Department of

Informatics, Systems and Communication

PhD program Computer Science   Cycle XXXVI

Enhancing Deep Learning Methods for Vehicle Localization

Surname ....Vaghi.......... Name ....Matteo..........

Registration number ....780915..........

Tutor: ....Prof. Raffaella Rizzi..........

Supervisor: ....Prof. Domenico G. Sorrenti..........

Coordinator: ..Prof. Leonardo Mariani..........

ANNO ACCADEMICO / ACADEMIC YEAR ..2022/2023..

ABSTRACT

Accurate localization plays a crucial role in the operation of self-driving cars. Inaccurate pose estimates can undermine the reliability of subsequent actions, often leading to unrecoverable failures. However, commonly used sensors such as Global Navigation Satellite Systems (GNSSs) do not meet the reliability standards required by intelligent vehicles in urban areas, and several Deep Neural Network (DNN)-based methods have emerged in recent years to address this issue. Despite the outstanding results achieved so far by localization techniques, some challenges remain unsolved and compromise the deployment of deep learning models in real scenarios. This thesis proposes techniques to address two main problems: providing more robust DNN models in challenging conditions and developing strategies to add robustness by detecting localization failures. Improving the reliability of autonomous driving systems is critical, especially given recent serious accidents involving human road users. To move towards more reliable deep learning localization systems, this thesis provides the following three contributions. Firstly, a 3D DNN that utilizes Light Detection And Ranging (LiDAR) data is proposed. This model covers the entire localization pipeline, including loop closure detection and point cloud registration, with a primary focus on the well-known issue of reverse loop detection. We chose to utilize LiDARs due to their reliability in accurately reconstructing the 3D navigation scene and their property of being invariant to lighting conditions. Secondly, we propose an uncertainty-aware method for global localization without any initial GNSS measurements. In particular, this approach enables the matching between a recorded LiDAR observation and a pre-built map of the navigation environment. Furthermore, thanks to the employment of a sampling-based uncertainty estimation method, the proposed DNN allows for the detection of localization failures. Thirdly, a local refinement localization approach is presented. Different techniques for estimating uncertainty are integrated into an existing pose regression Convolutional Neural Network (CNN), enabling direct estimation of vehicle position and orientation uncertainties. These estimates are then used in an Extended Kalman Filter (EKF) to mimic a realistic application scenario. The presented experimental activity is conducted on publicly available automotive datasets that depict urban scenarios, such as KITTI, Argoverse2, Robotcar, and Mulran, allowing the research community to perform future comparisons.

Qui-Gon: *"You almost got us killed!*
Are you brainless? "
Jar Jar: *"I speak! "*
Qui-Gon: *"The ability to speak does*
not make you intelligent."

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Prof. Domenico G. Sorrenti, who introduced me to the fields of robotics and computer vision when I was a bachelor student, and who has continued to support me during these Ph.D. years. I will always remember our challenging and long meetings, which allowed me to expand my knowledge and understand the importance of pursuing an idea if you believe in it.

I am very grateful to Augusto and Daniele, who have taught me a lot from the moment I took my first steps in IRALab. They are not only colleagues and co-authors but also a source of great inspiration. Special thanks to Simone, Federica, and all the *iralabbers* who have been part of my academic journey. Thanks to my *Doc5* colleagues, a.k.a. the coolest office in the department, and the other people with whom I spent a very entertaining time during our lunches. Thank you for sharing successful and unsuccessful moments with me and making these years unforgettable.

I would like to express my gratitude to the INVETT research group led by Prof. Miguel Angel Sotelo in Alcalá De Henares. The *bocadillos de Mercadona* will always be in my heart.

I would like to mention Ilaria and Alessandro, who have supported me all these years and pushed me to do my best. Without you, my friends, this experience would probably have been even more challenging. Thanks also to Donzo, Carlo, and all my friends based in the cheerful and welcoming Brianza.

I must thank my girlfriend, Valentina, who has always been patiently by my side. She gave me all the emotional support I needed (thanks also to our cats Marple and Wilson) and made me realize that life cannot just be focused only on career goals, but that happiness must be built with the people around us.

