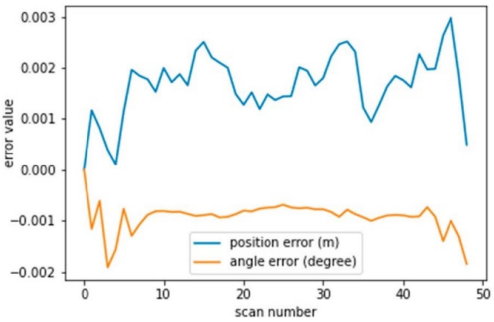
(b) vehicle trajectory in seq 0009.



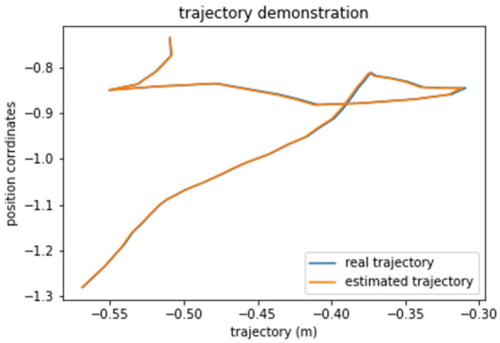
(c) positional and angular error in seq 0035.



(d) vehicle trajectory in seq 0035.



(e) positional and angular error in seq 0034.

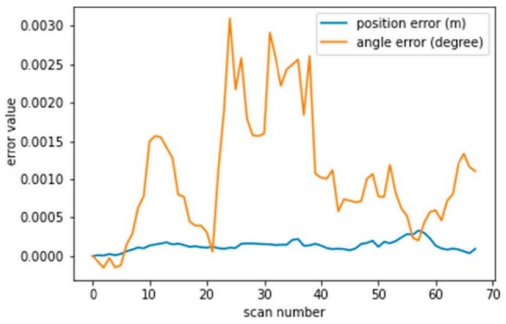


(f) vehicle trajectory in seq 0034.

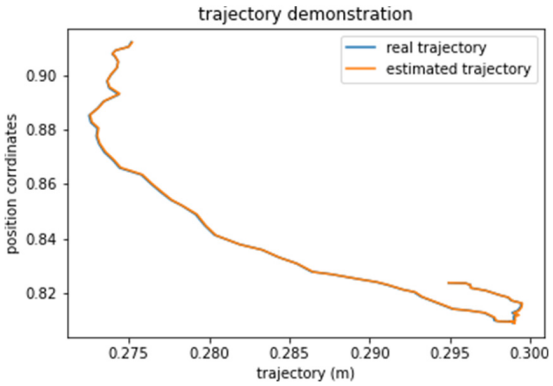
Figure 5.8: Graphical display of the average absolute position and angular error relative to the vehicle path for each sequence.

shorter time compared to state-of-the-art methods that have been evaluated on the Kitti dataset.

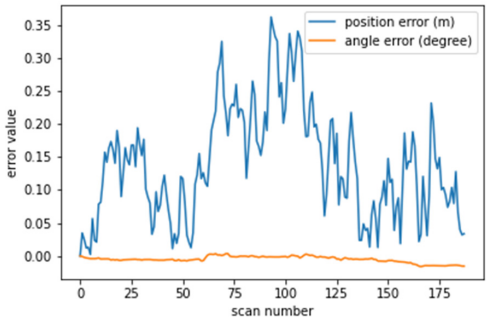
However, the accuracy of the proposed method sometimes deviated due to poor initialization of particle filters or poor selection of clusters or number of particles. Therefore, future research will involve finding the optimal initialization of the particle filter and finding the best parameters configuration. The particle filter suffers from the random initialization of the particles, which could be modelled by a stochastic differential equation



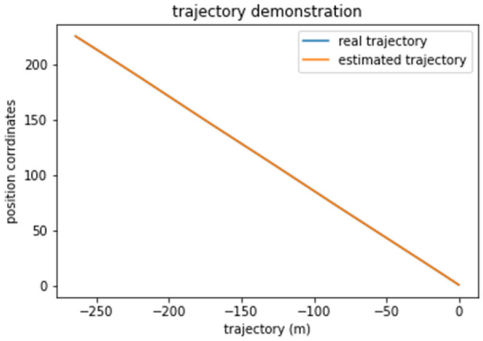
(a) positional and angular error in seq 0053.



(b) vehicle trajectory in seq 0053.



(c) positional and angular error in seq 0027.



(d) vehicle trajectory in seq 0027.

Figure 5.9: Graphical display of the average absolute position and angular error relative to the vehicle path for each sequence.

to control particle generation. On the other hand, the use of other clustering methods opens up a new approach to the representation of LiDAR measurements, which can provide insights into the use of other machine learning techniques, such as object tracking with deep learning (or transfer learning), to ensure better matching of data association. Thus, the position of the vehicle could be more accurately estimated.

Part III

vehicle localization: A deep learning approach

CHAPTER 6

XDLL: EXPLAINED DEEP LEARNING LIDAR-BASED LOCALIZATION AND MAPPING METHOD FOR SELF-DRIVING VEHICLES

6.1 Introduction

A self-driving vehicle requires more attention to be paid to the execution time of their components, since this may save many lives every day [4]. It must be ready to perform the right action in real time to prevent incidents. In order to achieve this level of control, the system should be built to provide rapid interaction between its components, to receive instant location information to perform the overtaking of vehicles and perform rapid path planning [18]. Localization of the vehicles is an essential step to ensure the functionality of other components. The information provided can be used to identify the distances between the vehicles and the structures in the environment, which is crucial information needed to avoid collisions, find the shortest distance to the destination, and so on [18]. Global navigation satellite systems (GNSS) are already equipped, in almost all autonomous vehicles, to provide position coordinates by triangulation using at least three satellites. However, this technology faces several issues for real-time execution when the satellite signals are interrupted, such as under bridges, in tunnels, and around tall buildings [174, 175]. Several researchers have thought to localize the vehicle based on internal sensors, such as light detection and ranging (LiDAR), cameras, inertial measurement units (IMUs), or even radar sensors [18].

Our review of the state-of-art in this area divides the related methods into two categories: direct methods and feature-based methods. Direct methods try to estimate vehicle's position directly by calculating the distances between two positions and using dead reckoning (DR) to deduce the position. The IMU unit from the inertial navigation system (INS) is one of the methods that uses this approach [176]. An IMU provides information about the acceleration and attitude rate of the vehicle. The double integration of the acceleration measurement will provide the position of the vehicle. However, the double integration is prone to errors that can be accumulated during the execution of the process [176]. Several researchers have tried to correct these errors by sending them to a machine learning model, such as an input delay neural network (IDNN) [104], multi-layer feed forward neural network (MFNN) [213], recurrent neural network (RNN) [106], or long short-term memory (LSTM) model [179, 177]. For more information about deep learning (DL) methods in self-driving vehicles, the readers are referred to [214].

Wheel odometry is a great tool for estimating the vehicle's position that works by using the velocity and integrating it to get the position [108]. However, as we mentioned before, the integration process provides an error that can be amplified and accumulated each time we re-execute the process. According to the article [108], using wheel odometry is better than using an IMU accelerometer, since it needs less integration time. The wheel odometry neural network (WhONet) [108] uses an RNN-based architecture to correct the wheel odometry errors, and an extensive experiment was performed during several GNSS outages.

Feature-based approaches for vehicle localization try to extract relevant features from data gathered from one or more sensors. We have surveyed in this work only LiDAR-based methods, since we believe that the camera images are easily affected by weather changes such as snow and rain [4, 18]. Extracting features from LiDAR measurements has been the subject of several papers in the literature. LiDAR odometry and mapping (LOAM) [200] distinguish edges and flat plan features to perform scan-to-scan and scan-to-map matching, which are techniques to track the vehicle's motions (translation and rotation) between consecutive sequences. However, the process consumes too much time when executed, which is covered in the article introducing the LOAM_Velodyne [201] method. To solve the same issue, lightweight and ground-optimized LOAM (LeGO-LOAM) [202] and A-LOAM [203] approaches remove noisy and useless features to reduce the computational time. Other methods, such LO-Net [204] and LO-Net-M [204], use end-to-end deep learning (DL) to improve the scan-to-matching process. SGLO [205] extracts geometric line

and plane features to improve the matching process, which has provided good results. However, the accuracy is dramatically affected by the initialization. Methods such as those of Kummerle et al. [27], Weng et al. [29], Sefati et al. [26], and A. Schaefer et al. [31] use a probabilistic perspective to localize the vehicles. Based on detecting poles and walls from LiDAR scans, the methods used a particle filter algorithm to correct the IMU's accumulative errors. The method of Schaefer et al. [31] achieved excellent results on the Kitti dataset [187] and the University of Michigan North Campus Long-Term Vision and LiDAR Dataset (NCLT) [190]. However, the features extracted for the operation may not exist in some environments, especially those with little in the way of texture, such as desert roads. That is why the work of Charroud et al. [43] proposed using non-semantic features to help perform the measurement-update step in the particle filter. An extension of this work was presented in the article [45], where the authors proposed a modified clustering particle filter that selects relevant particles to calculate the position by using sigma-point selection. Moreover, another extension of the work in [43] is the article [44], where it was proposed to extend the work on particle filters by selecting only the 10 best particles around the real position and regenerating the particles around them. This trick enables the particle filter to run fast and preserves the accuracy, as we generate particles close to the real position at each execution of the algorithm.

Our novel method uses non-semantic features as a pre-processing step to teach a machine-learning model the actual vehicle positions. The inputs are the extracted features, and the outputs are the positions. The model was extensively tested on short- and long-term trajectories using two benchmark datasets: Kitti [187] and NCLT [190]. In addition, the model is explained by studying the features that contribute positively or negatively to the output model. Based on the results of explaining the first model, another model was constructed and compared to evaluate the influence of the changes made. The article [215] discusses the concept of explainable artificial intelligence (XAI), which refers to the development of AI systems that are able to provide clear, interpretable explanations for their decisions and actions. The authors provide an overview of the various types of XAI approaches that have been proposed, including post hoc explanation methods, integrated explanation methods, and interactive explanation methods. They also explore the potential benefits and challenges of XAI, and discuss the importance of responsible AI development in ensuring the transparency, accountability, and trustworthiness of AI systems.

This paper contains the following contributions:

- As far as the authors are aware, this is the first method that uses only a few LiDAR cluster points to feed a deep-learning model for localization purposes.
- The paper presents, in Section 6.2, a robust mathematical formulation of the localization problem, which opens up some opportunities to develop more solutions based on optimization and stochastic-differential-equation-based methods.
- Deep Learning explanation methods were employed to find the most contributing cluster features in order to optimize the proposed model.

The sections of the paper are organized as follows. Section 6.2 provides an in-depth explanation of the problem and the architecture we propose to solve it. Section 6.3 presents the steps followed to create the proposed model. Section 6.4 presents the results of testing the model in the short and long-term scenarios. The contributions of the features are also discussed, and a comparison between the two models with further implications is presented in this section. Section 6.5 provides some conclusions.

6.2 Problem Description and Proposed Architecture

As mentioned before, any autonomous system requires a localization and mapping method to facilitate the scheduling of other principal tasks, such as path planning and overtaking vehicles. [4]. In particular, self-driving vehicles must be given more attention regarding the execution times of different components, since they need an instant interaction with the environment, which means that we need to ensure fast execution of the methods without hurting the accuracy too much. Quickness and accuracy are critical to getting a better experience from driving. Most of the state-of-the-art methods divide the process of localization and mapping into two parts: feature extraction and localization prediction. Extracting relevant features is a principal task in creating a global map, and helps to execute the localization task easily by providing the environmental objects or shapes around the vehicles. Figure 6.1 presents a simple workflow architecture that consists of two major steps: feature extraction (A) and learning the positions (B). In this section, we want to give the theoretical intuitions about the approaches followed to solve the localization and mapping problem for self-driving vehicles, which could be extended to other autonomous systems. A practical implementation of the method is delivered in the next section.

Our method localizes the vehicle based on LiDAR measurements only. We suppose that the LiDAR scans and the ground truth positions are in the body-frame coordinates, where at each timestamp each position has its corresponding scan. We use non-semantic features to distinguish some relevant prototype features. This technique can represent a wide range of objects inside the environment, in contrast to focusing on a single object (such as poles or edges), which is very advantageous to representing environments with less texture information. We chose to employ a clustering algorithm to extract clusters, which form our non-semantic features.

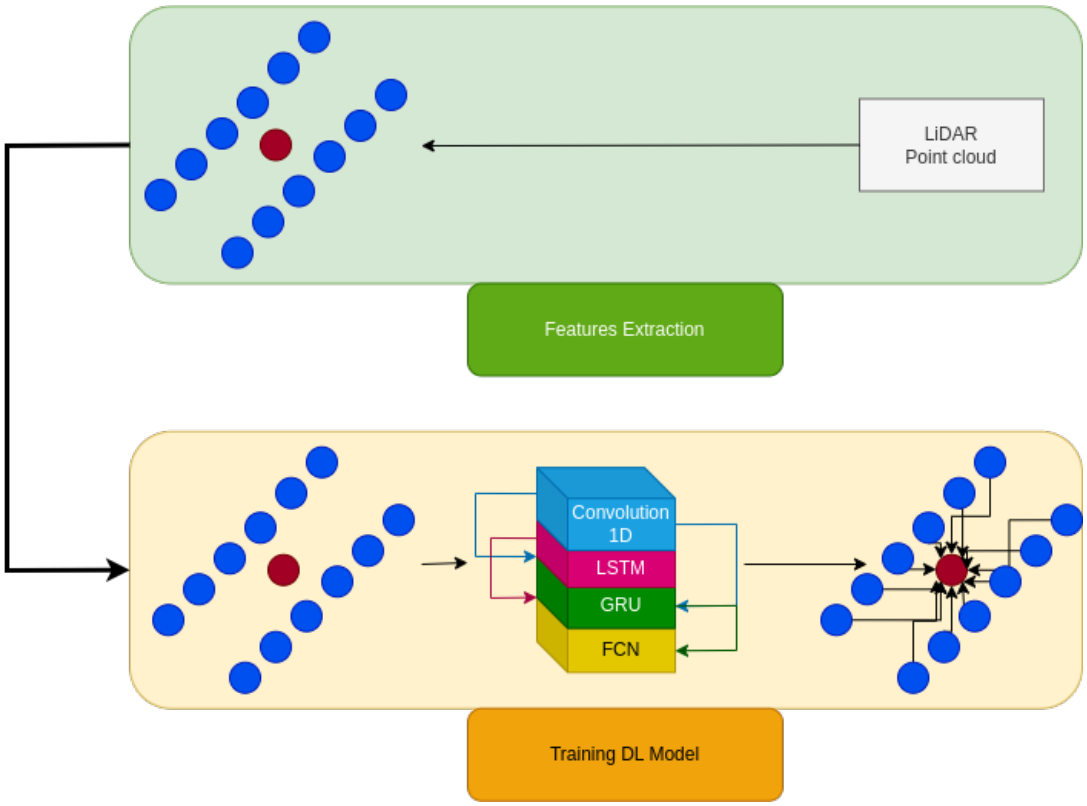


Figure 6.1: Representation of the workflow architecture. Feature extraction from LiDAR scans (blue dots) and training the DL model using convolutions of a 1D layer, an LSTM layer, a GRU layer, and a fully connected network layer to estimate the real position (brown dot).

Let us have some mathematical formulation:

Let P_t be a LiDAR scan at time t:

$$P_t = \{p_{t,i}\}_{i\in[1,D_t]} \quad \text{where} \quad p_{t,i} = [x_i, y_i, z_i] \tag{6.1}$$

where D_t is the number of LiDAR points at time t, and let also $s_t = [s_{t,i}, s_{t,j}]$ be the real position at time t (the z coordinate is supposed to be the same for the real position). We

apply a clustering technique by fixing the number of centers to a certain value (discussed in the next section), so we have:

$$C_t = \{c_{t,i}\}_{i\in[1,D'_t]} \quad \text{where} \quad c_{t,i} = [x'_{t,i}, y'_{t,i}, z'_{t,i}] \tag{6.2}$$

and D'_t is the number of clusters, C_t is the group of center clusters, and $c_{t,i}$ is the center of cluster i, where $i \in [i, D'_t]$. For simplicity, below we consider that the number of clusters stays the same along the trajectory ($D' = D'_t$ for all t).

The main idea of this paper is to find $\{\alpha_{t,1}, \alpha_{t,2}, ..., \alpha_{t,D'}\}$ and a function f that will minimize the problem (p):

$$(p) : \begin{cases} \text{Min}(\sum_{t=1}^{T} \sum_{i=1}^{D'} \|\alpha_{t,i} f(w_{t,i}, c_{t,i}) - s_t)\|_2^2) \\ w_{t,1}, \ldots, w_{t,D'} \in R^+ \\ \alpha_{t,1}, \ldots, \alpha_{t,D'} \in R^+ \end{cases} \tag{6.3}$$

T is the last timestamp value. We try to estimate a function f that will aggregate and manipulate the centers' clusters $C_t = \{c_{t,i}\}_{i\in[1,D']}$ in order to get an estimated position that will be close to the real position s_t. $w_{t,i}$ are some weights to ensure the reliability of the projection of the cluster from the 3D to the 2D environment. Scalars $\{\alpha_{t,i}\}$ are used to provide a percentage (probability) of the importance of the clusters $\{c_{t,i}\}$ after manipulation with the function f.

To solve the problem (p), we established a method based on a deep learning approach that takes the cluster features as the input and returns back the estimated positions. The architecture was chosen manually by testing the capabilities and robustness of several time series regression methods, such as LSTM, GRU, and RNN. We envisaged that the proposed architecture in Figure 6.2 is robust enough to learn and predict unseen data—i.e., that it has the capacity for generalization. The reader may remark that several convolutions were applied to eliminate the data outliers. LSTM and GRU are the main layers to create the memory inside the proposed model. We have demonstrated the capacity of this architecture by testing on several popular time-series methods. Table 6.1 presents the results of comparing the training and validation MAE (mean absolute error) of our method and other architectures.

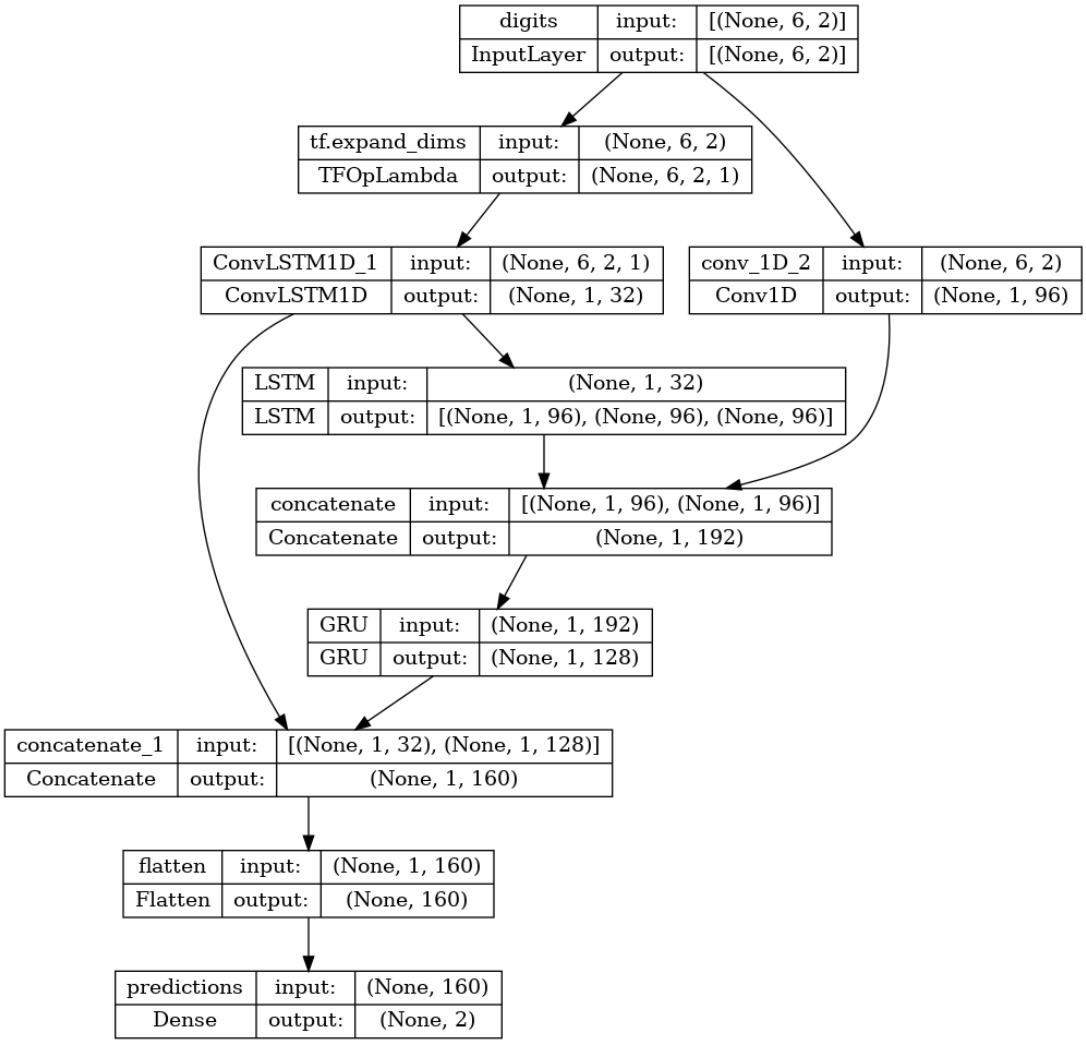


Figure 6.2: The architecture of the proposed model.

Table 6.1: Comparing several models to choose the best architecture.

Time Series Method	**Training MAE**	**Validation MAE**
LSTM	0.93	0.92
GRU	0.79	0.77
RNN	0.84	0.86
LSTM and RNN	0.86	0.86
GRU and RNN	0.76	0.77
OURs	0.75	0.75

6.3 Datasets and Parameters

6.3.1 Selection of Datasets

In order to train and test the model, we have used sequences from two popular benchmarks: Kitti [187] and Michigan North Campus Long-Term (NCLT) [190] datasets.

- **Kitti Dataset**: This dataset provides the opportunity to train the model in different categories provided in the dataset, including city, residential, road, campus, and person. This appropriate dataset enables also teaching the model in various challenging environments and investigating the performance of vehicles in short-term localization trajectories.
- **NCLT Dataset**: The North Campus Long-Term (NCLT) dataset was acquired with a Segway robot on one of the campuses of the University of Michigan, USA. It is a great option for training and testing the model's performance, since it contains long-term trajectory sequences with an average length of 5.5 km over a period of 15 months. The recordings included different times of day; different weather conditions; seasonal changes; indoor and outdoor environments; many dynamic objects, such as people and moving furniture; and two large, constantly changing construction projects. Although the trajectories vary considerably from session to session, there is considerable overlap.

Both datasets provide the information needed to train and test the model; the LiDAR point cloud was converted to the reference system, and ground truth of the positions was created based on GPS or/and a SLAM solution using LiDAR scan matching and high-accuracy RTK GPS. We have carefully distinguished sequences that have different sizes and contain various environmental scenarios, such as hard brakes, roundabouts, town drives, residential road drives, dirt roads, traffic, and sharp cornering, in order to teach robustly, as much as possible, the driving scenarios to the model. Table 6.2 depict the sequences used to train the model, which contain in total 50758 units of LiDAR scans (data input) and ground truth (target) with different velocity variations.

Table 6.2: The sequences used to train the model.