Finally, I would like to thank my parents, my siblings, and my whole family, who have always supported me in my decisions and made all of this possible. No words can express the amount of love I have for all of you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

With the recent advancements of self-driving technologies, we observe the emerging trend of intelligent vehicles together with their deployment in urban scenarios. Thanks to this phenomenon, car manufacturers were able to attract the interest of investors and researchers by promising to revolutionize the automotive industry in just few years. The primary objective was to improve the urban mobility by means of vehicles capable to outperform human drivers, leading to a reduction of the number of accidents, a better traffic management, and a decrease of air pollution. However, nowadays the current technology struggles to meet the expectations [10, 167], making investors more skeptical regarding the achievement of autonomously-driven vehicles in a short period. Furthermore, serious accidents involving self-driving cars have recently arose questions about their safety, making more difficult experimental activities in real scenarios [51].

Accurate localization, *i.e.,* the task of estimating the position and orientation of a vehicle with respect to a given map, is one of the tasks to accomplish to ensure a correct and safe navigation and it represents an open issue in the automotive field, since commonly used Global Navigation Satellite Systems (GNSSs) are not able to meet the required standard for autonomous driving in urban areas. Therefore, commonly used localization systems address such a problem by means of the scene understanding task, that is localization is achieved by exploiting observations given by a sensor mounted on-board the vehicle [27, 149, 195]. As localization is performed with respect to the device used for gathering the observation, this task takes also the name of *observer localization*.

In the last decade, Deep Learning had a huge impact in the automotive field, providing perception algorithms that made huge steps forward in terms of accuracy and generalization capability [64, 100, 142], allowing for a great improvement of autonomous self-driving systems. In particular, Deep Neural Network (DNN) models have been applied in a great variety of relevant tasks for autonomous driving [58, 107] including localization [4, 80], and demonstrated to be robust approaches for extracting high level knowledge from heterogeneous observations of the surrounding scene. However, this task, also known as the scene understanding task, still presents several challenges to overcome that undermine the deployment of self-driving cars in urban areas. For instance, in these environments complex structures often occur, *e.g.*, intersections, and human road users often perform unpredictable actions, *e.g.*, a pedestrian crossing a road unexpectedly.

Localization through scene understanding is the main topic addressed in this thesis work by considering different parts of a commonly used localization pipeline and by exploiting DNN-based techniques. This task is particularly relevant for autonomous driving, since localization failures impact negatively the subsequent actions of a self-driving car. From a technical perspective, localization is usually achieved by firstly estimating a rough global pose, which is refined during a successive step. Regarding global localization, a possible approach is to use a direct sensor measurement provided by a GNSS, or to exploit an observation depicting the surrounding scene, *e.g.*, a camera image, from which an algorithm extracts high-level knowledge that is matched with an a-priory representation of the environment. Instead, local estimation techniques leverage on an initial pose estimate, that reduces the localization hypothesis space, and allow for the matching of the observed current scene constituents against a local map.

In the state-of-the-art, observer localization represents a well discussed topic and we could observe the appearance of a large number of DNN-based approaches in the last decade, by also achieving outstanding results. Despite the performance shown so far, these models have several limitations also depending on the device used to perform localization. For instance, camera-based methods [8, 80] allow for a better understanding of the scene due to rich visual information depicted in images, but perform poorly in presence of challenging lighting conditions, and Light Detection And Ranging (LiDAR) based methods [4, 96, 180] enable a better comprehension of the 3D geometry of a scene by achieving also very accurate results, but often require computational expensive DNN architectures or assume the presence of costly LiDAR on-bard vehicles. Other approaches named multi-modal [28, 46, 166] take advantage from the combination of observations having a different nature, but the definition of a strategy to find associations between heterogeneous data is not a straightforward task.

Simultaneouos Localization And Mapping (SLAM) [36, 69, 186] is a typical method to jointly achieve accurate localization and to perform a mapping of an unknown environment during navigation, aiming to a reduction of the localization drift originated from noisy sensor measurements, *e.g.*, intrinsic errors in the odometry estimates. In particular, two main components intervene during the localization process: the loop closure detection and local refinement modules. The former matches a current observation with a previously visited location, and then the latter provides an accurate localization estimate given such a match. Finally, SLAM algorithms exploit this localization estimate to correct the whole vehicle trajectory. Visual SLAM exploits camera images to address the previous tasks, but generally fails in presence of poor illumination, making this approach not particularly reliable for autonomous driving [40, 128, 198]. Therefore, LiDAR-based SLAM is usually preferred, since LiDARs are invariant with respect to lighting conditions and directly provide a 3D reconstruction of the environment. Moreover, LiDAR-based SLAM usually achieves very accurate results. Despite several LiDAR-based methods have been proposed in the literature to face the loop closure detection task, both handcrafted [86, 165] and DNN-based methods [4, 36] often fail in the presence of reverse loops, *i.e.,* previously visited locations that present perceptual differences due to the different acquisition point of view. A similar issue emerges in the successive localization refinement step, when we try to estimate the rigid transform that allow us to align the current LiDAR observation with the detected loop. With LiDAR data, this problem takes the name of point cloud registration [1, 153, 199], and also in this case deep learning techniques [7, 25] provide inaccurate results in the presence of reverse loops. Since LiDAR-based SLAM represents a particularly robust method for urban localization, in this thesis work we address the previously mentioned challenges.

Another relevant issue in the autonomous driving field concerns the reliability of deep learning models that are employed in perception systems. On the one hand, DNNs demonstrated to achieve outstanding results on average, on the other hand, it is challenging to understand whether one can trust a model output or not. Therefore, the ability of detecting failures in DNNs is crucial by considering that self-driving cars operates within a critical domain. For instance, a trend in the automotive research community is to determine the effectiveness of novel DNNs by mainly measuring their accuracy on a narrow set of data, without considering that just a single wrong prediction could cause disastrous consequences in real world. Moreover, we should consider that training deep learning models on a limited amount of data exposes them to biases and to an approximated representation of the world. Therefore, having a model capable to associate a confidence score to its outputs could really improve the reliability on DNN-based perception systems together with the overall driving experience, finally making them more suitable for realistic deployment scenarios. Human drivers often perform actions in presence of partial knowledge of the surrounding environment and, nevertheless, they are able to generally avoid dangerous situations. This behavior results from the ability of humans in quantifying this lack of information, also referred as *uncertainty*, that allows them to understand the risk related to a particular action and then to act accordingly. Due to the importance of managing risk with DNNs, the topic of uncertainty estimation in deep learning models gained an increasing popularity in the literature [52, 73]. In particular, several approaches were proposed [3, 43, 54, 105, 129] to estimate the *aleatoric* part of uncertainty, *i.e.,* data uncertainty, or the *epistemic* part, *i.e.,* model uncertainty. Despite the presence of some methods in the literature, the application of uncertainty estimation techniques in DNN-based localization is still limited [24, 81, 122, 134, 148]. With the aim of providing uncertainty-aware DNNs for urban localization that estimate epistemic uncertainty, two different proposals are provided in this thesis work, by tackling both the global and local components of the localization pipeline.

To summarize, in this thesis we examine the issue of urban localization by concentrating on the reliability of deep learning-based techniques. The term "reliability" has a dual meaning in the proposed work: we investigate how to overcome the shortcomings of LIDAR-based SLAM systems using DNN models in challenging conditions, and we also address the problem of providing uncertainty-aware DNN methods in global and local localization. In particular, the contributions developed during these three years are reported as follows. The first proposal presented (chapter 3) tackles the entire localization pipeline and a LiDAR-based approach is proposed to deal with the problems of global and local localization. In particular, global localization is addressed as a place recognition problem, *i.e.,* the recognition of a previously visited location, with the objective of identifying reverse loop closures. Instead, the local refinement component deals with the point cloud registration task, *i.e.,* the problem of estimating a geometric transform to correctly align two 3D point clouds. Both tasks are performed by a novel 3D DNN named LCDNet, that is also integrated within a SLAM system. Despite the impressive results achieved by LCDNet, it is impossible for such a model to associate a confidence measure with respect to its predictions. Since SLAM systems are sensitive with respect to the presence of false loops, *i.e.,* observations wrongfully matched with a previously visited location, another proposal of this thesis aims to introduce an uncertainty-aware technique for LiDAR-based place recognition. In particular, the proposed method employs a sampling-based technique for estimating uncertainty in the feature representation produced by an ensemble model (chapter 4). Please note, the main goal of this proposal is to provide an exploratory study about the effect of sampling-based uncertainty estimation techniques in the place recognition problem and is not directly related to LCDNet. However, loop closure detection within a SLAM method would be its primary application. Finally, also the local refinement part of the localization pipeline is tackled in this thesis (chapter 5). In particular, an existing multi-modal Convolutional Neural Network (CNN) used for pose regression is modified to exploit three different uncertainty estimation techniques: two sampling-based approaches and a direct estimation method. Moreover, an integration within a filtering algorithm is proposed to demonstrate the importance of epistemic uncertainty within a realistic automotive scenario.