Sequence	**Dataset**	**Date**	**Frames Num**	**Kilometer**
00	Kitti	2011-10-03	4544	- [1]
05	Kitti	2011-09-30	2762	- [1]
06	Kitti	2011-09-30	1104	- [1]
07	Kitti	2011-09-30	1106	- [1]
08	Kitti	2011-09-30	5177	- [1]
09	Kitti	2011-09-30	1594	- [1]
10	Kitti	2011-09-30	1224	- [1]
00	NCLT	2012-01-08	28,127	6.4
25	NCLT	2013-01-10	5120	1.1
SUM	–	–	50,758	–

[1] The Kitti dataset did not mention the kilometers of driving the vehicles either in its official website or in its paper.

6.3.2 Data Processing and Parameters' Discussion

In this part, we have manipulated the data (input and target) to teach the model the vehicle's real positions (target) based on the features extracted from the LiDAR scan at each timestamp (input):

- **Processing the target**: Each dataset used here provides the ground truth of the vehicle positions. Along the trajectory, each timestamp of the position should coincide with the LiDAR scan timestamp or at least should be close to it (see Figure 6.3).
- **Processing the input data**: In order to prepare meaningful inputs for the model, a feature-extraction process had to be employed for each LiDAR scan to reduce the huge amount of points it contains. First, we performed a "cropping" step that cut the scan to keep only the region of interest. We kept only the points that surround the vehicle and do not exceed the height of the vehicle. In fact, we manually distinguished the points that respect these constraints, as seen in Figure 6.4b:

Let $P_t = [x_t, y_t, z_t]$ be a LiDAR point from the scan t:

$$\begin{aligned} -10 \text{ m} \leq x_t \leq 10 \text{ m} \\ -10 \text{ m} \leq y_t \leq 10 \text{ m} \\ z_t \leq 0.5 \text{ m} \end{aligned}$$

These limits were carefully chosen to ensure some overlap between scans. In addition, a fuzzy k-means clustering algorithm was performed to extract features from each cropped scan. The use of a clustering technique summarizes the LiDAR information into some core groups that are useful for explaining objects in the environment, e.g., trees, poles, and walls. Table 6.3 highlights how fast fuzzy k-means clustering is and that it is a good choice for this task, which is also proven in the article [43].

We decided to run the feature-extraction process with five center clusters, since the number of frames is big and each LiDAR frame (scan) contains a massive number of points, e.g., millions of points. Consequently, the calculation time will be huge if we choose a bigger number of clusters. However, we will analyze the contribution of each cluster (or feature) to the output model in the next section.
After calculating the five center cluster features at each scan, we calculate the mean of all those clusters and add it to the input as the sixth feature. Figure 6.5 provides an illustration of the inputs of our model, and Figure 6.4 illustrates an example of the following data processing step.

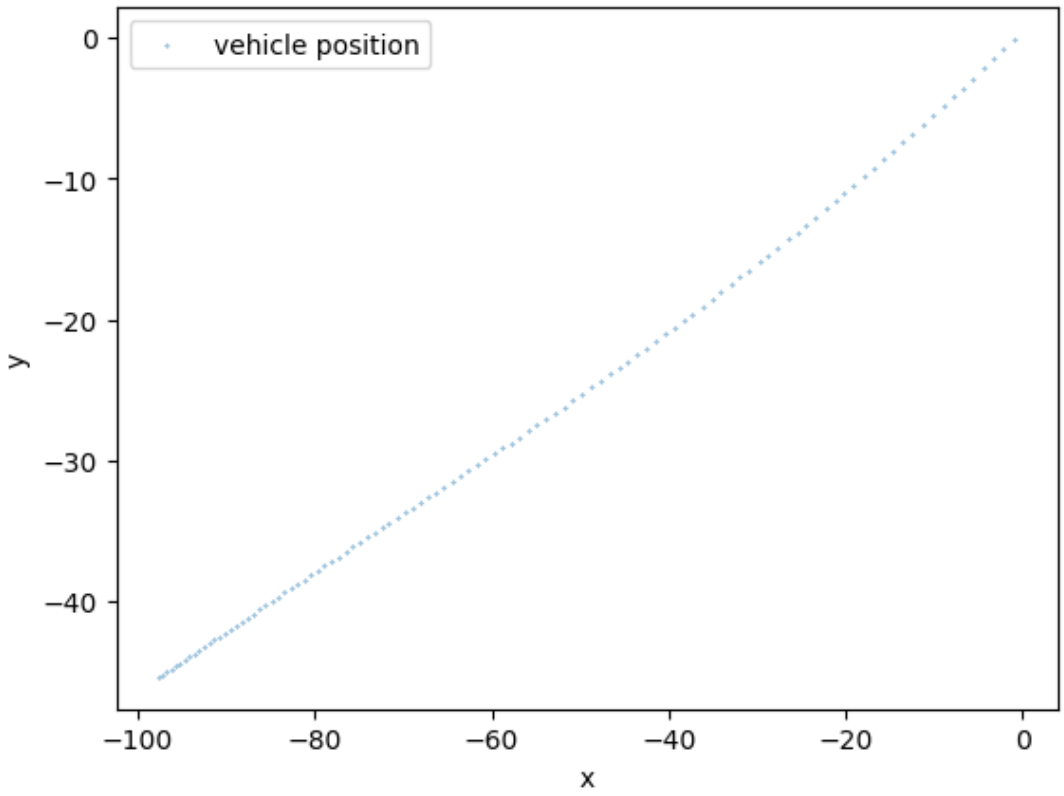
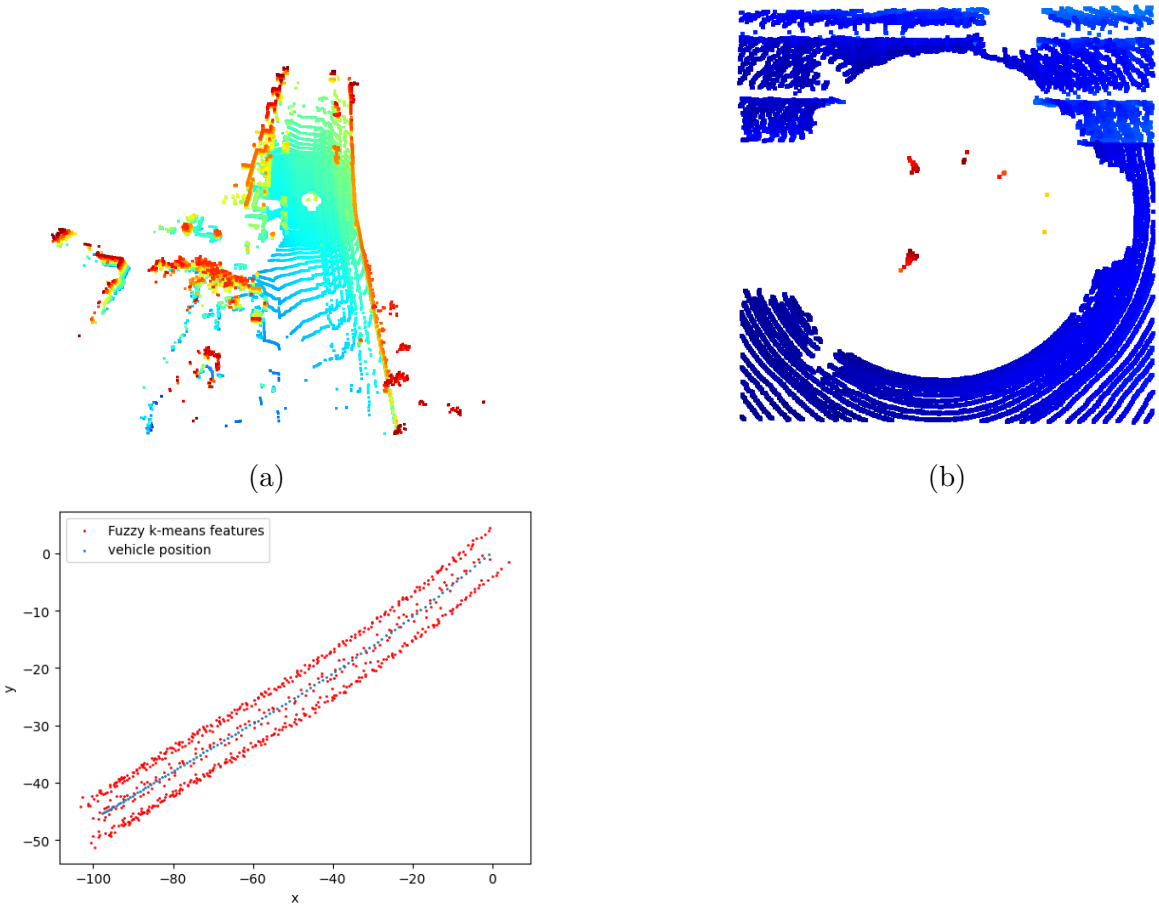


Figure 6.3: Example—sequence 0001 from Kitti dataset: Blue dots represent the real positions of the vehicle at each timestamp explained with Cartesian coordinates (x and y in meters).



(a) (b)

(c)

Figure 6.4: An illustration of the feature-extraction process, which crops the LiDAR scan into a square of 10 m × 10 m and with a height of 0.5 m. Then, a fuzzy k-means algorithm was applied to the cropped points. Figure 6.5 shows clearly how fuzzy k-means is able to extract relevant features for the trajectory. (**a**) LiDAR 3D points scan. (**b**) The cropped scan. (**c**) Fuzzy k-means features extracted (in red) and the ground truth positions (in blue)

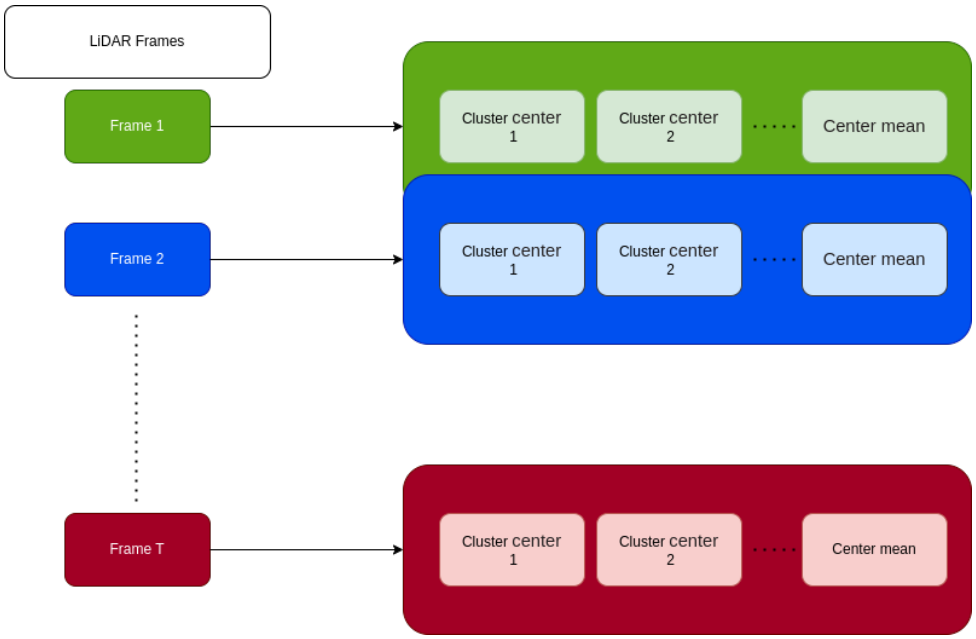


Figure 6.5: Inputs of the proposed model. Each frame contains 6 cluster-based features (five of them are the centers of clusters extracted using the fuzzy k-means algorithm, and the sixth is the mean of the five cluster centers). Each cluster contains 3D points. T is the number of frames.

Table 6.3: Time consumption (mm: ss) to execute the feature extraction for different clustering methods tested on the sequence 0001 from the Kitti dataset [**?**].

Clustering Method	**Time Cost**
KMeans	00:08
Fuzzy k-means	**00:07**
Gaussian mixture model	00:08
DBSCAN	00:10
Particle Swarm Optimized Clustering	00:09

6.3.3 Training the Model

- **Parameter tuning**:

  Finding the best parameter configuration is a crucial step to ensure the best accuracy with the minimum time cost, which is well needed for machine-learning applications. Self-driving vehicles require a fast and accurate positioning method to work in real-time. We have used a random algorithm that optimizes the parameters by selecting

random parameters after defining a range for each of them [216]. Table 6.4 depicts the results found after running 100 random trials. The best parameter combination achieved 1.21 m of the MAE in less than 30 s of training running. The choice of the optimization method was done manually by testing adam, adamax, SGD, RMSprop, and NAdam. We found that adamax gives the best results and converges to the minimum fastest.

- **Training the model**: The model was trained/validated on the datasets explained in Section 6.3.1 and based on the parameter search done as explained before. First, 30 epochs and 256 as a batch size were used to train the model. Early stopping was called if the loss value increased in 5 successive epochs. We reduced the learning rate if the loss metric stopped improving after 3 epochs.

Table 6.4: Parameter tuning for the model.

Layer	**Range of Choice**	**Best Choice**
ConvLSTM1D	[32, 130] [1]	32
LSTM	[32, 130] [1]	96
conv1D	[32, 130] [1]	96
GRU	[32, 130] [1]	128
activationLSTM	[“elu”, “linear”, “tanh”]	tanh
activationGRU	[“elu”, “linear”, “tanh”]	elu
IsDropout	[True, False]	False
learning rate	$[10^{-4}, 10^{-2}]$	0.0003

[1] 32 is the step size used here.

6.3.4 Model Explainability

- **Methods to investigate the contributions of features**:
  Explaining the deep learning model implies, in many cases, knowing how the features contribute (positively or negatively) to the output of the model, which makes it easy to understand for a human user. Recently, some improvements have been reported to provide more interpretation and transparency for the so called black box models. Several publications and open sources have been proposed to explain the models and make them human-readable, which is very useful in applications and promotes the use of machine learning. In this study, we selected and applied some methods

to explain the model. Note that all the methods cited here are adapted to be used for regression problems as in our case.

- Saliency: A saliency map is an image that marks the area on which people's eyes focus first. The purpose of a saliency map is to represent the degree of importance of a pixel to the human visual system [217]. In our case, it is the importance of point features to the vehicle position. We identified which features need to be changed least to affect the result of prediction the most.

Let us suppose at time t we have $C_t = \{c_{t,i}\}_{i\in[1,D']}$. Our clusters features are extracted at time t. We need to identify which $c_{t,i} = [x'_{t,i}, y'_{t,i}, z'_{t,i}]$ changed least, but affected the model most. Let f be a regression(the model). Then, the importance value $\Phi_{t,i}$ at time t for each cluster feature can be found by:

$$\Phi_{t,i} = \|\nabla_{c_{t,i}} f(c_{t,i})\|_\infty \tag{6.4}$$

where

$$\|x\|_\infty = \max(|x_0|, \ldots, |x_n|) = \max_{i=0}^{n} |x_i| \quad \text{where} \quad x \in \mathbb{R}^n$$

Note that $\nabla_{c_{t,i}} f(c_{t,i})$ is a vector of the gradient of each component of $c_{t,i}$. Calculating the infinity norm of $\nabla_{c_{t,i}} f(c_{t,i})$ provides only the importance value of the cluster feature i at time t. In order to identify which cluster contributes most to the output model, we need to find the index that corresponds to the maximum importance value:

$$j = argmax_i(\Phi_t) \quad \text{where} \quad \Phi_t = [\Phi_{t,1}, \ldots, \Phi_{t,D'}] \tag{6.5}$$

Consequently, $c_{t,j}$ will be the most important cluster that contributes most to the output model at time t. To calculate the importance values of clusters for all of the trajectory, we need to calculate the mean of the importance value of all the clusters over the trajectory:

$$\rho_1 = \frac{\Phi_{1,1} + \Phi_{2,1} + \cdots + \Phi_{T,1}}{T} \tag{6.6}$$

$$\rho_2 = \frac{\Phi_{1,2} + \Phi_{2,2} + \cdots + \Phi_{T,2}}{T} \tag{6.7}$$

$$\vdots$$

$$\rho_{D'} = \frac{\Phi_{1,D'} + \Phi_{2,D'} + \cdots + \Phi_{T,D'}}{T} \tag{6.8}$$

$$\tag{6.9}$$

In order to know which cluster feature contributes most to results output along the trajectory, we need to calculate:

$$o = argmax_k(\rho_1, \rho_2, \ldots, \rho_k, \ldots \rho_{D'}) \tag{6.10}$$

Cluster $c_{t,o}$ is the most interesting features that contribute to the output model for all of the trajectory, i.e., for every $t \in [1, T]$. We detailed above the method of how we get the importance value, which is supposed to be the same for all other methods, we just need to change the way we calculate the importance value, but everything else should be the same.

- Integrated gradient (IG): It computes the gradient of the model's prediction output relative to the input features and requires no modification of the original deep neural network. IG can be applied to any differentiable model used for images, text, or structured data [218].

$$\Phi_{t,i} = (c_{t,i} - x_0) \cdot \int_0^1 \nabla_{c_{t,i}} f\left(x_0 + \alpha\left(c_{t,i} - x_0\right)\right) d\alpha \tag{6.11}$$

where x_0 is the baseline, generally set to zero. The same method as in the salience method above is used to obtain the importance value and also to obtain the contributions of the cluster features.

- SmoothGrad: The gradient is averaged over several points corresponding to small perturbations around the point of interest. The smoothing effect by averaging reduces the visual noise and thus improves interpretation [219].

$$\Phi_{t,i} = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2)}\left(\nabla_{c_{t,i}} f(c_{t,i} + \delta)\right) \approx \frac{1}{N} \sum_{k=0}^{N} \nabla_{c_{t,i}} f\left(c_{t,i} + \delta_k\right) \tag{6.12}$$

It is the same calculation as in the salience method to obtain the importance value and the contributions of the cluster features. The idea is to introduce a random noise variable $\delta \sim \mathcal{N}\left(0, \sigma^2\right)$ in order to estimate the above expectation by sampling from the newly added noise (i.e., using a Monte Carlo estimator). N is the number of samples from δ.

- VarGrad: Similar to the Soomthgrad method. Instead of the mean gradient, the variance of the gradient is calculated [220].

$$\Phi_{t,i} = \mathbb{V}_{\delta \sim \mathcal{N}(0,\sigma^2)}\left(\nabla_{c_{t,i}} f(c_{t,i} + \delta)\right) \approx \frac{1}{N-1} \sum_{k=0}^{N} \left(\nabla_{c_{t,i}} f\left(c_{t,i} + \delta_k\right) - \hat{\mu}\right)^2 \tag{6.13}$$

where $\hat{\mu} = \frac{1}{N} \sum_{i=0}^{N} \nabla_{c_{t,i}} f\left(c_{t,i} + \delta_i\right)$ is the empirical mean. All other details are the same as for the SmoothGrad method above.

- **Methods' faithfulness**

  To measure the fidelity of these methods in explaining the model, the article [221] proposes several fidelity measures that indicate the extent to which one can trust the explanation of the results. We have chosen two methods of explanation:

  - Deletion: the metric evaluates the model's performance in making predictions while perturbing only the relevant features. A small value of deletion represents a more accurate explanation (note that this method was adapted for regression problems) [221].

  - Insertion: It measures how well a saliency-map-based explanation can find elements that are minimal for the prediction tasks. A higher value of insertion represents a more accurate explanation (note that this method was adapted for regression problems as well) [221].

6.4 Results and Discussion

The model was tuned, trained, and validated on a computer with 20GB RAM, 1TB Rom, and an i7 processor, which is capable of executing the commands rapidly. In order to test the performance of the model, we chose two types of trajectory scenarios, short-term and long-term. The short trajectories were chosen from the Kitti dataset [187], and the long trajectories were chosen for the NCLT dataset [190]. This section discusses the capacity of the model to predict unseen data in both scenarios (short-term and long-term trajectories). Then, we provide some explanations of the output results and provide an opportunity to improve the accuracy of the model. Finally, we compare the performances of the new and the old models and provide an investigation of their accuracy.

6.4.1 How the Model Performs:

According to Figure 6.6, the training loss decreased from 0.95 m to 0.75 m, which shows the importance of tuning the model. Meanwhile, the validation loss decreased from 0.915 to 0.75 m, which indicates that the model did not overfit.

Table 6.5 investigates the performance of the model in a series of tests on the Kitti dataset. We chose some small sequences to see how the model behaves in such a situation. We deduced that our model performs well in predicting the positions on these small

sequences. The mean absolute error of all the sequences did not surpass 1 m, which means that the majority of the error values are less than 1 m; we registered a maximum value of up to 2 m. Those sequences were chosen as they contain the most challenging scenarios, such as turning, roundabouts, acceleration, and weather changes.

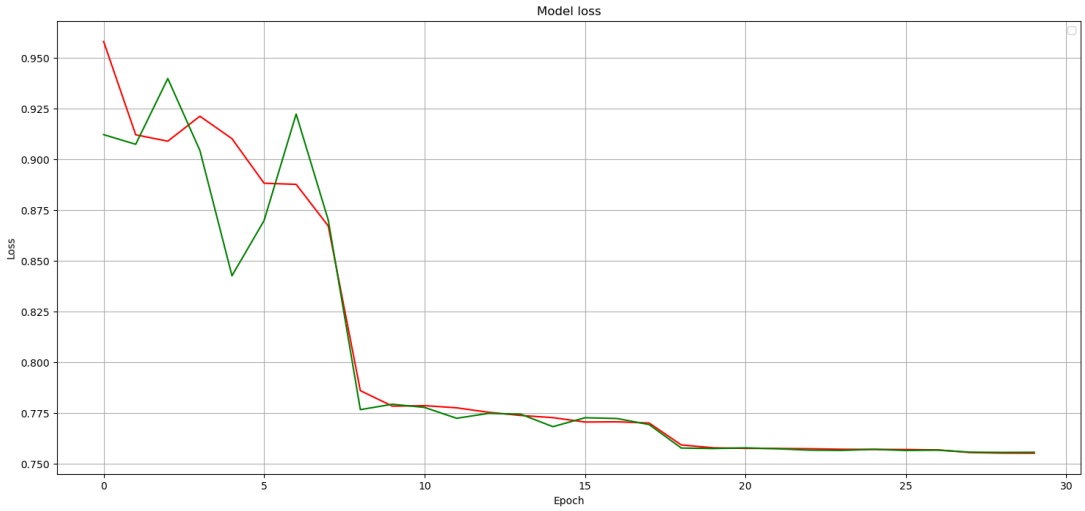


Figure 6.6: Graphical representation of the loss variation over epochs. The red curve is for the training loss, and the green curve is for validation.

Table 6.5: Error investigation for short-term trajectories (Kitti dataset).

Sequence	**Category**	**min**	**max**	**MAE** [1]
0001	City	0.0007	2.55	0.58
0009	City	0.003	3.4	0.63
0020	Residence	0.005	1.43	0.47
0035	Residence	0.0008	4.47	0.59
0052	Road	0.006	1.636	0.53
0027	Road	0.0077	6.47	0.82
0034	Campus	0.029	1.66	0.61
0035	Campus	0.004	1.45	0.53
0053	Person	0.0086	1.045	0.56
0054	Person	0.022	1.12	0.57

[1] MAE : mean absolute error.

Table 6.6 describes another situation where we roughly tested our model on long-term trajectories (more than 3 km) and in different environment scenarios and challenges. The model registered again a mean absolute error of less than 1 m between the predicted and

the real positions. However, is there any chance to improve the accuracy? The answer is yes, if we explain the results, e.g., discovering which features contribute positively or negatively to the output of the model.

Table 6.6: Error investigation for long-term trajectories (NCLT dataset).