In brief, the outline of this thesis is reported as follows:

- in Chapter 2, an introduction to the autonomous driving background is provided by also reporting relevant existing approaches in the literature. Finally, the relevant open challenges in the urban localization context are identified, that also represent the base of this thesis work;
- in Chapter 3, a model named LCDNet is introduced. In particular, this work tackles the problem of reverse loops in LiDAR-base SLAM both from a global and local perspective;
- in Chapter 4, we start to face the problem of uncertainty estimation in localization deep learning models. In particular a sampling-based method is proposed to estimate feature-wise uncertainty in the place recognition task, with the final aim to detect failures;
- in Chapter 5, an existing multi-modal pose regression CNN is modified to integrate different uncertainty estimation techniques, making it uncertainty-aware. To demonstrate the benefit achieved, this model is then integrated within a filtering algorithm;
- finally, in Chapter 6, a summary of the contributions of this thesis work are presented.

1.1 PUBLICATIONS

This section presents a list of publications produced during the three years of the PhD. The articles are categorized as "already published", "to be submitted", and "published but not strictly related to this thesis".

1.1.1 *Published papers*

Cattaneo Daniele, **Vaghi Matteo**, Valada Abhinav. "Lcdnet: Deep loop closure detection and point cloud registration for lidar slam". *IEEE Transactions on Robotics*, 2022, pp. 2074-2093. [32]

Vaghi Matteo, D'Elia Fabio, Ballardini Augusto Luis, Sorrenti Domenico Giorgio. "Understanding the Effect of Deep Ensembles in LiDAR-Based Place Recognition". *International Conference of the Italian Association for Artificial Intelligence*, 2023, pp. 295-309. [174]

1.1.2 *To be submitted papers*

Vaghi Matteo, Ballardini Augusto Luis, Fontana Simone, Sorrenti Domenico Giorgio. “Uncertainty-Aware DNN for Multi-Modal Camera Localization“. *Robotics and Autonomous Systems journal* (or other similarly relevant journals), in the following weeks. [173]

1.1.3 *Published papers that are not strictly related to this thesis*

Fontana Simone, Cattaneo Daniele, Ballardini Augusto Luis, **Vaghi Matteo**, Sorrenti Domenico Giorgio. “A benchmark for point clouds registration algorithms“. *Robotics and Autonomous Systems*, 2021, 140, 103734. [49]

2 BACKGROUND

This chapter provides an overview of the background of autonomous driving. The first part focuses on describing the devices typically used onboard self-driving cars, which are utilized by a general perception system. Additionally, it briefly discusses the impact of deep learning on autonomous driving perception. The second part introduces a typical localization pipeline and categorizes global and local refinement methods. Finally, this chapter describes the open challenges that will be tackled in this thesis work.

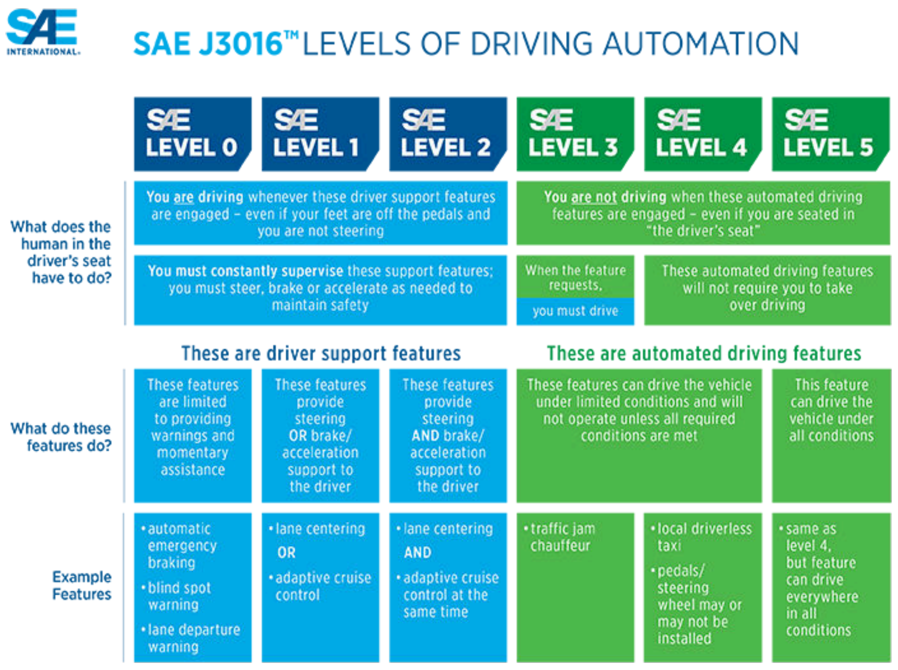
2.1 AUTONOMOUS DRIVING: A BRIEF INTRODUCTION