Sequence	**Length**	**min**	**max**	**mae** [1]
2012-02-04	5.5 km	2.74×10^{-5}	5.65	0.85
2012-04-29	3.1 km	0.003	3.4	0.82
2012-06-15	4.1 km	5.88×10^{-5}	6.21	0.83
2012-12-01	5.0 km	8.02×10^{-6}	5.49	0.91
2013-04-05	4.5 km	8.02×10^{-6}	5.49	0.88

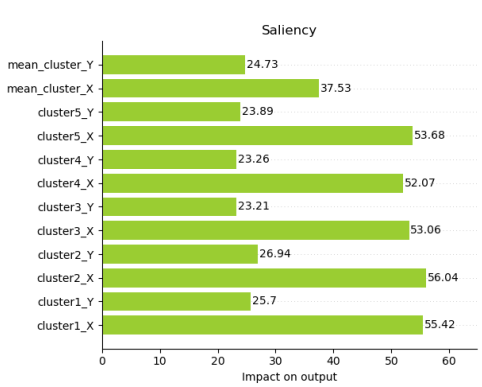
[1] mae : mean absolute error.

6.4.2 Why This Output?

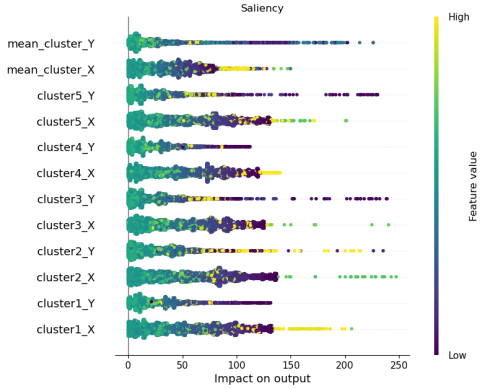
As mentioned in Section 6.3.4, four explainers were chosen to analyze the contribution ofs the features (including x and y axes) to provide the resulting output: saliency, integrated gradients, SmoothGrad, and VarGrad. We also used Kernal shap and the LIME explainer, which are very popular methods for regression problems. However, we got similar values for each feature, which does not provide much information about the contributions of the features. Figure 6.7 shows the mean impact of each feature (from x and y axes) according to each explainer (left column) and the punctual impact of each feature (from x and y axes) according to each explainer (right column). Our criterion for selecting features was the average impact value of the feature coordinates; on x and y axis. Based on that, saliency and integrated gradients (IG) nominated cluster 2 to be the most interesting feature. However, SmoothGrad selected the mean of the clusters as the least important feature, indicating that it contributes negatively to the model. Additionally, VarGrad confirmed that the mean of clusters has a major positive contribution to the model. Note that each explainer has its own philosophy to calculate the impact value. Thus, how can we decide which feature has the best contribution? We used two metrics of faithfulness: deletion and insertion, following another article [221]. According to Tables 6.7 and 6.8, SmoothGrad and VarGrad results are more trusted by the two criteria. SmoothGrad and VarGrad registered 89.61, and 105.72, respectively, in the deletion metric (lower is better), and 144.32 and 131.36, respectively, in the insertion metric (higher is better).

From Figure 6.7, we can extract more information from the punctual plots (right column), especially from Figure 6.7f,h. We remark that higher values for the mean of all feature data contribute positively, in contrast to the lowest values, which contribute negatively.

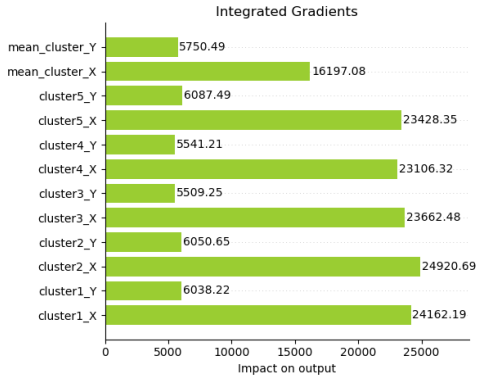
According to the analysis above, the mean of all the clusters is the most important feature that contributes positively to reducing the loss error.



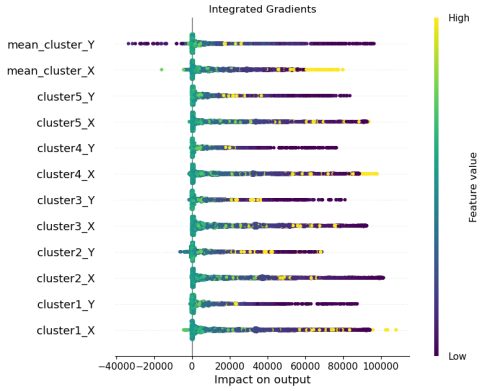
(a)



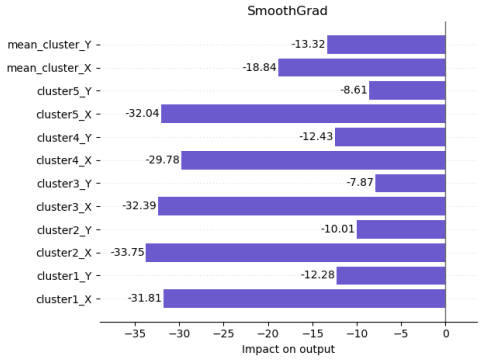
(b)



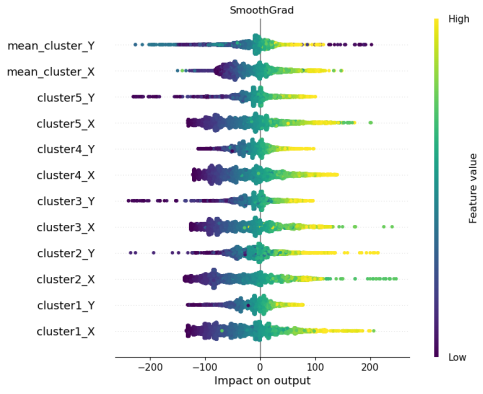
(c)



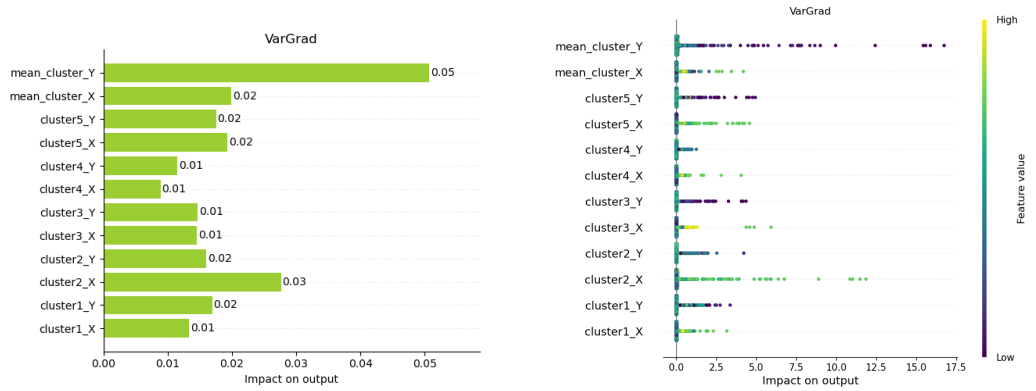
(d)



(e)



(f)



(g) (h)

Figure 6.7: Mean and punctual impact for all four explainers. (**a**) Mean impact for every sample of the saliency explainer. (**b**) Punctual impact for every sample of the saliency explainer. (**c**) Mean impact for every sample of the integrated gradients explainer. (**d**) Punctual impact for every sample of the integrated gradients explainer. (**e**) Mean impact for every sample of the SmoothGrad explainer. (**f**) Punctual impact for every sample of the SmoothGrad explainer. (**g**) Mean impact for every sample of the VarGrad explainer. (**h**) Punctual impact for every sample of the VarGrad explainer. Punctual impact explains the effect of each cluster's coordinates on our model, such as in images (**b**,**d**,**e**,**h**). On the right side of each image, there is a color bar that shows the contributions of the clusters: purple means low contributions and yellow means high contributions. On the x axis, we have the impact calculated according to each method.

Table 6.7: Deletion criterion results.

Method Name	**Deletion Score (Lower Is Better)**
Saliency	139.592417
Integrated Gradients	140.074941
SmoothGrad	89.615382
VarGrad	105.721736

6.4.3 What Will Happen If We Change That?

We then explored the fact that the mean of all the clusters is the most important feature to create another model with the same architecture as the first one and compared the prediction results of both models to investigate the improvement. Figure 6.8 presents the curve of the loss values (training and validation). Training loss decreased from 13.49 to

0.68, and validation loss decreased from 4.41 to 0.68, which is quite a lot lower than what we got in the first model.

Table 6.8: Insertion criterion results.

Method Name	**Insertion Score (Higher Is Better)**
Saliency	95.408970
Integrated Gradients	94.097169
SmoothGrad	144.325370
VarGrad	131.362521

Table 6.9 shows the prediction results of the newly developed model. We found that the MAE of almost all sequences was reduced compared to of the first model (Table 6.5). The average MAE of all sequences was 0.51 m, whereas it was 0.58 m for the previous model. The new model improved the position-prediction accuracy compared to the first model. Not only that, but we removed the cluster features that mislead the model and are too time consuming in the extraction and training phase. We underpinned these conclusions by testing the model in long-term sequences as well. Table 6.10 presents the results prediction for long-term trajectories, and our model achieved lower MAE values compared with the first model.

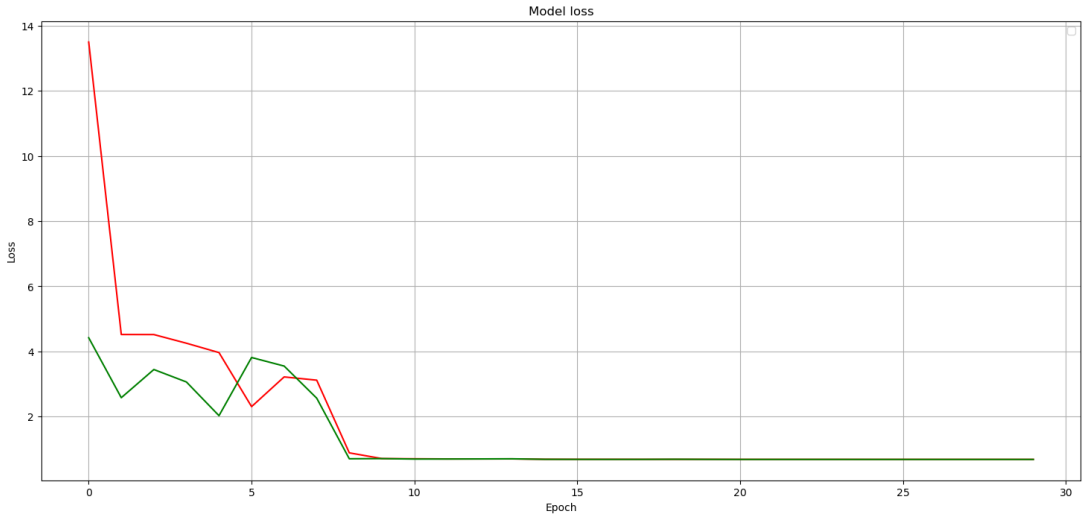


Figure 6.8: Graphical representation of the loss variation of the explained model. The red curve is for the training loss, and the green curve is for validation.

Table 6.9: Results predictions of the new explained model in short-term trajectories (Kitti dataset).

Sequence	**Category**	**min**	**max**	**MAE** [1]
0001	City	0.017	2.26	0.54
0009	City	0.0007	2.3	0.58
0020	Residence	0.0009	1.43	0.45
0035	Residence	0.003	4.36	0.54
0052	Road	0.012	2.15	0.59
0027	Road	0.002	5.1	0.612
0034	Campus	0.051	1.21	0.65
0035	Campus	0.051	1.21	0.48
0053	Person	0.001	0.97	0.374
0054	Person	0.059	0.97	0.37

[1] MAE : mean absolute error.

Table 6.10: Predictions of the new explained model for the long-term trajectories (NCLT dataset).

Sequence	**Category**	**min**	**max**	**MAE** [1]
2012-02-04	5.5 km	7.6×10^{-6}	6.04	0.80
2012-04-29	3.1 km	1.2×10^{-5}	5.44	0.779
2012-06-15	4.1 km	4.89×10^{-5}	6.086	0.77
2012-12-01	5.0 km	2.83×10^{-5}	5.76	0.86
2013-04-05	4.5 km	1.3×10^{-5}	5.97	0.82

[1] MAE : mean absolute error.

6.5 Conclusions

Localization and mapping in self-driving vehicles have been extensively treated with different approaches, including probabilistic, optimization, and other machine-learning methods. In this paper, we presented a novel deep-learning workflow for learning vehicle positions. The input features of the model were extracted from the LiDAR scans at each time point. The extraction process was based on the application of a fuzzy k-means algorithm that extracts features from clusters. The model architecture was based on a

combination of LSTM and GRU model layers and smoothing with a 1D convolution. We tuned the model to obtain the best hyperparameters, and we trained the model with different parameters, such as early stopping, reduction of the learning rate in the case of constant metrics, etc. The model has obtained very good short- and long-term results in different challenging environmental scenarios, such as weather changes and various trajectory scenarios. The model is able to keep the mean absolute positioning error below 1 m for all sequences in short- and long-term trajectories.

We also provided possible explanations for the model's results and examined the contribution of the clusters (features). We found that the mean of all the extracted clusters is the most important feature that contributes positively to the prediction result. We created a new explained model that takes only the mean of all clusters as input and ran the prediction again. According to the comparison results of both models, the new, more explainable model improves the accuracy of vehicle positioning and reduces the time and computational resources required to train and use these models.

CHAPTER 7

AUTOLOCAL: A MODIFIED AUTOENCODER-BASED APPROACH TO AUTOMATED LOCALIZATION FOR SELF-DRIVING VEHICLES

7.1 Introduction

Autonomous vehicles (AVs) are revolutionizing the auto industry and could radically change the way we travel. Self-driving vehicles are capable of independently driving through crossroads, evading obstacles, and making decisions without the need for human intervention. Self-driving vehicles use a wide range of sensors, including cameras, radar, and LiDAR, that enable them to detect and respond to their environment. They can be programmed with certain rules and parameters to guide their behavior and ensure the safety of the vehicle and its occupants. Self-driving vehicles offer a number of potential benefits, including increased safety, improved mobility for the elderly and disabled, and reduced traffic congestion and pollution. [4]

Localization is an essential component of autonomous vehicles, as a vehicle must be able to accurately determine its position and orientation relative to its surroundings. This is especially important for autonomous cars in order to drive on the road safely and efficiently. The primary objective of tackling the localization problem is to facilitate precise and secure navigation for autonomous vehicles, ultimately leading to a decrease in traffic accidents and fatalities attributable to human error [223] and emphasizes the significance

of monitoring the physiological side of the driver in understanding the implications of AVs comprehensively [224]. These potential benefits make localization an important and necessary task for autonomous vehicles [4] [18].

The main challenges of localization for self-driving vehicles are accuracy, robustness, and cost [18]. Self-driving vehicles must be able to accurately determine their position and orientation in relation to their environment in order to safely and efficiently navigate roads. This requires a combination of sensors, algorithms, and mapping techniques that are both accurate and cost-effective. Additionally, the localization system must be reliable and robust enough to handle varying environmental conditions such as snow, fog, and darkness. Finally, the system must be scalable and able to handle large amounts of data in order to keep up with the growth of autonomous vehicle technology [7].

The Global Navigation Satellite System (GNSS) represents a feasible solution for the localization of autonomous vehicles. This satellite-based navigation system offers precise location and speed data for the vehicle, and can also provide up-to-date traffic information to enable the self-driving vehicle to choose the most optimal route to its intended destination. However, GNSS cannot accurately locate a self-driving vehicle in certain situations, such as when the vehicle is in a tunnel or deep underground, or when tall buildings block the GNSS signal. GNSS can also be affected by signal interference from other electronic devices, or if the vehicle is moving faster than the signal can be updated. In addition, GNSS is not accurate enough to locate self-driving vehicles in small spaces, such as city streets. If a self-driving vehicle relies solely on GNSS for positioning, it will have difficulty navigating in such environments [174] [175]. Consequently, our objective will be to replace the GNSS solution with other methods that are more robust and accurate.

These major problems were covered in the literature using different approaches. The Inertial Navigation System (INS) provides vehicles with acceleration, velocity, and orientation information. By combining these measurements over time, INS can calculate the vehicle's position and orientation relative to its starting point. INS is an attractive option for locating self-driving cars because it is relatively accurate and can work in environments where GPS signals are weak or unavailable, such as covered parking lots. However, it has some limitations. One of the biggest problems with INS is that it can "drift" over

time. This means that even small errors in the initial position or orientation estimates can accumulate, causing the system to lose accuracy over time [176]. Several papers have tried to correct these uncertainties by using deep learning models which minimize the cost function:

$$||x_{ins} - x_{gnss}||$$

where x_{ins} are the positions calculated by the INS and x_{gnss}. long short-term memory has registered excellent results among the others [179, 177].

The state of art methods can be divide into registration-based methods, probabilistic methods, and deep learning methods. As the images can be significantly impacted by various weather conditions such as snow or rain, we are focusing solely on techniques that rely on LiDAR data. Registration-based methods employ matching scan strategies to track the vehicle's movements across consecutive sequences. Examples of registration-based methods include the LOAM [200] method which distinguishes edges and flat planar features for motion tracking between consecutive sequences. However, the LOAM method can be time-consuming to execute due to noisy and irrelevant features in LiDAR measurements. Two approaches, namely LeGO-LOAM [202] and A-LOAM [203], have addressed this challenge by eliminating such features. In addition, deep learning techniques have been used in LO-Net and LO-Net-M [204] to perform the matching scsan process. Another method called SGLO [205] has also demonstrated promising results by incorporating geometric features such as lines and planes. However, the accuracy of SGLO heavily relies on initialization. In vehicle localization also, probabilistic methods utilize a statistical approach to detect semantic objects such as poles and walls in LiDAR scans. This approach allows for the correction of accumulated errors in the inertial measurement unit (IMU) by means of particle filter, particle filter, or extension algorithms. Several examples of such methods are proposed in [27], [29], [26], and [31]. The method proposed by Schaefer et al. [31] has achieved the most accurate estimations of vehicle trajectories on both the Kitti dataset [187] and the NCLT dataset [190]. However, these methods may not work well in environments with minimal texture, such as desert roads. Charroud et al. [43] address this issue by proposing non-semantic features, extracted from LiDAR scan measurements, in the particle filter's measurement-update step. Those features are characterized by their robustness against the moving objects. This approach allows for the extraction of environmental information even in the absence of sufficient texture. To improve its speed, the authors modified the particle filter architecture and utilized sigma-point selection tech-

niques to choose the relevant particles for position calculation. This modification has been shown to enhance both the speed and accuracy of the method [45]. Charroud et al. [44] proposed an extension to their previous work by introducing a technique that guides particles to follow the best estimation. The authors select the 10 closest particles around the actual position after each iteration of the particle filter algorithm and generate particles around them. This extension has demonstrated fast execution and excellent accuracy. In addition, a machine learning approach was introduced in [222] to precisely determine the positions of the vehicle. The model leverages the center cluster obtained from [43] as input. The authors also performed an analysis of the features that had a positive or negative impact on the output of the model to provide insights into the results.

This work modifies the autoencoder architecture to provide the vehicle's positions by exploring the results of the latent space. Autoencoders are commonly utilized to decrease the complexity of input data by preserving the majority of its essential features. This is accomplished by minimizing the loss of information in the encoded representation. However, it is difficult to interpret the results of the latent space and give them a sense. The idea is to modify the objective function of the autoencoder by including the learning of the vehicle's positions and forcing the latent space to provide them. Such that, the input is represented by non-semantic features extracted from LiDAR scans [43]. This approach is inspired by the use of approach in [222]. Consequently, the model learns how to project the features on the vehicle coordinates dimension (x and y axis) and at the same time learns the vehicle's positions. The mathematical convergence of our algorithm was proven, which updates the parameters of the model. In addition to that, the model has undergone extensive testing on the benchmark sequences of the Kitti dataset [187], including the long-term ones, and has outperformed the existing state-of-the-art methods. This paper offers several significant contributions, which can be summarized as follows:

- To the best of our knowledge, this is the first research paper that concentrates on acquiring knowledge of the latent space and customizing the outcomes for detecting the locations of vehicles.
- Mathematical proofs for the convergence of the proposed algorithm are provided.
- Our methods register excellent accuracy compared with the state of the art. Moreover, we obtained approximately 0.5 mean absolute error for all the long-term sequences, which is recommended for safe navigation.

This paper is divided into several sections to effectively present the research and its findings. In section 7.2, the problem statement is introduced to explain the issue being addressed. The core of our proposed solution is presented in section 7.3, where we provide a proof of convergence for the algorithm. Our automatic localization (autolocal) architecture is detailed in section 7.4, which is utilized to train the model on a dataset. section 7.5 outlines the experiments that were conducted and the corresponding results. Finally, in section 7.6, a conclusion is drawn, summarizing the research, highlighting its contributions, and discussing potential future directions.

7.2 Problem statement:

Autoencoders are neural network architectures designed for unsupervised learning, and they consist of two parts - an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation known as a latent code or bottleneck, while the decoder expands the latent code back into the original data space. The primary objective of an autoencoder is to minimize the reconstruction error between the original data and the output generated by the decoder. See figure 7.1

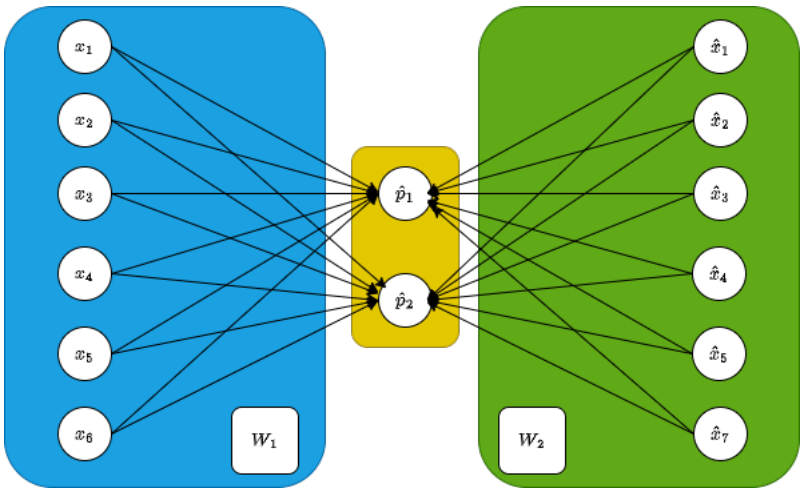


Figure 7.1: Example of autoencoder architecture. The blue phase is the encoder that transforms/projects the input data to the latent space (yellow box). The green phase sends back the data from the latent space to its original dimension. W_1 and W_2 are the corresponding vectors of weights of the hidden layer and output layer, respectively.

Mathematically, a two-layer autoencoder can be defined as a neural network that

consists of two main parts: an encoder and a decoder. Given an input data sample x, the encoder maps it to a lower-dimensional representation $\hat{p}$ through a non-linear mapping function f:

$$\hat{p} = f(W_1, x) \tag{7.1}$$

In this article, we denote $f(\langle ., . \rangle)$, interchangeably, by $f\langle ., . \rangle$ or $f(., .)$. W_1 are the weights of the hidden layer.

The decoder then maps the lower-dimensional representation $\hat{p}$ back to the original data space through a non-linear mapping function g:

$$\hat{x} = g(W_2, \hat{p}) \tag{7.2}$$

Again, in this article, we denote $g(\langle ., . \rangle)$, interchangeably, by $g\langle ., . \rangle$ or $g(., .)$. W_2 are the weights of the output layer.