Autonomous navigation always represented a central topic in robotics, since it has many practical applications that can ease human activities in different scenarios, such as, automotive. However, also the deployment of robots in dangerous scenarios for human workers, *e.g.,* mines or nuclear implants, can improve working conditions and allow companies to promptly take actions when severe accidents occur. Therefore, many challenges and projects have been funded over the last decades to keep the robotics community focused on this topic.

For instance, after the nuclear implant Fukushima disaster in 2011, the Defense Advanced Research Projects Agency (DARPA) in United States launched the *DARPA robotics challenge* to encourage the development of autonomous solutions for complex and dangerous environments [101]. A decade earlier, at the *DARPA grand challenge*, where different research groups competed to develop the first fully-autonomous vehicle navigating off-roads, we could observer the self-driving car named *Stanley* completing the competition by navigating several kilometers in the desert for the first time [170]. Few years later, the same challenge was extended to simulated urban environments [21].

Also in Europe different projects for autonomous cars were proposed in the past. For instance, in 1995 the Eureka Prometheus project demonstrated that navigating in highways with standard traffic conditions is a feasible task for an autonomous vehicle also with high cruising speed [168]. Another example is the ARGO project [19] in Italy, where an autonomously controlled car traveled a distance of 1900km on Italian highways.

Despite the results achieved in the previous projects, one should consider that vehicles were tested in simple scenarios, *e.g.,* highways,



SAE J3016™ LEVELS OF DRIVING AUTOMATION

	SAE LEVEL 0	SAE LEVEL 1	SAE LEVEL 2	SAE LEVEL 3	SAE LEVEL 4	SAE LEVEL 5
What does the human in the driver's seat have to do?	You **are** driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You **are not** driving when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 2.1: Scheme of the six SAE levels.

where the number of road user is limited and clear road signs and markings are usually available. Instead, urban areas represent the most challenging scenarios for a self-driving driving system, since a vehicle can experience the presence of many different road structures and must interact with a greater variety of external actors who often take unpredictable actions. For instance, a pedestrian can cross the roads unexpectedly without perceiving the presence of an incoming vehicle or human car drivers can perform hazardous manoeuvres by breaking the law and road regulations.

With the aim of creating a standard and defining a recommended practice, the Society of Automobile Engineers (SAE) defined the so-called SAE levels of autonomous driving in 2014 (fig. 2.1). This standard comprises six different levels that one can divide in two distinct groups: the *driver support features* and *automated driving features*. The former technologies require constant attention from human drivers, who must also be ready to intervene promptly, while the latter technologies, on the other hand, allow for fully autonomous navigation, reducing the need for human intervention. This categorization is considered by institutions, automotive companies and researchers as the reference standard when developing automotive solutions. In particular, the current research focuses on the last three levels.

Over the last decade, many companies such as Volvo, BMW, Ford, General Motors and Tesla have invested a lot of effort and resources in developing autonomous navigation systems that have also been released in the automotive market. However, today's intelligent vehicles have mainly reached the SAE 2-3 level, and only recently SAE 4 level vehicles have been granted the permission to be tested on very limited public roads. More specifically, some of the existing solutions are the following:

- Tesla Autopilot is a self-driving product between 2 and 3 SAE level;
- Waymo Driver is a SAE 4 level navigation software not requiring a human driver;
- General Motors Cruise is another SAE 4 level self-driving system that enables driverless navigation in limited urban areas;
- Mercedes-Benz publicly released in the automotive market the first SAE 3 level technology in 2023;
- DeepRouteAI recently introduced its SAE 4 self-driving software

In the last decade, we could observe how the heads of relevant automotive companies predicted the advent of fully-autonomous cars in few years, encouraging investors and stakeholders to invest money for their projects. However, after several years we are still far away from the goals set. For example, Tesla CEO Elon Musk announced in 2016 that Tesla vehicles already had all the necessary hardware to achieve a fully automated driving experience, and he also stated that SAE 5 vehicles were one step away from being achieved. However, nowadays Tesla autopilot is still considered a technology between the SAE 2 and SAE 3 levels [167].

On the one hand, maintaining high the expectations regarding self-driving cars has had the advantage of attracting huge amount of investments over the years, which was useful for funding the research in the field. On the other hand, as promises began to be unfulfilled and other trends started to emerge regarding the application of machine learning algorithm, such as Large Language Models (LLMs), key investors began to lose interest and to question the ability of current automotive technologies to achieve SAE 5 level in the near future.

Regarding the previous shift of investors' interest, the story of Argo AI represents a perfect example [10]. Founded in 2016, Argo AI was initially an autonomous driving technology startup that became a joint venture backed by Ford Motors Co. and Volkswagen Group. Over the years, the two companies invested more than 3.6 billion dollars in the project, but as they struggled to find new investors and suffered large net losses, they decided to dissolve it in 2022. Each company decided to retain the technology developed and the staff involved in the project, but they also decided to shift the focus of the research from SAE 4 autonomous driving to SAE 2-3 technologies. This last aspect is very indicative, since it shows that the development of fully autonomous vehicles in urban areas is a tough challenge that is still far from being overcome.

There are several reasons behind the difficulties in achieving fully autonomous systems. Firstly, autonomous navigation involves not only the development of accurate and reliable systems, but also the achievement of agreements with the institutions responsible for road traffic regulation. This last aspect is crucial, as most countries in the world prohibit even the testing of autonomous vehicles in real-world scenarios without the appropriate permits. From an ethical point of view, this scepticism is justified when we consider the negative impact these experimental vehicles could have on everyday mobility and human safety. For instance, General Motors Cruise was recently involved in a serious car accidents by hitting and dragging a person for several meters [51]. Second, ensuring the reliability of Artificial Intelligence (AI) systems, which are typically installed in vehicles, is far from being a straightforward process, especially now with the rise of deep learning algorithms. For instance, since deep learning models are ususally trained on a limited amount of data, they are exposed to biases and limited knowledge of the world. Moreover, collecting and labelling data for autonomous driving is a difficult task that can also be expensive.

The main track of this thesis is to tackle the topic of reliability in autonomous driving localization systems and to ensure their robustness in challenging scenarios. In particular, two main topics are going to be presented: LiDAR-based loop closure detection and registration, and uncertainty estimation in Deep Neural Network (DNN)-based localization models. In this chapter, a brief introduction to the background of autonomous driving will be presented before discussing the work conducted during the PhD.

2.1.1 *Sensors typically available on-board autonomous vehicles*

Like human drivers, autonomous navigation systems rely on information gathered from the environment. However, information is obtained by analysing a large set of sensors data, which involves two successive steps: pattern recognition and classification. In addition, a human driver can associate multiple patterns to fully understand the current scene both from a global perspective and from specific local details, which are depending on the situation. For example, when driving in a crowded area with many pedestrians, a human driver has learned that pedestrians can be unpredictable and that their correct identification and tracking is crucial to avoid disastrous accidents. Furthermore, the ability to estimate their position within the navigation