The primary objective of using an autoencoder is to train a mapping function that can reduce the difference or reconstruction error between the input data x and the output or reconstructed data $\hat{x}$:

$$Loss_{\{W_1, W_2\}}(x) = ||x - \hat{x}||^2 = ||x - g(W_2, f(W_1, x))||^2 \tag{7.3}$$

where *Loss* is the reconstruction loss, and $||.||$ is a norm that measures the distance between x and $\hat{x}$. The loss function is minimized through an optimization process, such as stochastic gradient descent (SGD), to update the parameters of the encoder and decoder.

One of the main drawbacks of autoencoders is that the latent space learned by the encoder can be uncontrollable. The model learns the encoding and decoding processes, and the resulting representations may not always be interpretable or have the desired properties. We want to give a meaning to the results of the latent space, e.g. approximating vehicles' positions in our case.

We propose to modify the minimization problem of the cost function in equation 7.3 by adding another objective function that will control the learning of the latent space to the desired aim, e.g. provide an accurate approximation of the vehicle in our case. The mathematical formulation of the multi-parameter optimization for this problem can be written as follows.

The first objective is to minimize the reconstruction error between the original data x and the reconstructed data $\hat{x}$:

$$Loss^1_{\{W_1,W_2\}}(x) = ||x - \hat{x}||^2 \tag{7.4}$$

The second objective is to minimize the difference between the latent space vector $\hat{p}$ and the desired output, e.g. vehicle's positions p in our case:

$$Loss^2_{\{W_1\}}(x) = ||\hat{p} - p||^2 \tag{7.5}$$

The multi-parameter objective function of our optimization problem is then to find a solution for this minimization problem:

$$\min_{W_1,W_2} \quad J_{\{W_1,W_2\}} = \min_{W_1,W_2} (\alpha Loss^1_{\{W_1,W_2\}} + (1-\alpha) Loss^2_{\{W_1\}}) \tag{7.6}$$

where α is a penalty value added to favour the learning of vehicle's positions, i.e., minimizing $Loss_{\{W_1\}}$. An illustration of what we discussed above is shown in Figure 7.2.

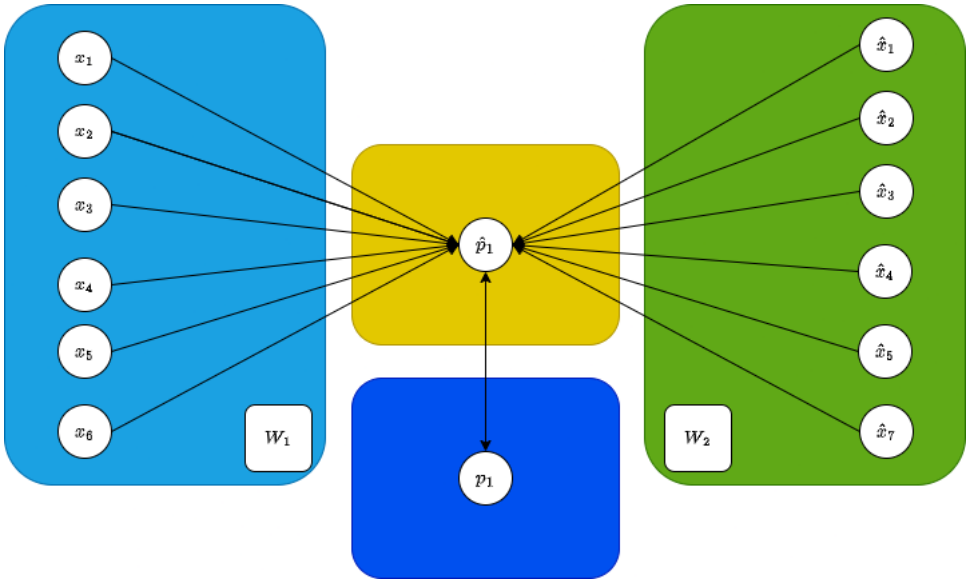


Figure 7.2: An adaptation of the autoencoder represented in Figure 1, where we have added another phase (dark blue color) to show the optimization process between the latent space and the real positions

In the following section, we will discuss the theoretical convergence of the minimization problem presented in equation 7.6 and Algorithm 1, which outlines the weight update procedure for W_1 and W_2. The convergence of Algorithm 1 largely depends on the convergence of the updated weights (W_1 and W_2), which will be demonstrated in the next

section.

Algorithm 1 Algorithm for updating autoencoder weights

Input: Training data $x = \{\mathbf{x}_i\}_{i=1}^{b}$ where $\mathbf{x}_i \in \mathbb{R}^3$ and $p = \{p_i\}_{i=1}^{b}$ where $p_i \in \mathbb{R}^2$, b is the batch size. Initialization of parameters W_1^0, W_2^0. α penalty parameter between $[0, 1]$, ϵ threshold to stop training, T number of iterations.

$t = 0$

while $[\|f\left\langle W_1^t, x\right\rangle - p\|_2^2 > \epsilon$ OR t!=T] **do**

$W_2^t \leftarrow \arg\min_W \left\|g\left\langle W, f\left\langle W_1^{t-1}, x\right\rangle\right\rangle - x\right\|_2^2$ // use Algorithm 3

$W_1^t \leftarrow \arg\min_W (\alpha \|g\left\langle W_2^t, f\left\langle W, x\right\rangle\right\rangle - x\|_2^2 + (1-\alpha)\|f\left\langle W, x\right\rangle - p\|_2^2)$ // use Algorithm 4

t=t+1

end while

7.3 Convergence of Algorithm 1

In this section, we delve into the convergence of the algorithm 1 in the context of an extended version of the Stochastic Gradient Descent (SGD) algorithm. Before we proceed with a detailed analysis of the convergence behavior, we provide a preliminary overview of the SGD algorithm and the modifications made to extend it. Also, we provide definitions and properties of the so-called strictly-local-quasi-convex functions and their role to ensure the convergence of the extended SGD algorithm.

7.3.1 Stochastic Normalized Gradient Descent (SNGD):

Stochastic gradient descent (SGD) is a widely used optimization algorithm that finds the optimal solution for convex and Lipschitz objectives using an unbiased gradient estimator that can be derived from a small number of data samples. However, its effectiveness is limited when applied to non-convex problems [225]. This is particularly evident in the field of deep learning, where stochastic non-convex optimization problems are prevalent. These problems often exhibit two contrasting behaviors: regions with very small gradients (plateaus) and regions with excessively high gradients (sharp cliffs). As a result, the

application of SGD to these problems is often reported to produce unsatisfactory results [225].

This paper proposes Stochastic Normalized Gradient Descent ("SNGD"), an extension of the SGD algorithm aimed at overcoming the constraints arising from non-convex optimization problems. Compared to the SGD algorithm, the SNGD algorithm is better adapted to these constraints and remains equally efficient at each iteration. The algorithm 2 is proposed to illustrate how the SNGD algorithm updates the parameters. The proposed approach is based on the work of [225].

Algorithm 2 Stochastic Normalized Gradient Descent (SNGD) [225]

Input: Iterations number T, $\mathbf{x}_1 \in \mathbb{R}^d$, learning rate η, minibatch size b, Dataset D^b.

for $t = 1 \dots T$ **do**

Sample: $\{\theta_i\}_{i=1}^{b} \sim \mathcal{D}^b$, and define

$\rho_t(\mathbf{x}) = \frac{1}{b}\sum_{i=1}^{b} \theta_i(\mathbf{x})$

Update:

$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\hat{\xi}_t$ where $\xi_t = \nabla\rho_t(\mathbf{x}_t), \hat{\xi}_t = \frac{\xi_t}{\|\xi_t\|}$

end for

Return: $\overline{\mathbf{x}}_T = \arg\min_{\{\mathbf{x}_1,\dots,\mathbf{x}_T\}} \rho_t(\mathbf{x}_t)$

7.3.2 Convergence of Strictly-Locally-Quasi-Convex (SQLC) with SNGD:

Definition 1. *:(Local-Quasi-Convexity [225]) Let* $\mathbf{x}$ *and* $\mathbf{z}$ *be vectors in* $\mathbb{R}^d$*, and let* κ *and* ϵ *be positive numbers. We say that a function* $\phi : \mathbb{R}^d \to \mathbb{R}$ *is* $(\epsilon, \kappa, \mathbf{z})$*-Strictly-Locally-Quasi-Convex (SLQC) at* $\mathbf{x}$ *if at least one of the following is true:*

1- The difference between $\phi(\mathbf{x})$ *and* $\phi(\mathbf{z})$ *is less than or equal to* ϵ*.*

2- The norm of the gradient of $\phi(\mathbf{x})$ *is greater than zero, and for every* $\mathbf{y} \in \mathbb{B}(\mathbf{z}, \epsilon/\kappa)$*, where* $\mathbb{B}(\mathbf{z}, \epsilon/\kappa)$ *is the ball of radius* ϵ/κ *centered at* $\mathbf{z}$*, it holds that* $\langle\nabla\phi(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle \leq 0$*.*

Lemma 1. *:*

Let

$$SQ = \{\phi : \mathbb{R}^d \mapsto \mathbb{R} \mid \phi \text{ is an } (\epsilon, \kappa, \mathbf{z}) - SQLC\}$$

If $\{\phi_1, \phi_2, ..., \phi_b\}$ *and* ψ *are functions from SQ set then:*

- $\sum_{i=1}^{b} \phi_i \in SQ$
- $\lambda\psi \in SQ$ *for* $\lambda \geq 0$

Proof of the lemma :

Before demonstrating the lemma 1, we prove the following remark:

Remark 1. *In the definition 1, the condition 2 imply the condition 1.This remark will help to demonstrate the Lemma 1.*

Proof of the remark :

Let $y \in \mathbb{B}(z, \frac{\epsilon}{\kappa})$

where

$$\langle \nabla\phi(x), y - x\rangle \leq 0 \quad \text{and} \quad ||\phi(x)|| > 0$$

which means that

$$\langle \nabla\phi(x), x - y\rangle \geq 0$$

we set

$$a = x - y \quad \text{and} \quad \beta > 0$$

Then by Taylor expansion

$$\phi(x + \beta a) = \phi(x) + \langle \nabla\phi(x), \beta a\rangle + \beta||a||\xi(\beta a)$$

where $\lim_{\beta\to 0} \xi(\beta a) = 0$

Then

$$\phi(x + \beta a) - \phi(x) = \beta\langle \nabla\phi(x), x - y\rangle + \beta||x - y||\xi(\beta(x - y))$$

As

$$\langle \nabla\phi(x), x - y\rangle \geq 0 \quad \text{and} \quad \lim_{\beta\to 0} \xi(\beta a) = 0$$

We have the existence of $\beta_0 > 0, \beta_0 >> \beta$ where

$$\langle \nabla\phi(x), x - y\rangle + ||x - y||\xi((x - y)) \geq 0$$

Then

$$\phi(x + \beta a) \geq \phi(x)$$

we chose $z = x - \alpha(y - x)$ then

$$\phi(z) \geq \phi(x)$$

which give

$$\phi(x) - \phi(z) \leq 0 < \epsilon$$

Proof of $\sum_{i=1}^{b} \phi_i \in SQ$:

Let $x, z \in \mathbb{R}^d$ and λ a scalar. Let $\{\phi_1, \phi_2, ..., \phi_b\}$ be a suite of functions where ϕ_i is $(\epsilon_i, \kappa_i, z) - SQLC$ and let ψ also be $(\epsilon, \kappa, z) - SQLC$

In order to demonstrate the lemma, we distinguish between three cases:

- $\{\phi_1, \phi_2, ..., \phi_b\}$ are $(\epsilon_i, \kappa, z) - SQLC$ and verified the first condition
- $\{\phi_1, \phi_2, ..., \phi_b\}$ are $(\epsilon_i, \kappa, z) - SQLC$ and some of them verified the first condition. However the other verified the second condition.
- $\{\phi_1, \phi_2, ..., \phi_b\}$ are $(\epsilon_i, \kappa, z) - SQLC$ and verified the second condition

$\star$ First case

we have:

$$\begin{aligned}
&(\sum_{i=1}^{b} \phi_i)(x) - (\sum_{i=1}^{b} \phi_i)(z) \\
&\quad = (\sum_{i=1}^{b} \phi_i(x) - \sum_{i=1}^{b} \phi_i(z)) \\
&\quad = (\sum_{i=1}^{b} \phi_i(x) - \phi_i(z)) \\
&\quad \leq \epsilon_1 + \epsilon_2 ... + \epsilon_b = \epsilon'
\end{aligned} \tag{7.7}$$

Thus

$$\sum_{i=1}^{b} \phi_i \in SQ$$

$\star$ Second and Third case:

Based on Remark 1, any function that satisfies the second condition will automatically satisfy the first condition as well. Therefore, proving the first case is sufficient to demonstrate the validity of this lemma. In other words, if we establish the correctness of the first condition, it ensures that the second condition is also met without requiring a separate proof.

Proof of $\lambda\psi \in SQ$:

Also we have two cases to achieve the demonstration:

- ψ is $(\epsilon, \kappa, z) - SQLC$ and verified the first condition
- ψ is $(\epsilon, \kappa, z) - SQLC$ and verified the second condition

$\star$ First case

$$\begin{aligned}(\lambda\psi)(x) - (\lambda\psi)(z) \\ =&\lambda(\psi(x) - \psi(z)) \\ \leq&\lambda\epsilon = \epsilon'\end{aligned} \tag{7.8}$$

$\star$ Second case

According to remark 1, ψ verified the first condition which is already proven in the first case.

The article [225] presents evidence on **Theorem 1** that the SNGD algorithm converges to the minimum for the class of SQLC.

Theorem 1. *(Hazan [225]) Let δ, ϵ, G, M, and κ be positive constants. Assume we use SNGD for T iterations with step size $\eta = \epsilon/\kappa$ and batch size $b \geq \max(\frac{M^2 \log(4T/\delta)}{2\epsilon^2}, b_0(\epsilon, \delta, T))$, where $b_0(\epsilon, \delta, T)$ is a function of ϵ, δ, and T. Suppose that for $b \geq b_0(\epsilon, \delta, T)$, with probability at least $1 - \delta$ and for all $t \in [T]$, the function ρ_t defined in Algorithm 2 is M-bounded and $(\epsilon, \kappa, \mathbf{x})$-Strictly-Locally-Quasi-Convex (SLQC) in $\mathbf{x}_t$, where $\mathbf{x}^*$ is the optimal solution. Then, with probability at least $1 - 2\delta$, we have $\rho(\overline{\mathbf{x}}_T) - \rho(\mathbf{x}^*) \leq 3\epsilon$, where $\overline{\mathbf{x}}_T$ is the average of the iterates $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and $\rho(\mathbf{x})$ is the objective function evaluated at $\mathbf{x}$.*

7.3.3 Main results:

Our goal here is to demonstrate the convergence of our Algorithm 1. One way to do that is to demonstrate the convergence of each of the parameters W_1 and W_2 asymptotically. We propose to prove the convergence of the SNGD algorithm for both parameters, i.e. we need to prove the convergence of Algorithms 3, and 4.

We define:

- $W_1 \in \mathbb{R}^{d \times d'}$ as the weights of the first layer, i.e. from the input layer with dimension d to the hidden layer with dimension d'.
- We denote $W^1_{:,k}$ the $k - th$ vector of dimension d of W_1.

- $W_2 \in \mathbb{R}^{d' \times d}$ is the corresponding weights of the second layer i.e from the hidden layer with dimension d' to the output layer with dimension d.
- We denote $W^2_{:,j}$ the $j-th$ vector of dimension d' of W_2.
- $x \in \mathbb{R}^d$ is the input data of dimension d
- $p \in \mathbb{R}^{d'}$ is true data to compare with the latent space (vehicle's positions in our case).
- $g : \mathbb{R} \longrightarrow \mathbb{R}$ is the decoder function
- $f : \mathbb{R}^{d'} \longrightarrow \mathbb{R}^{d'}$ is the encoder function.
- We define an important function $h : \mathbb{R} \longrightarrow \mathbb{R}$ which is an equivalent function of f but its output is on $\mathbb{R}$

The objective of this study involves minimizing two objective functions for a two-layer autoencoder that has multiple neurons in its input/output layers and one hidden layer.

$$\min_W L1_{\{W_1\}}(x) = \min_W \|g\left\langle W, f\left\langle W_1, x\right\rangle\right\rangle - x\|_2^2$$

and

$$\min_W L2_{\{W_1\}}(x) = \min_W(\alpha \|g\left\langle W_2, f\left\langle W, x\right\rangle\right\rangle - x\|_2^2 + (1-\alpha)\|f\left\langle W, x\right\rangle - p\|_2^2)$$

which are an explicit explanation of the equation 7.6.

To make it easy to understand, we only need to prove that (b is the batch size of $x \in \mathbb{R}^d$ and b' is the batch size of $p \in \mathbb{R}^{d'}$)

$$L1_{\{W_1\}}(x) = \frac{1}{b}\sum_{i=1}^{b}\left(g\left\langle W_{:,i}, f\left\langle W_1, x\right\rangle\right\rangle - x_i\right)^2 \tag{7.9}$$

and

$$L2_{\{W_1\}}(x) = \sum_{i=1}^{b}\frac{\alpha}{b}\left(g\left\langle W^2_{:,i}, f\left\langle W, x\right\rangle\right\rangle - x_i\right)^2 + \sum_{k=1}^{b'}\frac{(1-\alpha)}{b'}\left(h\left\langle W_{k,:}, x\right\rangle - p_k\right)^2 \tag{7.10}$$

are SQLC functions for some conditions on f, g, and W. Consequently, we can apply **Theorem 1** to deduce the convergence of the SNGD algorithm for both cost functions. Thus, deducing the convergence algorithm 1.

Theorem 2. *: if we satisfied these conditions:*

- *f admits a maximum value M_f*
- *g admits a maximum value M_g*
- *g is a differentiable function and its derivative admits a minimum value M'_g*
- *g is an $H - Lipschitz$ function*

Then $L1$ is an $(\epsilon, \kappa, z) - SQLC$ function for $\kappa \geq 2HM_gM_f$

Prof: To demonstrate that $L1$ is an SQLC, we prove that $L1$ verifies the second point from **definition 1**. Consider $W \in \mathbb{R}^{d \times d'}$ and $x \in \mathbb{R}^d$ such that $\mathbf{x}_i = g\left\langle W^*_{:,i}, f\left\langle W_1, x\right\rangle\right\rangle$ and $S_i = \left(g\left\langle W_{:,i}, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 \geq \epsilon$, for i arabitrary in $\{1, \ldots, b\}$. b is the batch size. Notice that

$$L1_{\{W_1\}}(x) = \frac{1}{b}\sum_{i=1}^{b}\left(g\left\langle W_{:,i}, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 = \frac{1}{b}\sum_{i=1}^{b} S_i(W_{:,i})$$

According to the results of **lemma 1**, demonstrating that S_i is SQLC for all $i \in \{1, ..., b\}$ is sufficient to deduce that $L1$ also is SQLC.
Let i arbitrary from $\{1, ..., b\}$. Also let v be a point ϵ/κ-close to minima W^*. Note that W_1 is fixed. Then, we need to demonstrate one of the following equivalences:

$$\langle \nabla S_i(w), v - w\rangle \leq 0 \Longleftrightarrow -\langle \nabla S_i(w), w - v\rangle \leq 0 \Longleftrightarrow \langle \nabla S_i(w), w - v\rangle \geq 0$$

Note that w and v should be a vector of $\mathbb{R}^d$. We define S_i as follow:

$$S_i(w) = \left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2$$

$$
\begin{aligned}
&\langle \nabla S_i(w), w - v \rangle \\
&= 2\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle \left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle\right)', (w - v)\right\rangle \\
&= 2\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right) g'(\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle \left\langle \right) f\left\langle W_1, x\right\rangle, (w - v)\rangle \\
&= 2\left(g\left\langle w, f\left\langle W_1, \right\rangle\right\rangle - \mathbf{x}_i\right) g'(\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle) \left\langle f\left\langle W_1, x\right\rangle, w - w^* + w^* - v\right\rangle \\
&\geq 2g'(\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle) \\
&[H^{-1}\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 + \left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle f\left\langle W_1, x\right\rangle, w^* - v\right\rangle] \\
&\geq 2M_g'[H^{-1}\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 + \left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle f\left\langle W_1, x\right\rangle, w^* - v\right\rangle] \\
&\geq 2M_g'[H^{-1}\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 - |g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i|\,||f\left\langle W_1, x\right\rangle||||w^* - v||] \\
&\geq 2M_g'[H^{-1}\epsilon - 2M_g M_f||w^* - v||] \\
&\geq 2M_g'[H^{-1}\epsilon - 2M_g M_f \epsilon/\kappa]
\end{aligned}
\tag{7.11}
$$

Thus

$$
\begin{aligned}
&\langle \nabla S_i(w), w - v \rangle \geq 0 \\
&\iff 2M_g'[H^{-1}\epsilon - 2M_g M_f \epsilon/\kappa] \geq 0 \\
&\iff 2M_g M_f \epsilon/\kappa \leq H^{-1}\epsilon \\
&\iff \kappa \geq 2HM_g M_f
\end{aligned}
\tag{7.12}
$$

Finally, S_i is an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq 2HM_gM_f$, and i arbitrary from $\{1, ..., b\}$. According to **lemma 1** and what we discussed at the beginning of the proof, $L1$ is also an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq 2HM_gM_f$

Proof Explanation:

- In lines 1, 2, and 3 we calculated the derivative and added w^*
- In line 4, we used the fact the g is $H - Lipschitz$. i.e

  $$\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle f\left\langle W_1, x\right\rangle, w - w^*\right\rangle \geq H^{-1}\left(g\left\langle w, f\left\langle W_1, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2$$

  where $\mathbf{x}_i = g\left\langle w^*, f\left\langle W_1, x\right\rangle\right\rangle$
- In line 5, 7, 8 we use the conditions of f, g', and g cited in the **Theorem 2**. Finally, $||w^* - v|| \leq \epsilon/\kappa$

Theorem 3. *: if these conditions were satisfied :*

- *f admit a maximum value M_f*
- *g admit a maximum value M_g*
- *g is a differentiable function and its derivative admits a minimum value m_g*
- *f is a differentiable function and its derivative admits a minimum value m_f*
- *g is an $H-$ Lipschitz function*
- *f is an $H'-$ Lipschitz function*

Then L2 is an SQLC function for $\kappa \geq \max((1-\alpha)2M_fBH', \frac{2\alpha M_g BHH'}{C})$

Prof: To demonstrate that $L2$ is an SQLC, we prove that $L2$ verifies the second point from definition 1 at every $x \in \mathbb{B}_d(0,B)$. Consider $\mathbf{x}_i = g\left\langle W^2_{:,i}, f\left\langle W^*, x\right\rangle\right\rangle$, $||W_2|| \leq C$, $||\mathbf{x}|| \leq B$ such that $R_i = \left(g\left\langle W^2_{:,i}, f\left\langle W, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 \geq \epsilon$, and $\mathbf{p}_k = h\left\langle W^*_{k,:}, x\right\rangle$ such that $P_k = \left(h\left\langle W_{k,:}, x\right\rangle - \mathbf{p}_k\right)^2 \geq \epsilon$ for i arabitrary in $\{1,\ldots,b\}$ and k arabitrary in $\{1,\ldots,d'\}$. b is the batch size. Also let v be a point ϵ/κ-close to minima w^*. W_2 is fixed.
Notice that

$$\begin{aligned} L2_{\{W_1\}}(x) &= \sum_{i=1}^{b} \frac{\alpha}{b}\left(g\left\langle W^2_{:,i}, f\left\langle W, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 + \sum_{k=1}^{b'} \frac{(1-\alpha)}{b'}\left(h\left\langle W_{k,:}, x\right\rangle - \mathbf{p}_k\right)^2 \\ &= \frac{\alpha}{b}\sum_{i=1}^{b} R_i(W) + \frac{1-\alpha}{b'}\sum_{k=1}^{b'} P_k(W_{k,:}) \end{aligned} \tag{7.13}$$

where

$$R_i(w) = \left(g\left\langle W^2_{:,i}, f\left\langle w, x\right\rangle\right\rangle - \mathbf{x}_i\right)^2$$

and

$$P_k(w) = \left(h\left\langle w, x\right\rangle - \mathbf{p}_k\right)^2$$

According to the results of lemma 1, demonstrating that R_i and P_k are SQLC for all $i \in \{1,...,b\}$ and k arbitrary in $\{1,\ldots,d'\}$ is sufficient to deduce that $L2$ also is SQLC. Let i arbitrary from $\{1,...,b\}$ and k arbitrary in $\{1,\ldots,d'\}$. Then, we need to demonstrate one of the following equivalence:

$$\langle \nabla R_i(w), v-w\rangle \leq 0 \Longleftrightarrow -\langle \nabla R_i(w), w-v\rangle \leq 0 \Longleftrightarrow \langle \nabla R_i(w), w-v\rangle \geq 0$$

Same also for P_k

$$\langle \nabla P_k(w), v-w\rangle \leq 0 \Longleftrightarrow -\langle \nabla P_k(w), w-v\rangle \leq 0 \Longleftrightarrow \langle \nabla P_k(w), w-v\rangle \geq 0$$

Note that w and v should be a matrix of $\mathbb{R}^{d\times d'}$.

$$
\begin{aligned}
&\langle \nabla R_i(w), w-v\rangle \\
&= 2\left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle \left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)', w-v\right\rangle \\
&= 2\left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right) g'(\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i)\left\langle \left\langle W_{:,i}^2, f(\langle w,x\rangle)'\right\rangle, w-v\right\rangle \\
&= 2\left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle W_{:,i}^2, f'(\langle w,x\rangle)\right\rangle g'(\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i)\langle x, w-v\rangle \\
&= 2\left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)\left\langle W_{:,i}^2, f'(\langle w,x\rangle)\right\rangle g'(\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i)[\langle x, w-w^*\rangle + \langle x, w^*-v\rangle] \\
&\geq 2\left\langle W_{:,i}^2, f'(\langle w,x\rangle)\right\rangle g'(\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i) \\
&[H^{-1}H'^{-1}||W_{:,i}^2||^{-1}\left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)^2 + \left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)\langle x, w^*-v\rangle] \\
&\geq 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}||W_{:,i}^2||^{-1}\epsilon + \left(g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right)\langle x, w^*-v\rangle] \\
&\geq 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}||W_{:,i}^2||^{-1}\epsilon - \left|g\left\langle W_{:,i}^2, f\left\langle w,x\right\rangle\right\rangle - \mathbf{x}_i\right| ||x||||w^*-v||] \\
&\geq 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}||W_{:,i}^2||^{-1}\epsilon - 2M_g B\epsilon/\kappa] \\
&\geq 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}||W_{:,i}^2||^{-1}\epsilon - 2M_g B\epsilon/\kappa] \\
&\geq 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}C\epsilon - 2M_g B\epsilon/\kappa]
\end{aligned}
\tag{7.14}
$$

Thus

$$
\begin{aligned}
&\langle \nabla R_i(w), w-v\rangle \geq 0 \\
&\iff 2\min_i(W_{:,i}^2) m_f m_g [H^{-1}H'^{-1}C\epsilon - 2M_g B\epsilon/\kappa] \geq 0 \\
&\iff 2M_g B\epsilon/\kappa \leq H^{-1}H'^{-1}C\epsilon \\
&\iff \kappa \geq \frac{2M_g BHH'}{C}
\end{aligned}
\tag{7.15}
$$

Finally, R_i is an $(\epsilon,\kappa,z) - SQLC$ for $\kappa \geq \frac{2M_g BHH'}{C}$, and i arbitrary from $\{1,...,b\}$. According to **Lemma 1** and what we discuss at the beginning of the proof, $\frac{\alpha}{b}\sum_{i=1}^{b} R_i$ is also an $(\epsilon,\kappa,z) - SQLC$ for $\kappa \geq \frac{2\alpha M_g BHH'}{C}$

Proof Explanation:

- In lines 1, 2, 3, and 4 we calculated the derivative and added w^*
- In line 5, we used the fact the g is $H - Lipschitz$ and f is $H' - Lipschitz$. i.e

$$
\begin{aligned}
&|g\left\langle W_2, f\left\langle w,x\right\rangle\right\rangle - g\left\langle W_2, f\left\langle w^*,x\right\rangle\right\rangle| \\
&\leq H|\left\langle W_2, f\left\langle w,x\right\rangle\right\rangle - \left\langle W_2, f\left\langle w^*,x\right\rangle\right\rangle| \\
&\leq H||W_2|||f\left\langle w,x\right\rangle - f\left\langle w^*,x\right\rangle| \\
&\leq HH'||W_2|||\left\langle w,x\right\rangle - \left\langle w^*,x\right\rangle|
\end{aligned}
\tag{7.16}
$$

- In line 5, 7, 8 f, g', and g admit m, M as minimum and M_g as maximum, respectively. Finally, $||w^* - v|| \leq \epsilon/\kappa$, $||W_2|| \leq C$ and $||\mathbf{x}|| \leq B$

$$
\begin{aligned}
&\langle \nabla P_k(w), w - v \rangle = \\
&= 2\left(f\left\langle w, x\right\rangle - p_k\right)\left\langle \left(f\left\langle w, x\right\rangle - p_k\right)', w - v\right\rangle \\
&= 2\left(f\left\langle w, x\right\rangle - p_k\right)\left\langle \left(f\left\langle w, x\right\rangle - p_k\right)', w - v\right\rangle \\
&= 2\left(f\left\langle w, x\right\rangle - p_k\right) f'\left\langle w, x\right\rangle \left\langle x, w - v\right\rangle \\
&= 2\left(f\left\langle w, x\right\rangle - p_k\right) f'\left\langle w, x\right\rangle \left[\left\langle x, w - w^*\right\rangle + \left\langle x, w^* - v\right\rangle\right] \\
&\geq 2f'\left\langle w, x\right\rangle \left[H'^{-1}\left(f\left\langle w, x\right\rangle - p_k\right)^2 + \left(f\left\langle w, x\right\rangle - p_k\right)\left\langle x, w^* - v\right\rangle\right] \\
&\geq 2m_f[H'^{-1}\epsilon - |f\left\langle w, x\right\rangle - p_k|\,||x||||w^* - v||] \\
&\geq 2m_f[H'^{-1}\epsilon - 2M_f B\epsilon/\kappa]
\end{aligned}
\tag{7.17}
$$

Thus

$$
\begin{aligned}
&\langle \nabla P_k(w), w - v \rangle \geq 0 \\
&\iff 2m_f[H'^{-1}\epsilon - 2M_f B\epsilon/\kappa] \geq 0 \\
&\iff 2M_f B\epsilon/\kappa \leq H'^{-1}\epsilon \\
&\iff \kappa \geq 2M_f BH'
\end{aligned}
\tag{7.18}
$$

Finally, P_k is an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq 2M_f BH'$, and k arbitrary from $\{1, ..., b'\}$. According to **Lemma 1** and what we discuss at the beginning of the proof, $\frac{1-\alpha}{b'}\sum_{k=1}^{b'} P_k$ is also an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq (1-\alpha)2M_f BH'$
Proof explanation:

- In lines 1, 2, 3, and 4 we calculated the derivative and added w^*
- In line 5, we used the fact the f is $H' - Lipschitz$.
- In line 5, 7, 8 f' and admit m_f as minimum and f admit M_f as maximum value. Finally, $||w^* - v|| \leq \epsilon/\kappa$

Proof Conclusion: We have demonstrated:

- $\frac{\alpha}{b}\sum_{i=1}^{b} R_i$ is an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq \frac{2\alpha M_g BHH'}{C}$
- $\frac{1-\alpha}{b'}\sum_{k=1}^{b'} P_k$ is an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq (1-\alpha)2M_f BH'$

- $L2$ is also an $(\epsilon, \kappa, z) - SQLC$ for $\kappa \geq \max((1-\alpha)2M_f BH', \frac{2\alpha M_g BHH'}{C})$

Remark 2. *$L1$ and $L2$ are $(\epsilon, \kappa, z) - SQLC$ for*

$$\kappa \geq \max(\max((1-\alpha)2M_f BH', \frac{2\alpha M_g BHH'}{C}), 2HM_g M_f)$$

Remark 3. *The conditions of **Theorem 2** are included in **Theorem 3***

Remark 4. *We connect both **Theorem 2** and **Theorem 3** with **Theorem 1** to prove the convergence of the SNGD for our algorithms 3 and 4 and thus the convergence for algorithm 1*

Remark 5. *Sigmoid function satisfies the conditions of **Theorem 3** as mentioned in the article [225] (proof of lemma 3.1). Moreover, the generalized relu function also satisfies those conditions as mentioned in article [226] (poof of theorem 3.4). The function $f(x)$, where $0 < a < b$ and $a, b \in \mathbb{R}$, is a generalized ReLU function defined as:*

$$f(x) = \begin{cases} ax & x \leq 0 \\ bx & x > 0 \end{cases}$$

Remark 6. *It should be noted that activation functions such as tanh and leaky ReLU are different versions of the aforementioned functions, namely sigmoid and ReLU, respectively.*

Algorithm 3 Algorithm to update W_2 with SNGD

Input: Number of iterations T, training data $S = \{\mathbf{x}_i\}_{i=1}^b$, $\mathbf{x}_i \in \mathbb{R}^3$, learning rate η, minibatch size b, Initialization of parameters W^0, Obtain value of W_1 at each iteration i.e $W_1^0, W_1^1, ..., W_1^t$.

for t=1 ... T **do**

Select a random mini-batch of training points by sampling $\{\mathbf{x}_i\}_{i=1}^b \sim \text{Uniform}(S)$

$A_t(W^t) = \frac{1}{b}\sum_{i=1}^b \left(g\left\langle W_{:,i}^t, f\left\langle W_1^{t-1}, S\right\rangle\right\rangle - \mathbf{x}_i\right)^2$

Update:

$W^{t+1} = W^t - \eta \hat{B}_t$ where $B_t = \nabla A_t\left(W^t\right), \hat{B}_t = \frac{B_t}{\|B_t\|}$

end for

Return: $W_T^* = \arg\min_{\{W^1,...,W^T\}} A_t\left(W^t\right)$

7.4 Automatic Localization (AutoLocal) Architecture

Our AutoLocal architecture is based on an autoencoding neural network with a special extension that makes sense of the latent space results. The main goal of this architecture

Algorithm 4 Algorithm to update W_1 with SNGD

Input: Number of iterations T, training data $S = \{\mathbf{x}_i\}_{i=1}^{b}$, $\mathbf{x}_i \in \mathbb{R}^3$, and $P = \{p_i\}_{i=1}^{b'}$, $p_i \in \mathbb{R}^2$, learning rate η, minibatch size b and b', Initialization of parameters W^0, Obtain value of W_2 at each iteration i.e $W_2^0, W_2^1, ..., W_2^t$.

for t=1 ... T **do**

Select a random mini-batch of training points by sampling $\{\mathbf{x}_i\}_{i=1}^{b} \sim \text{Uniform}(S)$ and $\{p_k\}_{k=1}^{b'} \sim \text{Uniform}(P)$

$A_t(W^t) = \frac{1}{b}\sum_{i=1}^{b} \alpha \left(g\left\langle {}^{t-1}W_{:,i}^2, f\left\langle W^t, S\right\rangle\right\rangle - \mathbf{x}_i\right)^2 + \frac{1}{b'}\sum_{i=k}^{b'}(1-\alpha)\left(h\left\langle W_{k,:}^t, S\right\rangle - p_k\right)^2$

Update:

$W^{t+1} = W^t - \eta\hat{B}_t$ where $B_t = \nabla A_t\left(W^t\right), \hat{B}_t = \frac{B_t}{\|B_t\|}$

end for

Return: $W_T^* = \arg\min_{\{W^1,...,W^T\}} A_t\left(W^t\right)$

is to teach the model to learn vehicle positions and use this knowledge to make accurate predictions for unseen data. To achieve this, we modified the autoencoder objective function to cover the latent space learning and show that the algorithm converges to the minimum (as shown in the last section). In this work, we use the LiDAR measurement as input to the autoencoder, as it provides an accurate representation of the environment. However, in order not to overload the data model, we use a Fuzzy K-means algorithm to cluster all the points, reducing the information to a few central clusters. We have chosen to extract five cluster features from each scene of LiDAR as it was used in [43], [45], [44], [222]. Also, we have also incorporated a sixth feature, which is the mean of the cluster. This decision was based on the findings of the article [222], which demonstrated that the mean of the cluster is a crucial feature that contributes to the accurate estimation of the vehicle position. To further optimize our autolocal architecture, we have limited the range of the LiDAR measurement data considered for each scene. Specifically, we only take into account the points in the range of -10 and 10 for both the x and y coordinates. By doing so, we are able to reduce the computational load on the model, while still retaining enough data to make accurate predictions about the vehicle's position. This approach also helps to filter out any noise or irrelevant data points that may be present in the full LiDAR measurement data, enabling the model to focus solely on the data that is most relevant to the task.

The learning process explained above is illustrated in Figure 7.3; The model learns the positions of the vehicle (depicted in red) by triangulating the blue features. The reliability

of LiDAR measurements allows the blue features to accurately represent various objects in the environment, including moving vehicles and pedestrians, making it a robust method for feature extraction. Additionally, the green feature, which represents the mean of all the blue features, significantly aids the learning process. This is due to its proximity to the actual position, as compared to the individual blue features, providing further justification for the results presented in [222].

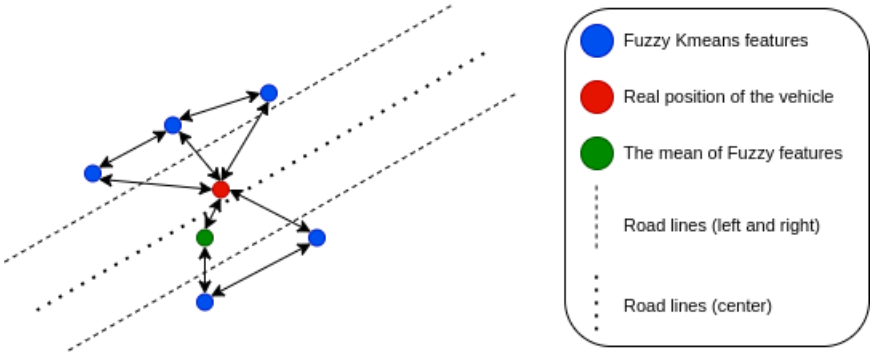


Figure 7.3: An illustration of the learning process inside the model

Practically and as illustrated in Figure 7.2, we have 6 neurons in the input and output of the model where each neuron is a 3d vector coordinates (cluster). The latent space result is a 2d vector coordinates representing the vehicle position in real word coordinates system. We extract the actual vehicle positions from open datasets such as Kitti [187] and NCLT [190]. We use the NCLT data for training and validation since they contain long trajectories, and the Kitti dataset to test the model performance on unseen data. With this approach, we expect the AutoLocal architecture to provide reliable predictions, enabling autonomous vehicles to navigate safely and accurately.

Table 7.1: Training Sequences Summary - Frame Count and Trajectory Length

Sequence	**Dataset**	**Date**	**Frames Num**	**Kilometer**
00	NCLT	2012-01-08	28127	6.4
25	NCLT	2013-01-10	5120	1.1
25	NCLT	2013-04-05	20901	4.5
25	NCLT	2012-12-01	23016	5
25	NCLT	2012-06-15	16545	4.1
25	NCLT	2012-04-29	12971	3.1
SUM	–	–	130186	–

7.5 Experiments

This section details the experiments performed to assess the trained auto-encoders performance using the NCLT dataset. The auto-encoder has six input neurons representing the cluster centers and the mean of all the clusters and a hidden layer with one neuron for position determination. The experiments were conducted on a personal computer with an i7 processor and 20 GB of RAM, and a constant learning rate of 0.0001 was employed throughout the training process. The performance of the model was evaluated using the mean absolute error in meters as the metric.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |p_i - \hat{p}_i|}{n}$$

$$\text{MAE} = \text{mean absolute error}$$
$$\hat{p}_i = \text{predicted position}$$
$$p_i = \text{true position}$$
$$n = \text{total number of data points}$$

We evaluate the model in terms of three aspects: reconstruction loss, position loss, and overall loss. Position loss calculates the error between the position estimated from the latent space and the actual position. Reconstruction loss measures the difference between the input and output data reconstructed by the model. Finally, the overall loss is the reconstruction loss multiplied by α penalty parameter plus the position loss $1-\alpha$ penalty parameter. We fixed the number of epochs to 3000, since the number of training data is large and the model is quite complex.

We trained the autoencoder model with different penalty values to study their effect on the model's performance. Specifically, we trained the model with penalty values of $\alpha = 0.9$, $\alpha = 0.5$, $\alpha = 0.1$, and $\alpha = 0.01$.

To evaluate how well the model generalizes to new data, we conducted a test on the Kitti dataset, using long-term trajectories that were not seen during training. We compared the results obtained by our model, which uses a single hidden layer with one neuron for position determination, with those of other advanced methods such as LOAM [200], LOAM Velodyne [201], LeGO-LOAM [202], A-LOAM [203], SGLO [205], LO-Net and LO-Net-M [204].

The evaluation results are presented and analyzed in detail. We examine the perfor-

mance of the autoencoder model for different penalty values and compare it with state-of-the-art methods. The results provide valuable information about the performance of the autoencoder model and its potential applications in different fields.

7.5.1 The choice of penalty parameter α:

Where an additional hyperparameter is introduced, denoted by α, which controls the learning of the latent space, by choosing different α values in order to make as priority the learning of the latent space rather than learning the reconstruction of the centers' clusters which can impact the overall performance of the model. This research article investigates the impact of different values of α on the performance of an autoencoder. Multiple experiments were conducted to study this relationship, employing the Bayesian optimization algorithm for parameter estimation to determine the optimal value of α. In order to accurately estimate α, a strategic approach was employed due to the random selection nature of the algorithm. Initially, the algorithm was executed on the entire interval [0,1] to identify the best sub interval, such as [0,0.5] or [0.5,1], using the training data. The parameters of the algorithm was kept as default.

As shown in Figure 7.4 (bottom image), the overall loss decreased for smaller values of α (< 0.5). The best estimated value of α was found to be 0.02, resulting in an overall loss of 0.93. However, due to the algorithm's random selection, the estimation may not be precise. Consequently, the algorithm was rerun on the [0,0.5] interval, yielding an overall loss of 0.92 with $\alpha = 0.0103$. See Figure 7.4 (top - left). To ensure the exploration of the best α in the other interval, the algorithm was further rerun on the [0.5,1] interval, leading to an overall loss of 2.4 with $\alpha = 0.5$. See Figure 7.4 (top - right). In conclusion, the optimal value of α representing the overall loss was determined to be $\alpha = 0.01$.

Upon analyzing the Table 7.2, an important observation can be made: regardless of the varying values of α, the reconstruction loss remains stable. This might lead one to believe that it can be disregarded, and only the position loss should be considered. However, such a notion is incorrect since the reconstruction phase plays a crucial role in ensuring a resilient projection of the 3D coordinate clusters to 2D coordinate positions. It should be noted that the best losses for the model was achieved when α was set to 0.01. The evolution of the three losses across the epochs for $\alpha = 0.01$ is depicted in Figure 7.5

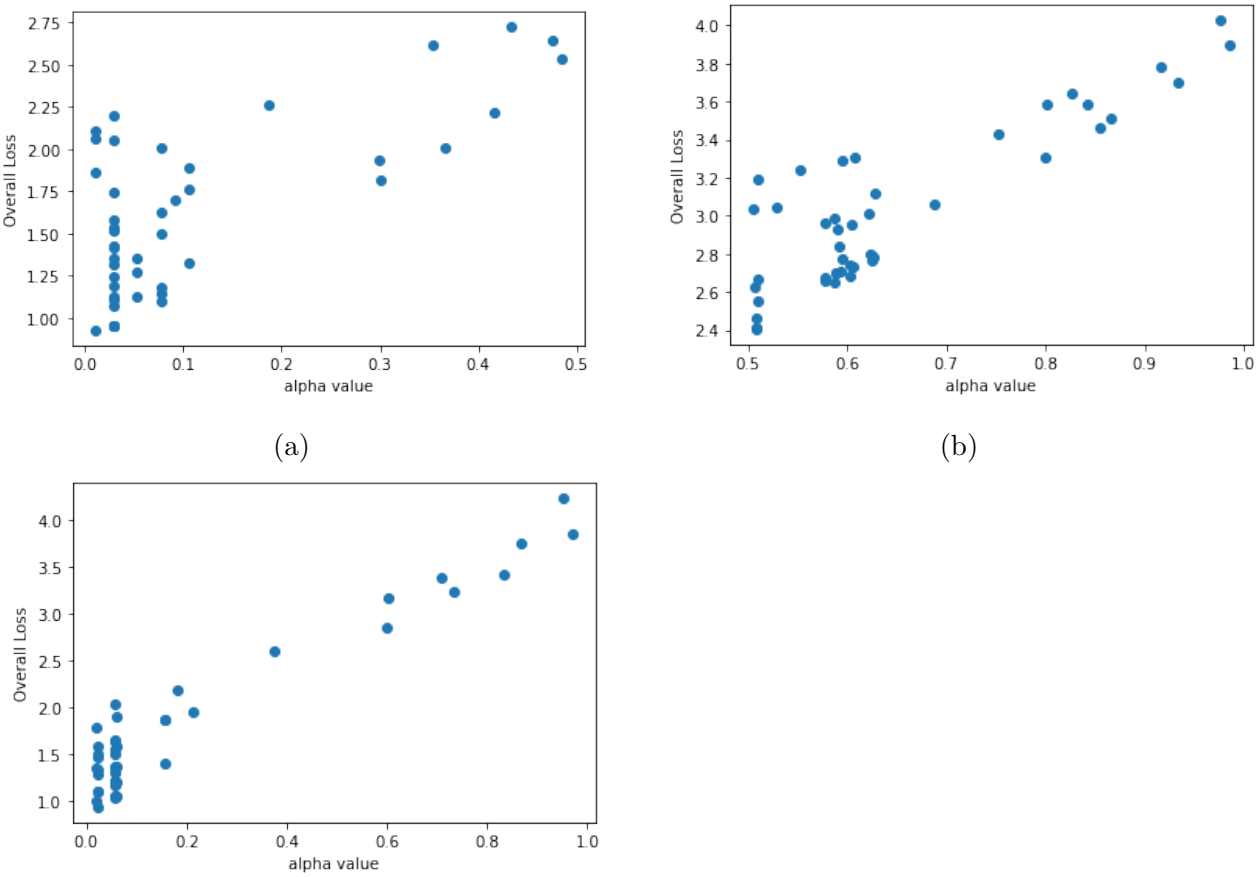


Figure 7.4: The results obtained from applying the Bayesian optimization algorithm on three distinct intervals, namely the interval [0,1] (c), the interval [0,0.5] (a), and the interval [0.5,1] (b), are presented.

Table 7.2: The performance of the model for different values.