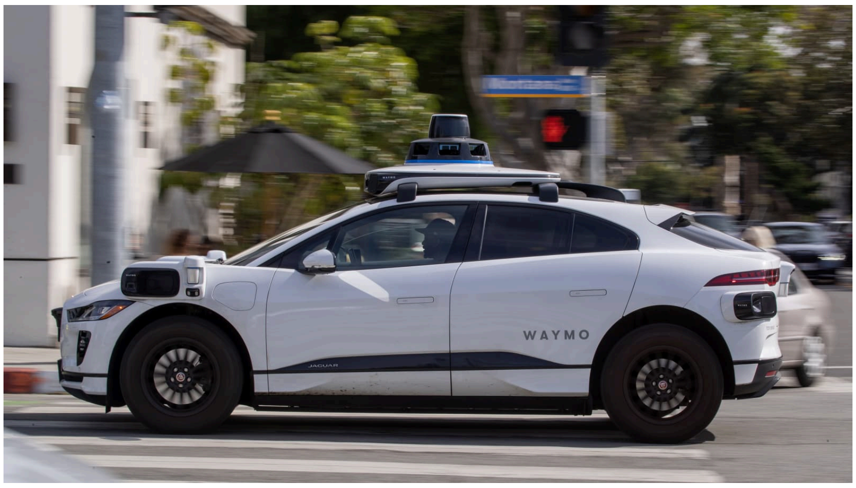


Figure 2.2: Depiction of an autonomous vehicle with several sensors equipped on-board.

environment makes it possible to increase the focus on pedestrians close to the vehicle, closeness depending on the vehicle speed.

In the previous example, the knowledge gained by the human driver results from a collection of different data and their interpretation. In particular, data can be classified according to their nature, and we typically use visual and geometric data when driving. Visual data usually allows us to distinguish between instance classes, while geometric data usually provides useful information about the presence of obstacles together with their 3D shape and distance from the vehicle during navigation.

In the context of autonomous driving, we aim to replicate the typical behaviour of a human driver, *i.e.*, to collect data from the navigation environment in order to extract the necessary information to ensure a correct and safe navigation. To achieve this, different sensors are used on board an autonomous vehicle, such as cameras, Light Detection And Ranging (LiDAR) devices, radars and Global Navigation Satellite Systems (GNSSs), as depicted in Figure 2.2.

Cameras are sensors that provide visual information about the current scene and are suitable for mass production due to their low cost. In fact, it is not unusual to see an autonomous vehicle equipped with several cameras. Since cameras need external light to compute a visual representation of the current scene, and the quality of the outcome depends on the lighting conditions of the environment, they are also known as passive sensors. For this reason, the quality of the data captured depends on the lighting conditions of the environment in which an image is captured.

LiDARs sensors observe the scene by emitting light and measuring the time of flight of the returned signals. A LiDAR sensor records a 3D set of points also known as a *point cloud* and, unlike cameras, LiDARs actively scan a scene. Furthermore, LiDAR sensors are valuable instruments for obtaining precise geometry of the navigation environment. However, they can be costly, and cheaper alternatives may not provide the necessary quality for autonomous driving. One of the best features of LiDARs is their independence from lighting conditions, since they use their own light source, making them suitable for both day and night. On the other hand, they do not perform well in the presence of fog and highly reflective surfaces. They also suffer from the problem of distortion when the vehicle on which the LiDARs are mounted is travelling at high speed even though it can be corrected.

Similar to LiDARs, radars actively observe the scene, but instead of emitting light, they rely on waves of a much longer wavelength. In general, radars are less accurate than LiDARs in providing geometric information about the environment, but they are very useful in detecting obstacles, even in the presence of occlusions. Radars are also very cheaper than LiDARs.

One of the key elements of navigation is the ability to locate a vehicle in relation to a map of the environment. To achieve this, GNSSs are used to provide an estimate of the vehicle's position and orientation, also known jointly as *pose*. GNSSs can achieve really high accuracy, but their performance varies greatly in the presence of external factors that degrade the incoming signal. For example, the presence of clouds or urban canyons, *i.e.*, areas where the incoming signal is occluded by urban infrastructures, has a negative impact on the accuracy of the final estimate. In these areas, the estimates provided by GNSSs in urban areas are often unreliable, making accurate localisation in these scenarios an open research problem.

2.1.2 *Representing the navigation environment*

While navigating, drivers must constantly estimate their position and orientation with respect to the world, as this affects their subsequent decisions. Inaccurate estimates can cause drivers to become lost or, in more serious cases, to encounter dangerous situations. The same task is faced by autonomous vehicles and takes the name of localization.