Loss	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.9$
Reconstruction loss	**3.9**	3.9	4.1	4
Positioning loss	**0.88**	0.9	1.3	1
Overall loss	**0.91**	1.2	2.7	3.7

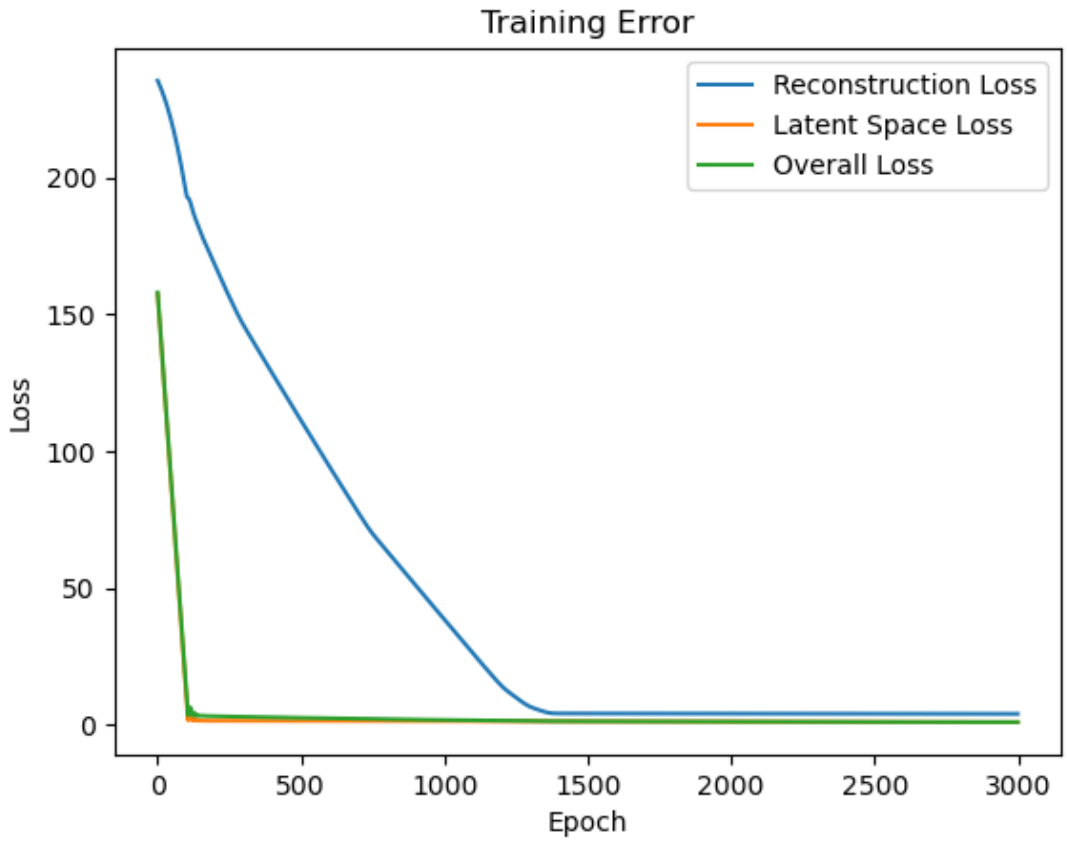


Figure 7.5: Graphical representation of training loss values over epochs.

7.5.2 Discussion and comparison

In recent years, many state-of-the-art methods have been proposed to estimate the positions of moving vehicles using LiDAR sensors. These methods mainly rely on registration techniques to align the LiDAR point clouds and estimate the positions. In this study, we compare our proposed method with several of these state-of-the-art methods, including LOAM [200], LOAM Velodyne [201], LeGO-LOAM [202], A-LOAM [203], SGLO [205], LO-Net and LO-Net-M [204]. These methods have shown promising results in various environments and scenarios. Therefore, we evaluate our method against them to demon-

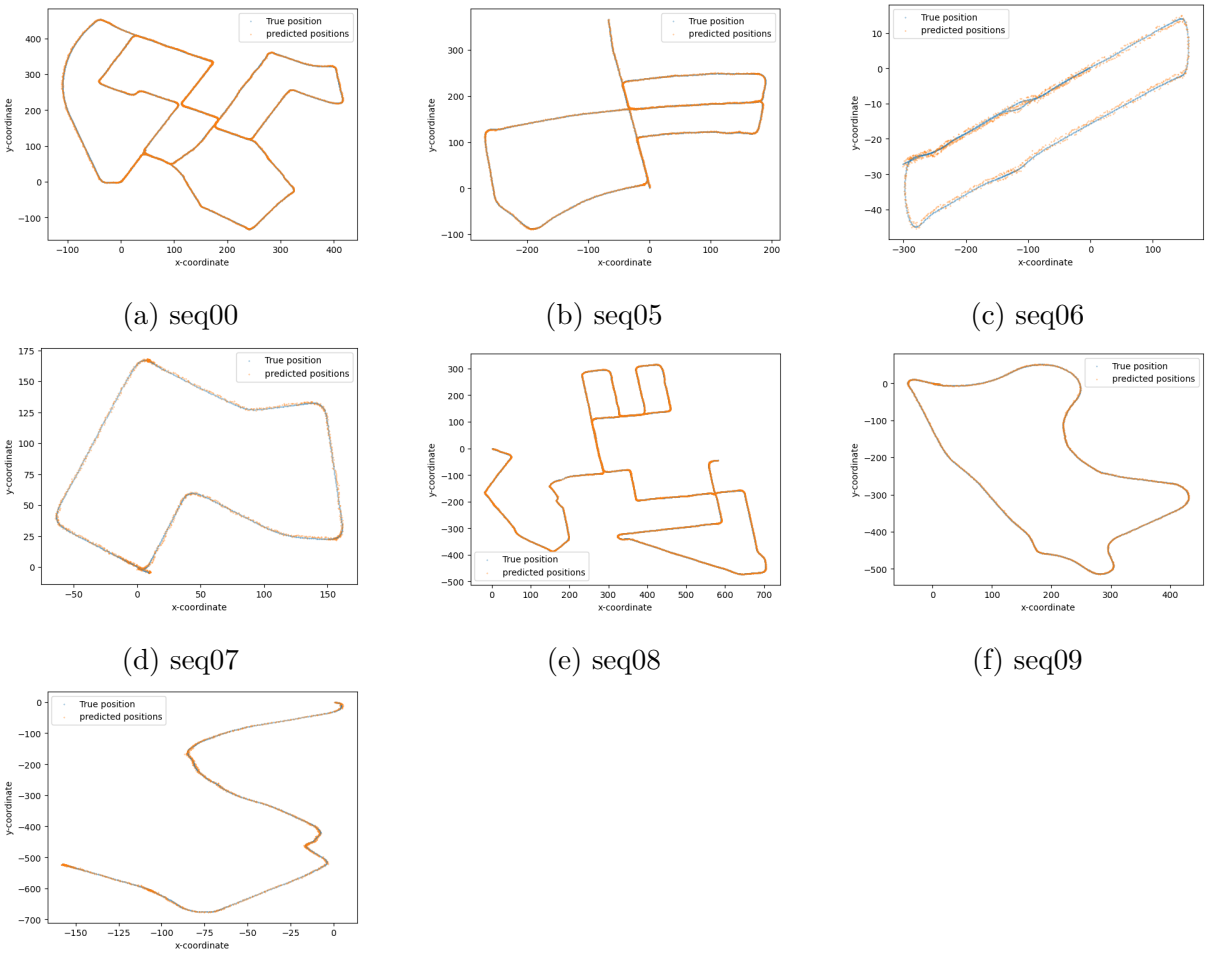
strate the effectiveness and robustness of each approach.

Table 7.3: Accuracy evaluation of our method with state-of-art registration methods

Seq	LOAM [200]	LOAM Velodyne [201]	LeGO- LOAM [202]	A- LOAM [203]	SGLO [205]	LO-Net [204]	LO-Net-M [204]	Ours
00	0.78	3.410	1.347	0.791	0.775	1.47	0.78	**0.54**
05	0.57	1.318	0.887	**0.504**	0.596	1.04	0.62	0.52
06	0.65	1.007	0.87	0.615	0.483	0.71	0.55	**0.47**
07	0.63	1.256	0.77	**0.453**	0.515	1.70	0.56	0.55
08	1.12	2.159	1.337	1.107	1.059	2.12	1.08	**0.53**
09	0.77	1.437	1.282	0.739	0.733	1.37	0.77	**0.52**
10	0.79	1.912	1.675	1.006	0.977	1.80	0.92	**0.49**

Table 7.3 presents the accuracy evaluation of our method compared to the state-of-the-art registration methods on the Kitti dataset. The evaluation is conducted based on the mean absolute error in meters.

In terms of the evaluation results, our method outperforms most of the compared methods in almost all tested sequences, achieving the lowest mean absolute error. Specifically, in sequences 00, 06, 08, 09, and 10, our method achieved the lowest mean absolute error among all compared methods, demonstrating the effectiveness of our proposed method. Additionally, in sequences 05 and 07, our method performed competitively with the best-performing method. These results suggest that our proposed method can accurately estimate the position and outperform the state-of-the-art registration methods in most tested scenarios.



(a) seq00 (b) seq05 (c) seq06

(d) seq07 (e) seq08 (f) seq09

(g) seq10

Figure 7.6: Graphical representation of the model prediction (in orange) comparing with the real position in (in blue)

The registration methods mentioned in the table rely on close information to the ground truth positions, such as IMU measurements and other sensors. In contrast, our method only uses LiDAR points to predict the positions, making it a more lightweight solution. Moreover, the registration techniques are often known for their high computational cost, whereas our method can make direct predictions without the need for additional computation. Figure 7.6 present the graphical results of predicting vehicle trajectories (orange color) compared to the ground-truth positions (blue color).

7.6 Conclusion:

In this work, we proposed an extension of the autoencoder model that allows for the prediction of the vehicle's position from the LiDAR point cloud. We achieve this by modifying the main objective function of the autoencoder and adding another objective function that is responsible for the encoder layer predicting better the positions. Additionally, we introduce a penalty parameter that is multiplied by the objective functions in order to emphasize the learning of the latent space. This allows us to train our model to learn the relationships between the input data and the vehicle's position in a more effective manner.

To demonstrate the convergence of the new objective function, we used the Stochastic Natural Gradient Descent (SNGD) algorithm and explored its convergence to the minimum for the Stochstic-local-quasi-convex (SQLC) functions. We show that the new objective function is an (SQLC) function, which facilitates the convergence analysis.

We trained our model using the NCLT dataset, which contains a large number of sequential trajectories and is well-suited for this task. Our model was implemented using PyTorch, and we used the Adam optimizer to update the network weights during training.

We investigated the performance of the model for different values of α penalty parameter. We found that $\alpha = 0.01$ is the best choice to train our model with a minimum loss function.

We evaluated our method on the KITTI dataset and compared it with several state-of-the-art registration techniques, including LOAM, LOAM Velodyne, LeGO-LOAM, A-LOAM, SGLO, LO-Net, and LO-Net-M. Our method achieved the best performance in terms of mean absolute error for most of the KITTI sequences, outperforming all the other methods.

One important advantage of our method is that it does not require any additional sensors, such as IMUs or GPS, to predict the positions of the vehicle. This is in contrast to the registration techniques that rely on these sensors to provide additional information

about the motion of the vehicle. Moreover, our method is computationally efficient, as it can make direct predictions without the need for iterative optimization, which is a common step in registration techniques.

Overall, our method presents a promising solution for autonomous driving, as it offers an accurate and computationally efficient position prediction using only raw LiDAR data. However, our method registers some perturbation to predict some part of the trajectory, especially in a roundabout or when the vehicle turns sharply, which needs more investigation to be done in our future work.

Part IV

Conclusions

CHAPTER 8

CONCLUSION AND FUTURE DIRECTIONS

8.1 Conclusions

This thesis investigate the problem of localization and mapping in self driving vehicles especially, the defects of GNSS systems to achieve the task in some particular environments that cut the signal between the satellites and the vehicles.

By conducting a comprehensive literature review, we have explored various methods and aspects in this field. Through the adoption of an architecture comprising feature extraction, mapping, and localization, we have analyzed these methods, identifying their strengths and weaknesses (**contribution 1 (Literature review)**). Based on that, our proposed Lidar-based approach for feature extraction has shown promise results in enhancing the speed and accuracy of the localization and mapping process (**contribution 2**).

We have presented two distinct approaches for vehicle localization: stochastic and optimization. In the stochastic approach, our modified particle filter algorithm and the selection of specific points for weight updating have demonstrated fast and accurate vehicle localization. Furthermore, by selecting only the closest particles to the real positions and regenerating particles around them, we have achieved improved control and sustainability (**contribution 3, 4**). Within the optimization approach, we have introduced a deep learning model that predicts vehicle positions based on Lidar measurements. By utilizing Explainability AI techniques, we have gained insights into the model's predictions

and provided detailed analysis. Additionally, our employment of a modified autoencoder model, guided by a specific loss function, has allowed for effective position prediction by leveraging the projection and position losses of Lidar points (**contribution 5, 6**).

8.2 Challenges and future work

In this section we provide the challenges and and future direction of features extraction, mapping, and localization since we believe that any method should use this architecture.

In addition, our enquiry results showcase the efficacy of our methods. We're pleased to denote that our set about has exceeded the existing posit of the art crosswise a wide straddle of sequence datasets, including Kitti and NCLT, further underlining the robustness of our planned techniques in real-world scenarios.

Looking ahead, the integration of features extraction, mapping, and localization is essential for any method acting operating inside this architecture. As we advance, we acknowledge that addressing challenges in each of these aspects and charting the path for future search will collectively put up to the evolution of effective self-driving vehicle technologies.

8.2.1 Features extraction

In order to localize itself within the environment, the vehicle needs to explore the received information from the sensors. However, the huge amount of data received makes impossible to be used in real-time vehicle execution since the vehicle needs an instant interaction with the environment, e.g. accelerating, breaking, steering the wheel. That is why the on-board systems need effective feature extraction methods that will distinguish relevant features for better execution of the localization process.

After surveying and analyzing related papers, the following challenges were identified:

- Features should be robust against any external effect like weather changes, other moving objects, etc. e.g trees that move due to winds
- Provide the possibility of re-use in other tasks.
- The detection system should be capable to extract features even with a few objects

in the environment.

- The proposed algorithms to extract features should not hurt the system by requiring calculation and execution time.
- One issue that should be taken into consideration is about safe and dangerous features. Each feature must provide a degree of safety (expressed as percentage) which helps to determine the nature of the feature and where they belong to. e.g belonging to the road is safer than being on the walls.

8.2.2 Mapping

- Data storage is a big challenge for the AVs. We need to find the minimum information required to run the localization algorithm.
- Maps are prone to changes frequently, so an update system should take place here to update and maintain the changes. Also, the vehicle connectivity system should support this task and supply the changes for other vehicles.
- Preserving the information on vehicle localization and ensuring privacy.

8.2.3 Localization

We have surveyed in this section the approaches used to solve the problem of localization in self-driving vehicles (even for any autonomous systems in some cases). We have summarized in Table 4 those approaches, applicable sensors, their accuracy in real environment scenarios, and the main challenges that should be taken into account in the upcoming research. Using one of the approaches must ensure fast interaction with other vehicle's tasks, which reduces time and energy consumption for the vehicles and save battery life for the long term. Moreover, we need to ensure the reliability of the methods, and provide options to supply other vehicles with information, which creates the challenge of privacy and security in this application domain. Also, another interesting challenge is the conflict between information received from the sensors and what we got from the nearby connected vehicles. Which one is more likely to provide the right position? An investigation should take place in this area as well.

Table 8.1: Summary of existing approaches to solve the localization problem.

Approach	Sensors	Score	Main Challenges
Dead Reckoning	GPS INS Wheel odometry	Low	Prone to error in case drifting
Triangulation	GNSS	High	Limited in some environment senarios
Motion tracking	LiDAR Cameras	Passable	High cost of time and energy A few objects in the environment mislead the matching process.
Optimization	LiDAR Cameras	High	High computational time to solve the matrix calculation
Probability	LiDAR Cameras	High	Problem of initialization
Cooperative localization	Vehicle connectivity	High	Limitation if there is a long distance between vehicles

BIBLIOGRAPHY

[1] "Road traffic injuries",World Health Organization, June 2021,[Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[2] H. Du, G. Zhu, and J. Zheng, "Why travelers trust and accept self-driving cars: An empirical study," Travel Behav. Soc., vol. 22, no. March 2020, pp. 1–9, 2021, https://doi:10.1016/j.tbs.2020.06.012.

[3] A. Kopestinsky, "25 Astonishing Self-Driving Car Statistics for 2021", PolicyAdvice, 2021, Accessed: jul.02, 2021. [Online]. Available: https://policyadvice.net/insurance/insights/self-driving-car-statistics/

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," IEEE Access, vol. 8, pp. 58443–58469, 2020, https://doi: 10.1109/ACCESS.2020.2983149.

[5] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car-part i: Distributed system architecture and development process," IEEE Trans. Ind. Electron., vol. 61, no. 12, pp. 7131–7140, 2014, https://doi: 10.1109/TIE.2014.2321342.

[6] J. L. Blanco-Claraco, "A tutorial on **SE**(3) transformation parameterizations and on-manifold optimization," no. 3, 2021, [Online]. Available: https://arxiv.org/abs/2103.15980.

[7] H. sjafrie, "Introduction to self-driving vehicle technology", Taylor francis, 2020.

[8] N. Piasco, D. Sidibé, C. Demonceaux, and V. Gouet-Brunet, "A survey on Visual-Based Localization: On the benefit of heterogeneous data," Pattern Recognit., vol. 74, pp. 90–109, 2018, https://doi: 10.1016/j.patcog.2017.09.013.

[9] E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," Rob. Auton. Syst., vol. 64, pp. 1–20, 2015, https://doi: 10.1016/j.robot.2014.11.009.

[10] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," J. F. Robot., vol. 37, no. 3, pp. 362–386, 2020, https://doi: 10.1002/rob.21918.

[11] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, "Deep 3D Object Detection Networks Using LiDAR Data: A Review," IEEE Sens. J., vol. 21, no. 2, pp. 1152–1171, 2021, https://doi: 10.1109/JSEN.2020.3020626.

[12] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," Sensors (Switzerland), vol. 19, no. 19, pp. 1–22, 2019, https://doi: 10.3390/s19194188.

[13] B. Huang, J. Zhao, and J. Liu, "A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks," pp. 1–17, 2019, [Online]. Available: https://arxiv.org/abs/1909.05214.

[14] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," IEEE Trans. Intell. Veh., vol. 2, no. 3, pp. 194–220, 2017, https://doi: 10.1109/TIV.2017.2749181.

[15] L. Xia, J. Cui, R. Shen, X. Xu, Y. Gao, and X. Li, "A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots," Int. J. Adv. Robot. Syst., vol. 17, no. 3, pp. 1–17, 2020, https://doi: 10.1177/1729881420919185.

[16] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006, https://doi: 10.1109/MRA.2006.1638022.

[17] F. B. Günay, E. Öztürk, T. Çavdar, Y. S. Hanay, and A. ur R. Khan, "Vehicular Ad Hoc Network (VANET) Localization Techniques: A Survey", vol. 28, no. 4. Springer Netherlands, 2021, https://doi: 10.1007/s11831-020-09487-1.

[18] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications," IEEE Internet Things J., vol. 5, no. 2, pp. 829–846, 2018, https:// doi: 10.1109/JIOT.2018.2812300.

[19] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," Expert Systems with Applications, vol. 165, p. 113816, 2021.

[20] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," IEEE Open Journal of Intelligent Transportation Systems, vol. 3, pp. 458–488, 2022.

[21] Kim, T. Chung and K. Yi, "Lane map building and localization for automated driving using 2D laser rangefinder," 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 680-685, https:// doi: 10.1109/IVS.2015.7225763.

[22] Jun-Hyuck Im , Sung-Hyuck Im, G.-I. Jee, "Extended Line Map-Based Precise Vehicle Localization Using 3D LIDAR," Sensors, vol. 18, p. 3179, 2018, https://doi: 10.3390/s18103179.

[23] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," in IEEE Transactions on Systems Man and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, https:// doi: 10.1109/TSMC.1979.4310076.

[24] Y. Zhang, J. Wang, X. Wang, C. Li and L. Wang, "A real-time curb detection and tracking method for UGVs by using a 3D-LIDAR sensor," 2015 IEEE Conference on Control Applications (CCA), 2015, pp. 1020-1025, https://doi: 10.1109/CCA.2015.7320746.

[25] L. Wang, Y. Zhang, and J. Wang, "Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR," IFAC-PapersOnLine, vol. 50, no. 1, pp. 276–281, 2017, https:// doi: 10.1016/j.ifacol.2017.08.046.

[26] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks," 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 13-19, https:// doi: 10.1109/IVS.2017.7995692.

[27] J. Kummerle, M. Sons, F. Poggenhans, T. Kuhner, M. Lauer, and C. Stiller, "Accurate and efficient self-localization on roads using basic geometric primitives," Proc. - IEEE Int. Conf. Robot. Autom., vol. 2019-May, pp. 5965–5971, 2019, https://doi: 10.1109/ICRA.2019.8793497.

[28] C. Zhang, M. H. Ang, and D. Rus, "Robust LIDAR Localization for Autonomous Driving in Rain," IEEE Int. Conf. Intell. Robot. Syst., pp. 3409–3415, 2018, https://doi: 10.1109/IROS.2018.8593703.

[29] L. Weng, M. Yang, L. Guo, B. Wang and C. Wang, "Pole-Based Real-Time Localization for Autonomous Driving in Congested Urban Scenarios," 2018 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2018, pp. 96-101, https://doi: 10.1109/RCAR.2018.8621688.

[30] F. Lu, G. Chen, J. Dong, X. Yuan, S. Gu, and A. Knoll, "Pole-based Localization for Autonomous Vehicles in Urban Scenarios Using Local Grid Map-based Method," ICARM 2020 - 2020 5th IEEE Int. Conf. Adv. Robot. Mechatronics, pp. 640–645, 2020, https://doi: 10.1109/ICARM49381.2020.9195330.

[31] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, "Long-term vehicle localization in urban environments based on pole landmarks extracted from 3-D lidar scans," Rob. Auton. Syst., vol. 136, p. 103709, 2021, https://doi: 10.1016/j.robot.2020.103709.

[32] J. Gim, C. Ahn, and H. Peng, "Landmark Attribute Analysis for a High-Precision Landmark-based Local Positioning System," IEEE Access, vol. 9, 2021, https://doi: 10.1109/ACCESS.2021.3053214.

[33] S. Pang, D. Kent, D. Morris and H. Radha, "FLAME: Feature-Likelihood Based Mapping and Localization for Autonomous Vehicles," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 5312-5319, https://doi: 10.1109/IROS40897.2019.8968082.

[34] O. Shipitko, V. Kibalov, and M. Abramov, "Linear Features Observation Model for Autonomous Vehicle Localization," 16th IEEE Int. Conf. Control. Autom. Robot. Vision, ICARCV 2020, pp. 1360–1365, 2020, https://doi: 10.1109/ICARCV50220.2020.9305434.

[35] O. Shipitko and A. Grigoryev, "Ground vehicle localization with particle filter based on simulated road marking image," Proc. - Eur. Counc. Model. Simulation, ECMS, no. Thrun, pp. 341–347, 2018, https://doi: 10.7148/2018-0341.

[36] O. S. Shipitko, M. P. Abramov, and A. S. Lukoyanov, "Edge detection based mobile robot indoor localization." International Conference on Machine Vision. 2018, [Online]. Available: https://www.semanticscholar.org/paper/Edge-detection-based-mobile-robot-indoor-Shipitko-Abramov/51fd6f49579568417dd2a56e4c0348cb1bb91e78

[37] F. Wu, H. Wei, and X. Wang, "Correction of image radial distortion based on division model," Opt. Eng., vol. 56, no. 1, p. 013108, 2017, https://doi: 10.1117/1.oe.56.1.013108.

[38] L. Weng, V. Gouet-Brunet, and B. Soheilian, "Semantic signatures for large-scale visual localization," Multimed. Tools Appl., pp. 1–12, 2020, https://doi: 10.1007/s11042-020-08992-6.

[39] C. Hungar, S. Brakemeier, S. Jürgens, and F. Köster, "GRAIL : A Gradients-of-Intensities-based Local Descriptor for Map-based Localization Using LiDAR Sensors," IEEE Intelligent Transportation Systems Conference, ITSC 2019 (pp. 4398–4403), 2019, https://doi.org/10.1109/ITSC.2019.8917525

[40] C. Hungar, J. Fricke, J. Stefan, and K. Frank, "Detection of Feature Areas for Map-based Localization Using LiDAR Descriptors,",In 16th Workshop on Posit. Navigat. and Communicat. ,2019, https://doi: 10.1109/WPNC47567.2019.8970180.

[41] B. Gu, J. Liu, H. Xiong, T. Li, and Y. Pan, "Ecpc-icp: A 6d vehicle pose estimation method by fusing the roadside lidar point cloud and road feature," Sensors, vol. 21, no. 10, 2021, https://doi: 10.3390/s21103489.

[42] A. Burt, M. Disney, and K. Calders, "Extracting individual trees from lidar point clouds using treeseg," Methods in Ecology and Evolution, 2018.

[43] A. Charroud, A. Yahyaouy, K. E. Moutaouakil, and U. Onyekpe, "Localisation and mapping of self-driving vehicles based on fuzzy K-means clustering: A non-semantic approach," 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), 2022.

[44] A. Charroud, K. El Moutaouakil, A. Yahyaouy, U. Onyekpe, V. Palade, and M. N. Huda, "Rapid localization and mapping method based on adaptive particle filters," Sensors, vol. 22, no. 23, p. 9439, 2022.

[45] A. Charroud, K. E. Moutaouakil, and A. Yahyaouy, "Fast and accurate localization and mapping method for self-driving vehicles based on a modified clustering particle filter," Multimedia Tools and Applications, 2022.

[46] Y. Zou, X. Wang, T. Zhang, B. Liang, J. Song, and H. Liu, "BRoPH: An efficient and compact binary descriptor for 3D point clouds," Pattern Recognit., vol. 76, pp. 522–536, 2018, https://doi: 10.1016/j.patcog.2017.11.029.

[47] L. Kiforenko, B. Drost, F. Tombari, N. Krüger, and A. Glent Buch, "A performance evaluation of point pair features," Comput. Vis. Image Underst., vol. 166, no. November 2016, pp. 66–80, 2018, https://doi: 10.1016/j.cviu.2017.09.004.

[48] K. Berker Logoglu, S. Kalkan, and A. Temizel, "CoSPAIR: Colored Histograms of Spatial Concentric Surflet-Pairs for 3D object recognition," Rob. Auton. Syst., vol. 75, pp. 558–570, 2016, https://doi: 10.1016/j.robot.2015.09.027.

[49] A. G. Buch and D. Kraft, "Local point pair feature histogram for accurate 3D matching," Br. Mach. Vis. Conf. 2018, BMVC 2018, 2019, [Online] Available: https://www.reconcell.eu/files/publications/Buch2018.pdf

[50] H. Zhao, M. Tang, and H. Ding, "HoPPF: A novel local surface descriptor for 3D object recognition," Pattern Recognit., vol. 103, p. 107272, 2020, https://doi: 10.1016/j.patcog.2020.107272.

[51] L. Wu et al., "Pptfh: Robust local descriptor based on point-pair transformation features for 3d surface matching," Sensors, vol. 21, no. 9, pp. 1–21, 2021, https://doi: 10.3390/s21093229.

[52] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao, "TOLDI: An effective and robust approach for 3D local shape description," Pattern Recognit., vol. 65, no. November 2016, pp. 175–187, 2017, https://doi: 10.1016/j.patcog.2016.11.019.

[53] S. M. Prakhya, J. Lin, V. Chandrasekhar, W. Lin, and B. Liu, "3DHoPD: A Fast Low-Dimensional 3-D Descriptor," IEEE Robot. Autom. Lett., vol. 2, no. 3, pp. 1472–1479, 2017, https://doi: 10.1109/LRA.2017.2667721.

[54] Z. Hu, T. Qianwen, and F. Zhang, "Improved intelligent vehicle self- localization with integration of sparse visual map and high-speed pavement visual odometry", 235(1), 177–187., 2021, https://doi: 10.1177/0954407020943306.

[55] Y. Li, Z. Hu, Y. Cai, H. Wu, Z. Li, and M. A. Sotelo, "Visual Map-Based Localization for Intelligent Vehicles from Multi-View Site Matching," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 2, pp. 1068–1079, 2021, https://doi: 10.1109/TITS.2019.2962183.

[56] G. Ge, Y. Zhang, Q. Jiang, and W. Wang, "Visual features assisted robot localization in symmetrical environment using laser slam," Sensors, vol. 21, no. 5, pp. 1–23, 2021, https://doi: 10.3390/s21051772.

[57] DBow3. (2017) DBow3 [Source code]. :https://github.com/rmsalinas/DBow3.

[58] A. Holliday and G. Dudek, "Scale-invariant localization using quasi-semantic object landmarks," Auton. Robots, vol. 45, no. 3, pp. 407–420, 2021, https://doi: 10.1007/s10514-021-09973-w.

[59] "Convolutional neural network", Wikipedia, Accessed: jul. 28, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[60] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," Robot. Sci. Syst., vol. 12, 2016, https://doi: 10.15607/rss.2016.xii.042.

[61] K. Minemura, H. Liau, A. Monrroy, and S. Kato, "LMNet: Real-time multiclass object detection on CPU using 3D LiDAR," Proc. 2018 3rd Asia-Pacific Conf. Intell. Robot Syst. ACIRS 2018, pp. 28–34, 2018, https://doi: 10.1109/ACIRS.2018.8467245.

[62] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 4490–4499, 2018, https://doi: 10.1109/CVPR.2018.00472.

[63] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: A 3D Object Detection Framework from LiDAR Information," IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC, vol. 2018-Novem, pp. 3517–3523, 2018, https://doi: 10.1109/ITSC.2018.8569311.

[64] A. Barrera, C. Guindel, J. Beltrán, and F. García, "BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View," 2020 IEEE 23rd Int. Conf. Intell. Transp. Syst. ITSC 2020, 2020, https://doi: 10.1109/ITSC45102.2020.9294293.

[65] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-YOLO: Real-time 3D Object Detection on Point Clouds," pp. 1–14, 2018, [Online]. Available: https://arxiv.org/abs/1803.06199.

[66] W. Ali, S. Abdelkarim, M. Zidan, M. Zahran, and A. El Sallab, "YOLO3D: End-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11131 LNCS, pp. 716–728, 2019, https://doi: 10.1007/978-3-030-11015-4_54.

[67] G. Te, A. Zheng, W. Hu, and Z. Guo, "RGCNN: Regularized graph Cnn for point cloud segmentation," MM 2018 - Proc. 2018 ACM Multimed. Conf., pp. 746–754, 2018, https://doi: 10.1145/3240508.3240621.

[68] B. S. Hua, M. K. Tran, and S. K. Yeung, "Pointwise Convolutional Neural Networks," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 984–993, 2018, https://doi: 10.1109/CVPR.2018.00109.

[69] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," Adv. Neural Inf. Process. Syst., vol. 2018-Decem, pp. 820–830, 2018. [online] Available https://arxiv:1801.07791

[70] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using geo-cnn," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 998–1008, 2019, https://doi: 10.1109/CVPR.2019.00109.

[71] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph Cnn for learning on point clouds," ACM Trans. Graph., vol. 38, no. 5, 2019, https://doi: 10.1145/3326362.

[72] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11212 LNCS, pp. 90–105, 2018, https://doi: 10.1007/978-3-030-01237-3_6.

[73] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," Robot. Sci. Syst., vol. 11, 2015, https://doi: 10.15607/RSS.2015.XI.035.

[74] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," Proc. - 30th IEEE Conf. Comput. Vis.

Pattern Recognition, CVPR 2017, vol. 2017-January, pp. 77–85, 2017, https://doi: 10.1109/CVPR.2017.16.

[75] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," Adv. Neural Inf. Process. Syst., vol. 2017-Decem, pp. 5100–5109, 2017, [Online] Avaliable: https://arxiv:1706.02413

[76] K. Wong, Y. Gu, and S. Kamijo, "Mapping for autonomous driving: Opportunities and challenges," IEEE Intell. Transp. Syst. Mag., vol. 13, no. 1, pp. 91–106, 2021, https:// doi: 10.1109/MITS.2020.3014152.

[77] H. G. Seif and X. Hu, "Autonomous driving in the iCity—HD maps as a key challenge of the automotive industry," Engineering, vol. 2, no. 2, pp. 159–162, 2016. https://doi: 10.1016/J.ENG.2016.02.010.

[78] G. P. Gwon, W. S. Hur, S. W. Kim, and S. W. Seo, "Generation of a precise and efficient lane-level road map for intelligent vehicle systems," IEEE Trans. Veh. Technol., vol. 66, no. 6, pp. 4517–4533, 2017. https:// doi: 10.1109/ TVT.2016.2535210.

[79] "HD Map for Autonomous Vehicles Market",Markets-and-markets,Jul 2021 ,Accessed: Aug. 15, 2021, [Online].Available https://www.marketsandmarkets.com/Market-Reports/hd-map-autonomous-vehicle-market-141078517.html

[80] S. Hausler, M. Milford, "P1-021: Map creation, monitoring and maintenance for automated driving-Literature Review," 2021, Accessed: sep. 07, 2021. [Online]. Available: https://imoveaustralia.com/wp-content/uploads/2021/01/P1%E2%80%90021-Map-creation-monitoring-and-maintenance-for-automated-driving.pdf

[81] R. Herrtwich, "The evolution of the HERE HD Live Map at Daimler.", 2018, Accessed: sep. 07, 2021. [Online]. Available: https://360.here.com/the-evolution-of-the-hd-live-map.

[82] K. Chellapilla, "Rethinking Maps for Self-Driving", Medium, 2018, Accessed: sep. 06, 2021. [Online]. Available: https://medium.com/wovenplanetlevel5/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6

[83] H. Vardhan, "HD Maps: New age maps powering autonomous vehicles", Geo spatial word, 2017, Accessed: sep. 07, 2021. [Online]. Available:https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/

[84] T. Dahlström, "How Accurate Are HD Maps for Autonomous Driving and ADAS Simulation?", 2020, Accessed: sep. 07, 2021. [Online]. Available: https://atlatec.de/en/blog/how-accurate-are-hd-maps-for-autonomous-driving-and-adas-simulation/

[85] M. Schreier, "Environment representations for automated on-road vehicles," At-Automatisierungstechnik, vol. 66, no. 2, pp. 107–118, 2018, https://doi: 10.1515/auto-2017-0104.

[86] S. Thrun, "Probabilistic robotics," Commun. ACM, vol. 45, no. 3, pp. 52–57, 2002, https://doi: 10.1145/504729.504754.

[87] J. Elfring, E. Torta and R. van de Molengraft, "Particle Filters: A Hands-On Tutorial", Sensors, vol. 21, no. 2, p. 438, 2021. https://doi: 10.3390/s21020438.

[88] S. Schumann, "Why we're mapping down to 20cm accuracy on roads" HERE 360 blog, 2014. Accessed: sep. 03, 2021. [Online]. Available: https://360.here.com/2014/02/12/why-were-mapping-down-to-20cm-accuracy-on-roads/

[89] "Building maps for a self-driving car", Waymo, 2016, sep. 11, 2021. [Online]. Available: https://blog.waymo.com/2019/09/building-maps-for-self-driving-car.html

[90] "HD Map—Highly accurate border-to-border model of the road," TomTom, The Netherlands, 2017. Accessed: sep. 04, 2021. [Online]. Available: http://download.tomtom.com/open/banners/HD-Map-Product-Info-Sheet-improved-1.pdf

[91] R. Liu, J. Wang, and B. Zhang, "High Definition Map for Automated Driving: Overview and Analysis," J. Navig., vol. 73, no. 2, pp. 324–341, 2020, https://doi: 10.1017/S0373463319000638.

[92] "High definition (HD) maps for autonomous vehicles," Sanborn, 2019.Accessed: sep. 04, 2021. [Online]. Available: https://www.sanborn.com/highly-automated-driving-maps-for-autonomous-vehicles/

[93] "Company Information", Ushr, Accessed: sep. 04, 2021. [Online]. Available: https://static1.squarespace.com/static/5a4d1c29017db266a358080 7/t/5ac25a1df950b77ed28e2d21/1522686495724/20180309_Snapsh ot+Backgrounder_final.pdf

[94] "Self-Driving Safety Report" NVIDIA, US, 2021, Accessed: sep. 04, 2021. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/safety-report/

[95] "Maps to the Future", ZENRIN, 2020, sep. 12, 2021. [Online]. Available: http://www.zenrin-europe.com/

[96] "Overview", NavInfo, 2021 , Accessed: sep. 13, 2021. [Online]. Available: https://navinfo.com/en/autonomousdriving

[97] K. Korosec, "This startup is using Uber and Lyft drivers to bring self-driving cars to market faster", The Verge, 2017, Accessed: sep. 13, 2021. [Online]. Available: https://www.theverge.com/2017/7/19/16000272/lvl5-self-driving-car-tesla-map-lidar

[98] "High Definition Maps for Autonomous Driving and Simulation", atlatec, 2021, Accessed: sep. 13, 2021. [Online]. Available: https://atlatec.de/en/

[99] H. Zhang, N. Chen, G. Fan, and D. Yang, "An improved scan matching algorithm in SLAM," 2019 6th Int. Conf. Syst. Informatics, ICSAI 2019, no. 61702320, pp. 160–164, 2019, https://doi: 10.1109/ICSAI48974.2019.9010259.

[100] F. Jamil, N. Iqbal, S. Ahmad, and D. H. Kim, "Toward accurate position estimation using learning to prediction algorithm in indoor navigation," Sensors (Switzerland), vol. 20, no. 16, pp. 1–27, 2020, https://doi: 10.3390/s20164410.

[101] N. Joram, "Dead Reckoning — A Nature-Inspired Path Integration That Made Us Discover the New World", Medium, 2021, Accessed: sep. 26, 2021. [Online]. Available: https://medium.com/geekculture/dead-reckoning-a-nature-inspired-path-integration-that-made-us-discover-the-new-world-ce67ee9d407d

[102] C. Fuchs, N. Aschenbruck, P. Martini, and M. Wieneke, "Indoor tracking for mission critical scenarios: A survey," Pervasive Mob. Comput., vol. 7, no. 1, pp. 1–15, 2011, https://doi: 10.1016/j.pmcj.2010.07.001.

[103] "True-range multilateration", Wikipedia, 2021, , Accessed: sep. 23, 2021. [Online]. Available: https://en.wikipedia.org/wiki/True-range_multilateration

[104] A. Noureldin, A. El-Shafie, and M. Bayoumi, "GPS/INS integration utilizing dynamic neural networks for vehicular navigation," Inf. Fusion, vol. 12, no. 1, pp. 48–57, 2011, https://doi: 10.1016/j.inffus.2010.01.003.

[105] M. Malleswaran, V. Vaidehi, A. Saravanaselvan, and M. Mohankumar, "Performance analysis of various artificial intelligent neural networks for GPS/INS Integration," Appl. Artif. Intell., vol. 27, no. 5, pp. 367–407, 2013, https://doi: 10.1080/08839514.2013.785793.

[106] H. fa Dai, H. wei Bian, R. ying Wang, and H. Ma, "An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network," Def. Technol., vol. 16, no. 2, pp. 334–340, 2020, https://doi: 10.1016/j.dt.2019.08.011.

[107] U. Onyekpe, V. Palade, S. Kanarachos, and S. R. G. Christopoulos, "Learning Uncertainties in Wheel Odometry for Vehicular Localisation in GNSS Deprived Environments," Proc. - 19th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2020, pp. 741–746, 2020, https://doi: 10.1109/ICMLA51294.2020.00121.

[108] U. Onyekpe, V. Palade, A. Herath, S. Kanarachos, and M. E. Fitzpatrick, "WhONet: Wheel Odometry neural Network for vehicular localisation in GNSS-deprived environments," Eng. Appl. Artif. Intell., vol. 105, no. July, p. 104421, 2021, https://doi: 10.1016/j.engappai.2021.104421.

[109] S. Harvey, A. Lee, "Introducing: The Fingerprint Base Map™ for Autonomous Vehicle Mapping and Localization ", Medium, 2018, Accessed: sep. 26, 2021. [Online]. Available: https://medium.com/@CivilMaps/introducing-the-fingerprint-base-map-for-autonomous-vehicle-mapping-and-localization-649dbd1e4810

[110] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2. pp. 239–256, 1992, https://doi: 10.1109/34.121791.

[111] H. Zhu et al., "A review of point set registration: From pairwise registration to groupwise registration," Sensors (Switzerland), vol. 19, no. 5, pp. 1–20, 2019, https://doi: 10.3390/s19051191.

[112] F. Wang and Z. Zhao, "A survey of iterative closest point algorithm," Proc. - 2017 Chinese Autom. Congr. CAC 2017, vol. 2017-January, pp. 4395–4399, 2017, https://doi: 10.1109/CAC.2017.8243553.

[113] P. Biber, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," IEEE Int. Conf. Intell. Robot. Syst., vol. 3, no. November 2003, pp. 2743–2748, 2003, https://doi: 10.1109/iros.2003.1249285.

[114] D. Silver, "NDT Matching", Medium, 2017, Accessed: sep. 27, 2021. [Online]. Available : https://medium.com/self-driving-cars/ndt-matching-acff8e7e01cb

[115] X. Shi, J. Peng, J. Li, P. Yan, and H. Gong, "The Iterative Closest Point Registration Algorithm Based on the Normal Distribution Transformation," Procedia Comput. Sci., vol. 147, pp. 181–190, 2019, https://doi: 10.1016/j.procs.2019.01.219.

[116] "Point set registration", wikipedia, Accessed: sep. 27, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Point_set_registration

[117] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," Comput. Vis. Image Underst., vol. 89, no. 2–3, pp. 114–141, 2003, https://doi: 10.1016/S1077-3142(03)00009-2.

[118] B. Jian and B. C. Vemuri, "Robust Point Set Registration Using Gaussian Mixture Models," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 8, pp. 1633-1645, Aug. 2011, https://doi: 10.1109/TPAMI.2010.223.

[119] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz, "DeepGMR: Learning Latent Gaussian Mixture Models for Registration," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12350 LNCS, pp. 733–750, 2020, https://doi: 10.1007/978-3-030-58558-7_43.

[120] H. Deng, T. Birdal, and S. Ilic, "3D local features for direct pairwise registration," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 3239–3248, 2019, https://doi: 10.1109/CVPR.2019.00336.

[121] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 7156–7165, 2019, https://doi: 10.1109/CVPR.2019.00733.

[122] Y. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," Proc. IEEE Int. Conf. Comput. Vis., pp. 3522–3531, 2019, https://doi: 10.1109/ICCV.2019.00362.

[123] X. Huang, G. Mei, J. Zhang, and R. Abbas, "A comprehensive survey on point cloud registration," pp. 1–17, 2021, [Online]. Available: https://arxiv.org/abs/2103.02690.

[124] A. Woo, B. Fidan, and W. W. Melek, "Localization for Autonomous Driving," Handb. Position Locat., pp. 1051–1087, 2019, https://doi: 10.1002/9781119434610.ch29.

[125] Z. Jiang, Z. Xu, Y. Li, H. Min, and J. Zhou, "Precise vehicle ego-localization using feature matching of pavement images," J. Intell. Connect. Veh., vol. 3, no. 2, pp. 37–47, 2020, https://doi: 10.1108/jicv-12-2019-0015.

[126] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," Proc. IEEE Int. Conf. Comput. Vis., vol. 2, no. Iccv, pp. 1470–1477, 2003, https://doi: 10.1109/iccv.2003.1238663.

[127] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," 26th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR, 2008, https://doi: 10.1109/CVPR.2008.4587635.

[128] S. Cao and N. Snavely, "Graph-based discriminative learning for location recognition," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 700–707, 2013, https://doi: 10.1109/CVPR.2013.96.

[129] M. Aubry et al., " Alignment via Discriminative Visual Elements," 2014, [Online]. Available: https://hal.inria.fr/hal-00863615v1/document

[130] G. Lu, Y. Yan, L. Ren, J. Song, N. Sebe and C. Kambhamettu, "Localize Me Anywhere, Anytime: A Multi-task Point-Retrieval Approach," 2015 IEEE International Conference on Computer Vision (ICCV), pp. 2434-2442, 2015, https://doi: 10.1109/ICCV.2015.280.

[131] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg and V. Lepetit, "Instant Outdoor Localization and SLAM Initialization from 2.5D Maps," in IEEE Transactions on Visualization and Computer Graphics, vol. 21, no. 11, pp. 1309-1318, 15 Nov. 2015, https://doi: 10.1109/TVCG.2015.2459772.

[132] C. Poglitsch, C. Arth, D. Schmalstieg and J. Ventura, "[POSTER] A Particle Filter Approach to Outdoor Localization Using Image-Based Rendering," 2015 IEEE International Symposium on Mixed and Augmented Reality, 2015, pp. 132-135, https://doi: 10.1109/ISMAR.2015.39.

[133] Y. Song, X. Chen, X. Wang, Y. Zhang and J. Li, "6-DOF Image Localization From Massive Geo-Tagged Reference Images," in IEEE Transactions on Multimedia, vol. 18, no. 8, pp. 1542-1554, Aug. 2016, https://doi: 10.1109/TMM.2016.2568743.

[134] T. Sattler, B. Leibe and L. Kobbelt, "Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 9, pp. 1744-1756, 1 Sept. 2017, https://doi: 10.1109/TPAMI.2016.2611662.

[135] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi and A. Fitzgibbon, "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images," 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2930-2937, https://doi: 10.1109/CVPR.2013.377.

[136] A. Kendall, M. Grimes and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2938-2946, https://doi: 10.1109/ICCV.2015.336.

[137] J. Zhang, D. Ai, Y. Xiang, Y. Wang, X. Chen, and X. Chang, "Bag-of-words based loop-closure detection in visual SLAM," vol. 1081618, p. 45, 2018, https://doi: 10.1117/12.2500819.

[138] S. Z. Ahmed, V. B. Saputra, S. Verma, K. Zhang, and A. H. Adiwahono, "Sparse-3D Lidar Outdoor Map-Based Autonomous Vehicle Localization," IEEE Int. Conf. Intell. Robot. Syst., pp. 1614–1619, 2019, https://doi: 10.1109/IROS40897.2019.8967596.

[139] Y. Chen, Y. Chen, and G. Wang, "Bundle Adjustment Revisited," 2019, [Online]. Available: https://arxiv.org/abs/1912.03858.

[140] J. Li et al., "Attention-SLAM: A Visual Monocular SLAM Learning from Human Gaze," IEEE Sens. J., vol. 21, no. 5, pp. 6408–6420, 2021, https://doi: 10.1109/JSEN.2020.3038432.

[141] M. Burri et al., "The EuRoC micro aerial vehicle datasets," Int. J. Rob. Res., vol. 35, no. 10, pp. 1157–1163, 2016, https://doi: 10.1177/0278364915620033.

[142] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, https://doi: 10.1109/TRO.2015.2463671.

[143] Z. Liu and F. Zhang, "BALM: Bundle Adjustment for Lidar Mapping," IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 3184–3191, 2021, https://doi: 10.1109/LRA.2021.3062815.

[144] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," Auton. Robots, vol. 41, no. 2, pp. 401–416, 2014, https://doi: 10.1007/s10514-016-9548-2.

[145] K. Wang, S. Ma, F. Ren, and J. Lu, "SBAS: Salient Bundle Adjustment for Visual SLAM," IEEE Trans. Instrum. Meas., vol. 70, no. 8, pp. 1–11, 2021, https://doi: 10.1109/TIM.2021.3105243.

[146] S. Lu, Y. Zhi, S. Zhang, R. He, and Z. Bao, "Semi-Direct Monocular SLAM with Three Levels of Parallel Optimizations," IEEE Access, vol. 9, pp. 86801–86810, 2021, https://doi: 10.1109/ACCESS.2021.3071921.

[147] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," IEEE Intell. Transp. Syst. Mag., vol. 2, no. 4, pp. 31–43, 2010, https://doi: 10.1109/MITS.2010.939925.

[148] M. Sualeh and G. W. Kim, "Simultaneous Localization and Mapping in the Epoch of Semantics: A Survey," Int. J. Control. Autom. Syst., vol. 17, no. 3, pp. 729–742, 2019, https://doi: 10.1007/s12555-018-0130-x.

[149] C. Stachniss, " Graph-Based SLAM in 90 Minutes" ,BONN University, 2020 Accessed: oct. 05, 2021. [Online]. Available : https://www.unravel.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaabebgwr&download=1

[150] C. N. R. Shohin Mukherjee, Michael Kaess, Joseph N. Martel, "EyeSAM: Graph-based Localization and Mapping of Retinal Vasculature during Intraocular Microsurgery," Physiol. Behav., vol. 176, no. 5, pp. 139–148, 2019, https://doi: 10.1016/j.physbeh.2017.03.040.

[151] S. Jung, D. Choi, S. Song, and H. Myung, "Bridge inspection using unmanned aerial vehicle based on HG-SLAM: Hierarchical graph- based SLAM," Remote Sens., vol. 12, no. 18, 2020, https://doi: 10.3390/RS12183022.

[152] J. H. Jo and C. bae Moon, "Development of a Practical ICP Outlier Rejection Scheme for Graph-based SLAM Using a Laser Range Finder," Int. J. Precis. Eng.

Manuf., vol. 20, no. 10, pp. 1735–1745, 2019, https://doi: 10.1007/s12541-019-00175-0.

[153] A. Akca and M. Ö. Efe, "Multiple Model Kalman and Particle Filters and Applications: A Survey," IFAC-PapersOnLine, vol. 52, no. 3, pp. 73–78, 2019, https://doi: 10.1016/j.ifacol.2019.06.013.

[154] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," Proc. - 8th Int. Conf. Intell. Networks Intell. Syst. ICINIS 2015, no. 10, pp. 74–77, 2016, https://doi: 10.1109/ICINIS.2015.35.

[155] S. Y. Chen, "Kalman filter for robot vision: A survey," IEEE Trans. Ind. Electron., vol. 59, no. 11, pp. 4409–4420, 2012, https://doi: 10.1109/TIE.2011.2162714.

[156] W. Li, Z. Wang, Y. Yuan, and L. Guo, "Particle filtering with applications in networked systems: a survey," Complex Intell. Syst., vol. 2, no. 4, pp. 293–315, 2016, https://doi: 10.1007/s40747-016-0028-2.

[157] M. Brossard, S. Bonnabel and A. Barrau, "Invariant Kalman Filtering for Visual Inertial SLAM," 2018 21st International Conference on Information Fusion (FUSION), 2018, pp. 2021-2028, https://doi: 10.23919/ICIF.2018.8455807.

[158] M. S. Bahraini, "On the Efficiency of SLAM Using Adaptive Unscented Kalman Filter," Iran. J. Sci. Technol. - Trans. Mech. Eng., vol. 44, no. 3, pp. 727–735, 2020, https://doi: 10.1007/s40997-019-00294-z.

[159] P. Slowak and P. Kaniewski, "Stratified particle filter monocular SLAM," Remote Sens., vol. 13, no. 16, 2021, https://doi: 10.3390/rs13163233.

[160] Q. Zhang, Y. Li, T. Ma, Z. Cong, and W. Zhang, "Bathymetric Particle Filter SLAM with Graph-Based Trajectory Update Method," IEEE Access, vol. 9, pp. 85464–85475, 2021, https://doi: 10.1109/ACCESS.2021.3088541.

[161] S. Zeadally, J. Guerrero, and J. Contreras, "A tutorial survey on vehicle-to-vehicle communications," Telecommun. Syst., vol. 73, no. 3, pp. 469–489, 2020, https://doi: 10.1007/s11235-019-00639-8.

[162] A. Chehri, N. Quadar, and R. Saadane, "Survey on localization methods for autonomous vehicles in smart cities," ACM Int. Conf. Proceeding Ser., no. October, 2019, https://doi: 10.1145/3368756.3369101.

[163] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," Comput. Secur., vol. 109, pp. 1–24, 2021, https://doi: 10.1016/j.cose.2021.102269.

[164] J. Halabi and H. Artail, "A Lightweight Synchronous Cryptographic Hash Chain Solution to Securing the Vehicle CAN bus," 2018 IEEE Int. Multidiscip. Conf. Eng. Technol. IMCET 2018, pp. 1–6, 2019, https://doi: 10.1109/IMCET.2018.8603057.

[165] K. T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," Proc. 25th USENIX Secur. Symp., pp. 911–927, 2016, [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity 16/sec16 paper cho.pdf

[166] Regulus Cyber LTD, "Tesla model 3 spoofed off the highway–regulus navigation system hack causes car to turn on its own.", Regulu, 2019,Accessed: oct. 20, 2021, [Online]. Available: https://www.regulus.com/blog/tesla-model-3-spoofed-off-the-highway-regulus-navigation-system-hack-causes-car-to-turn-on-its-own#: :text=Actual%20control%20of%20the%20car,while%20driving %20with%20NOA%20engaged.

[167] B. G. B. Stottelaar, "Practical cyber-attacks on autonomous vehicles," 2015, [Online]. Available: http://essay.utwente.nl/66766/.

[168] Y. Shoukry, P. Martin, Y. Yona, S. Diggavi, and M. Srivastava, "PyCRA: Physical challenge-response authentication for active sensors under spoofing attacks," Proc. ACM Conf. Comput. Commun. Secur., pp. 1004–1015, 2015, https://doi: 10.1145/2810103.2813679.

[169] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR," Blackhat.com, pp. 1–13, 2015, [Online]. Available: https://www.blackhat.com/docs/eu-15/materials/eu-15-Petit-Self-Driving-And-Connected-Cars-Fooling-Sensors-And-Tracking-Drivers-wp1.pdf

[170] G. Soatti, M. Nicoli, N. Garcia, B. Denis, R. Raulefs, and H. Wymeersch, "Implicit cooperative positioning in vehicular networks," IEEE Trans. Intell. Transp. Syst., vol. 19, no. 12, pp. 3964–3980, 2018, https://doi: 10.1109/TITS.2018.2794405.

[171] F. A. A. Ghaleb, A. Zainal, M. A. A. Rassam, and A. Abraham, "Improved vehicle positioning algorithm using enhanced innovation-based adaptive

Kalman filter," Pervasive Mob. Comput., vol. 40, pp. 139–155, 2017, https://doi: 10.1016/j.pmcj.2017.06.008.

[172] Zhenhong Shao, Wenfeng Li, Yi Wu and Lianfeng Shen, "Multi-layer and multi-dimensional information based cooperative vehicle localization in highway scenarios," 2010 IEEE 12th International Conference on Communication Technology, 2010, pp. 567-571, https://doi: 10.1109/ICCT.2010.5688916.

[173] G. M. Hoang, B. Denis, J. Harri, and D. T. M. Slock, "Robust data fusion for cooperative vehicular localization in tunnels," IEEE Intell. Veh. Symp. Proc., no. Iv, pp. 1372–1377, 2017, https://doi: 10.1109/IVS.2017.7995902.

[174] M. Karaim, M. Elsheikh, and A. Noureldin, "GNSS Error Sources," Multifunct. Oper. Appl. GPS, 2018, doi: 10.5772/intechopen.75493.

[175] R. S. Nerem and K. M. Larson, Global positioning system, theory and practice, 5th edition, vol. 82, no. 33. 2001.

[176] U. Onyekpe, V. Palade, and S. Kanarachos, "Learning to localise automated vehicles in challenging environments using inertial navigation systems (Ins)," Applied Sciences (Switzerland), vol. 11, no. 3, 2021, doi: 10.3390/app11031270.

[177] U. Onyekpe, S. Kanarachos, V. Palade, and S.-R. G. Christopoulos, Vehicular localisation at high and low estimation rates during gnss outages: A deep learning approach, vol. 1232. 2021. doi: 10.1007/978-981-15-6759-9_10.

[178] U. Onyekpe, V. Palade, S. Kanarachos, and S.-R. G. Christopoulos, "A quaternion gated recurrent unit neural network for sensor fusion," Information (Switzerland), vol. 12, no. 3, 2021, doi: 10.3390/info12030117.

[179] W. Fang et al., "A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages," Remote Sensing, vol. 12, no. 2, p. 256, Jan. 2020, doi: 10.3390/rs12020256.

[180] H. Yu, W. Zhen, W. Yang, and S. Scherer, "Line-Based 2-D-3-D Registration and Camera Localization in Structured Environments," IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 11, pp. 8962–8972, Nov. 2020, doi: 10.1109/TIM.2020.2999137.

[181] S.Sharath, "Particle Filter : A hero in the world of Non-Linearity and Non-Gaussian," Medium. Accessed: avr 25, 2022, [Online].Available: https://towardsdatascience.com/particle-filter-a-hero-in-the-world-of-non-linearity-and-non-gaussian-6d8947f4a3dc

[182] H. R. Künsch, "Particle filters," Bernoulli, vol. 19, no. 4, pp. 1391–1403, 2013, https://doi: 10.3150/12-BEJSP07.

[183] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM,"IEEE Trans. Robot., vol. 37, no. 6, pp. 1874–1890, 2021, https://doi: 10.1109/TRO.2021.3075644.

[184] R. Mur-Artal and J. D. Tardos,"ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,"IEEE Trans. Robot., vol. 33, no. 5, pp. 1255–1262, 2017, doi: 10.1109/TRO.2017.2705103.

[185] D. Galvez-L'opez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188–1197, 2012

[186] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepICP: An end-to-end deep neural network for point cloud registration,"Proc. IEEE Int. Conf. Comput. Vis., vol. 2019-Octob, pp. 12–21, 2019, doi: 10.1109/ICCV.2019.00010.

[187] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," International Journal of Robotics Research (IJRR), 2013.

[188] S. J. Godsill and A. Doucet, "Particle Filtering in Signal Processing," EURASIP J. Appl. Signal Processing.

[189] A. V Segal, D. Haehnel, and S. Thrun, "Generalized-ICP (probailistic ICP tutorial)," Robot. Sci. Syst., vol. 2, p. 435, 2009.

[190] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and lidar dataset," Int. J. Rob. Res., vol. 35, no. 9, pp. 1023–1035, 2016, doi: 10.1177/0278364915614638.

[191] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 5, pp. 815–830, 2010, doi: 10.1109/TPAMI.2009.77.

[192] Nadav, D.Y.M., Maron, H., Lipman, Y.: DS++: A flexible, scalable and provably tight relaxation for matching problems. ACM Trans. Graph. 36, (2017). https://doi.org/10.1145/3130800.3130826.

[193] Huang, X., Zhang, J., Wu, Q., Fan, L., Yuan, C.: A Coarse-to-Fine Algorithm for Matching and Registration in 3D Cross-Source Point Clouds. IEEE Trans. Circuits Syst. Video Technol. 28, 2965–2977 (2018). https://doi.org/10.1109/TCSVT.2017.2730232.

[194] Daszykowski, M., Walczak, B.: Density-Based Clustering Methods. Compr. Chemom. 2, 635–654 (2009). https://doi.org/10.1016/B978-044452701-1.00067-3.

[195] Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., Wang, Y., Yang, D.: PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving. IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC. 2021-September, 3095–3101 (2021). https://doi.org/10.1109/ITSC48978.2021.9565009.

[196] Du, S., Xu, Y., Wan, T., Hu, H., Zhang, S., Xu, G., Zhang, X.: Robust iterative closest point algorithm based on global reference point for rotation invariant registration. PLoS One. 12, 1–14 (2017). https://doi.org/10.1371/journal.pone.0188039.

[197] Van Der Merwe, R, Wan,R.: Sigma Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models. PhD Thesis, OGI School of Science & Engineering, Oregon Health & Science University, USA.(2004)

[198] Julier, Simon J.; Uhlmann, Jeffrey A New Extension of the Kalman Filter to Nonlinear Systems. Proc. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, 182 (July 28, 1997)

[199] Moireau, P, and Dominique, C. "Reduced-Order Unscented Kalman Filtering with Application to Parameter Identification in Large-Dimensional Systems." ESAIM: Control, Optimisation and Calculus of Variations 17, no. 2 (2010): 380–405. https://doi.org/10.1051/cocv/2010006.

[200] Zhang, J.; Singh, S. Low-drift and real-time lidar odometry and mapping. *Auton. Robot.* **2016**, *41*, 401–416. https://doi.org/10.1007/ s10514-016-9548-2

[201] Laboshinl. "Laboshinl/loam_velodyne." GitHub. Accessed August 8, 2022. https://github.com/laboshinl/loam_velodyne.

[202] Shan, T, and Brendan E. "Lego-Loam: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018. https://doi.org/10.1109/iros.2018.8594299.

[203] HKUST-Aerial-Robotics. "HKUST-Aerial-Robotics/A-LOAM: Advanced Implementation of Loam." GitHub. Accessed August 8, 2022. https://github.com/HKUST-Aerial-Robotics/A-LOAM.

[204] Li, Q, Chen, S, Wang, C, Li, X, Wen, C, and Cheng, M, LO-Net: Deep real-time Lidar odometry, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 8473–8482.

[205] Shuang, L, Cao, Z , Wang, C, Yu, J ans Wang, S. "A Novel Sparse Geometric 3-D Lidar Odometry Approach." IEEE Systems Journal 15, no. 1 2021 : 1390–1400. https://doi.org/10.1109/jsyst.2020.2995727.

[206] Dawen, X, Zhang, M, Yan, X, Bai, Y, Zheng, Y, Li, Y, and Li, H. "A Distributed WND-LSTM Model on Mapreduce for Short-Term Traffic Flow Prediction." Neural Computing and Applications 33, no. 7 2020: 2393–2410. https://doi.org/10.1007/s00521-020-05076-2.

[207] Wael, F. "Real-Time Autonomous Vehicle Localization Based on Particle and Unscented Kalman Filters." Journal of Control, Automation and Electrical Systems 32, no. 2 (2021): 309–25. https://doi.org/10.1007/s40313-020-00666-w.

[208] Team, CARLA. "Carla." CARLA Simulator. Accessed August 9, 2022. https://carla.org/.

[209] Levinson, J., M. Montemerlo, and S. Thrun. "Map-Based Precision Vehicle Localization in Urban Environments." Robotics: Science and Systems III, 2007. https://doi.org/10.15607/rss.2007.iii.016.

[210] Charroud, A.; Moutaouakil, K.E.; Yahyaouy, A. Fast and Accurate Localization and Mapping Method for Self-Driving Vehicles Based on a Modified Clustering Particle Filter. Multimed. Tools Appl. 23 2022. https://doi.org/10.1007/s11042-022-14111-4.

[211] Onyekpe, U.; Lu, Y.; Apostolopoulou, E.; Palade, V.; Eyo, U.E.; Kanarachos, S. Explainable Machine Learning for Autonomous Vehicle Positioning Using SHAP; In Explainable AI: Foundations, Methodologies and Applications. Intelligent Systems Reference Library, York, UK: 2023; Volume 232, Springer, Cham; pp. 157–83. https://doi.org/10.1007/978-3-031-12807-3_8

[212] Charroud, A. AnasCHARROUD/adaptative_particle_filter: Python Implementation of Our Paper 'Rapid Localization and Mapping Method Based on Adaptive Particle Filters'. GitHub. Available online: https://github.com/AnasCHARROUD/Adaptative_Particle_Filter (accessed on 11 November 2022).

[213] Chiang, K.-W. The Utilization of Single Point Positioning and Multi-Layers Feed-Forward Network for INS/GPS Integration. In Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003), Portland, OR, USA, 12 September 2003; pp. 258–266.

[214] Muhammad, K.; Ullah, A.; Lloret, J.; Ser, J.D.; de Albuquerque, V.H. Deep Learning for Safe Autonomous Driving: Current challenges and Future Directions. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 4316–4336. https://doi.org/10.1109/tits.2020.3032227

[215] Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; et al. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

[216] Navon, D.; Bronstein, A.M. Random Search Hyper-Parameter Tuning: Expected Improvement Estimation and the Corresponding Lower Bound. Machine Learning. *arXiv* **2022**. https://doi.org/10.48550/arXiv.2208.08170.

[217] Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* **2013**, arXiv:1312.6034.

[218] Sundararajan, M.; Taly, A.; Yan, Q. Axiomatic Attribution for Deep Networks. Machine Learning. *arXiv* **2017**, https://doi.org/ 10.48550/arXiv.1703.013.

[219] Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; Wattenberg, M. SmoothGrad: Removing noise by adding noise. *arXiv* **2017**, arXiv:1706.03825.

[220] Seo, J.; Choe, J.; Koo, J.; Jeon, S.; Kim, B.; Jeon, T. Noise-adding Methods of Saliency Map as Series of Higher Order Partial Derivative. *arXiv* **2018**, arXiv:1806.03000.

[221] Petsiuk, V.; Das, A.; Saenko, K. RISE: Randomized Input Sampling for Explanation of Black-box Models. Computer Vision and Pattern Recognition. *arXiv* **2018**. https://doi.org/10.48550/arXiv.1806.07421.

[222] Charroud, A., El Moutaouakil, K., Palade, V., Yahyaouy, A. (2023). XDLL: Explained Deep Learning LiDAR-Based Localization and Mapping Method for Self-Driving Vehicles. Electronics, 12(3), 567. MDPI AG. Retrieved from http://dx.doi.org/10.3390/electronics12030567

[223] Othman, Kareem. (2022). Multidimension Analysis of Autonomous Vehicles: The Future of Mobility. Civil Engineering Journal. 7. 71-93. 10.28991/CEJ-SP2021-07-06.

[224] Abdul Razak, S. F., Yogarayan, S., Abdul Aziz, A., Abdullah, M. F., & amp; Kamis, N. H. (2022). Physiological-based Driver Monitoring Systems: A scoping review. Civil Engineering Journal, 8(12), 3952–3967. https://doi.org/10.28991/cej-2022-08-12-020

[225] Hazan, E., & Levy, K. Y. (2015). Beyond Convexity: Stochastic Quasi-Convex Optimization. ArXiv. https://doi.org/10.48550/arXiv.1507.02030.

[226] Sinha, V. B., Kudugunta, S., Sankar, A. R., Chavali, S. T., Kar, P., & Balasubramanian, V. N. (2019). DANTE: Deep AlterNations for Training nEural networks. ArXiv. https://doi.org/10.1016/j.neunet.2020.07.026

APPENDIX A

AUTONOMOUS VEHICLES EQUIPMENT

Lidar (Light Detection and Ranging): Lidar is a remote sensing technology that uses laser light to measure distances and create high-resolution three-dimensional maps of the environment. It works by emitting laser pulses and measuring the time it takes for the reflected light to return, allowing objects to be accurately detected. See example in figure A.1

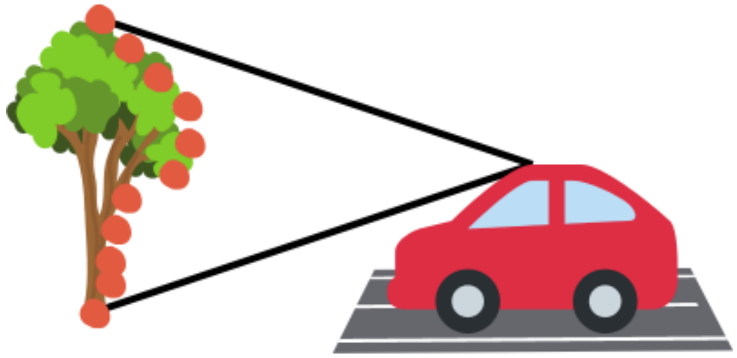


Figure A.1: Graphical representation of training loss values over epochs.

Some type of Lidar:

- Mechanical Spinning Lidar: A rotating lidar sensor that emits laser pulses in multiple directions, capturing a 360-degree view of the environment.
- Solid-State Lidar: A lidar sensor that uses solid-state components, such as MEMS mirrors or optical phased arrays, to scan the surroundings without moving parts.
- Flash Lidar: A lidar sensor that illuminates the entire field of view with a single pulse, capturing a snapshot of the environment in one instance.

Radar (Radio Detection and Ranging): Radar uses radio waves to detect and measure the distance, speed and angle of objects near the vehicle. It is often used to detect the presence of other vehicles, pedestrians and objects and provides information about their speed and relative position. Some type of Radar:

- Short-Range Radar (SRR): Radar sensors designed for detecting objects within a short distance, typically used for parking assistance and collision avoidance at low speeds.
- Medium-Range Radar (MRR): Radar sensors with a wider field of view, capable of detecting objects at moderate distances, commonly used for adaptive cruise control and blind-spot detection.
- Long-Range Radar (LRR): Radar sensors with an extended detection range, suitable for detecting objects at long distances, often used for highway driving and advanced driver assistance systems.

Cameras: Cameras capture visual information by recording images or video of the vehicle's surroundings. They are often used to detect, recognize and track objects. Cameras can provide valuable data for tasks such as lane detection, traffic sign recognition and pedestrian detection. Some type of Cameras:

- Monocular Camera: A single-lens camera that captures 2D images or video, providing information about the visual appearance of objects and their spatial relationships.
- Stereo Camera: A camera system with two lenses positioned at a specific distance from each other, enabling depth perception and 3D reconstruction of the environment.

- RGB-D Camera: A camera that captures both color (RGB) and depth (D) information, often using infrared sensors or structured light to estimate depth and create 3D maps.

Ultrasonic sensors: Ultrasonic sensors use sound waves to detect objects and measure distances. They emit high-frequency sound waves and measure the time it takes them to bounce back after hitting an object. Ultrasonic sensors are often used for proximity detection, parking assist and short-range obstacle detection. Some uses of Lidar:

- Proximity Sensors: Ultrasonic sensors used for detecting objects in close proximity to the vehicle, commonly used for parking assistance and collision avoidance at low speeds.
- Park Assist Sensors: Ultrasonic sensors positioned around the vehicle to assist in parking maneuvers by detecting nearby obstacles.

GPS (Global Positioning System): GPS is a satellite navigation system that provides accurate position and timing information. It uses a network of satellites to triangulate the vehicle's position, allowing for accurate location. GPS is widely used in autonomous vehicles for navigation and global positioning.

IMU (Inertial Measurement Unit): An IMU is a sensor package that combines accelerometers, gyroscopes and sometimes magnetometers to measure acceleration, angular velocity and vehicle orientation. IMUs are crucial for estimating vehicle motion, detecting changes in direction and aiding in localization. The output of an IMU:

- Accelerometer: Measures the acceleration forces acting on the vehicle in three axes (x, y, z).
- Gyroscope: Measures the angular velocity or rate of rotation of the vehicle in three axes.
- Magnetometer: Measures the strength and direction of the magnetic field to determine the vehicle's orientation relative to the Earth's magnetic field.