The localization task involves the reference to a map, that provides a representation of the world in which an autonomous system performs navigation. From a technical perspective, a map consists of a set of local elements composing the navigation environment that can be used to match sensor data recorded during navigation. Therefore, the localization task involves the matching between observations gathered from the current scene and the navigation map. This complements the GNSS estimates when operating in GNSS-covered areas, but it is the

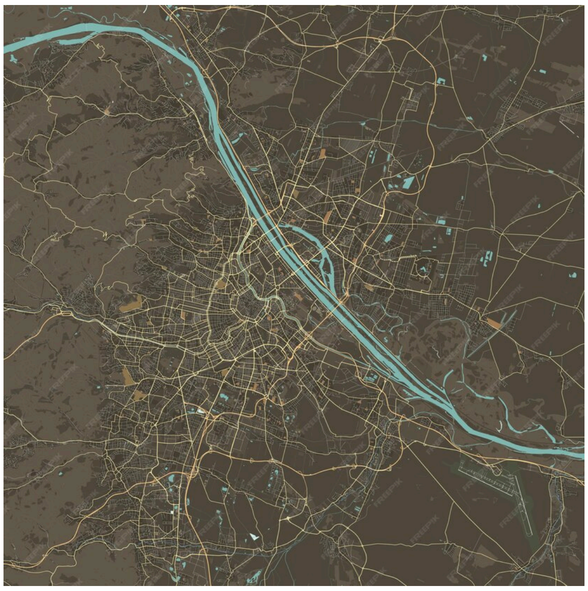


Figure 2.3: Example of an urban topological map.

only localization source. When operating in GNSS-denied scenarios, the achievement of satisfactory results depends also on the map type used

The most popular map types are the following:

- Topological maps
- Grid maps
- Point Cloud maps
- HD maps

TOPOLOGICAL MAPS These maps are used in common navigation systems, such as those utilized by navigation systems present in modern smartphones. However, these maps can also find various applications in robotics. In general, the environment is represented with a graph, where each node corresponds to a geo-referenced point and an edge represents an existing path between two nodes. A nice feature of topological maps is the possibility to enrich nodes and edges with additional information that can be useful for navigation. For example, one can specify road attributes such as the number of lanes, the speed limit or the name of the road. Over the years, companies such as Google, Here, and TomTom, have provided this type of map, but open source projects such as Open Street Map (OSM) [61] have also become popular. An example is depicted in Figure 2.3

GRID MAPS These maps are another type commonly used for robot navigation, especially in indoor scenes, and depending on the appli-

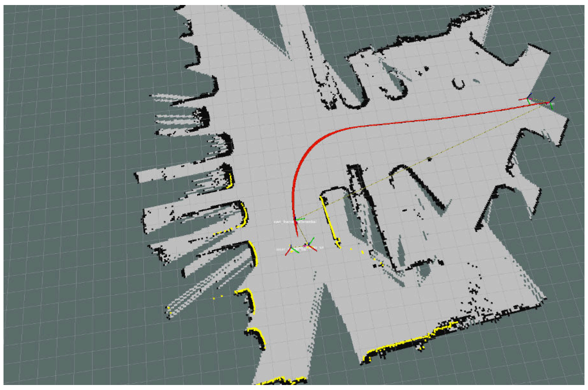


Figure 2.4: A 2D Grid map.

cation scenario, these maps encode a known space by using a 2D, 2.5D or 3D representation. 2D grids (Figure 2.4) provide a compact representation of the scene, where the navigation space corresponds to a set of two-dimensional cells and is usually represented by a Bird's-eye View (BEV). To each cell a corresponding state is assigned, *i.e.,* free, occupied or unknown, to determine whether a robot can pass through that part of the map. Technically speaking, the state is represented by a probability. This representation has the advantage of using less memory and computing power than the other types. However, it is suitable for simple scenes where there are no slopes, and, in general, the third dimension is negligible. 2.5 grid maps provide a compromise between efficiency and an explanatory representation of the environment. Specifically, for each cell the elevation data is reported in addition to the state. As for 3D grids, they subdivide 3D space with a voxel-based representation. On the one hand, they provide an explanatory representation of the environment, but on the other hand they are particularly memory consuming as the resolution increases. A popular and efficient 3D grid extension is *OctoMap* [70]: a 3D grid implementation based on a special data structure called *octree*. [123]. Octrees are hierarchical trees that efficiently partition 3D space, with each node corresponding to a 3D voxel that can be recursively subdivided into eight other 3D partitions. Given a target resolution, the octree algorithm stops when the minimum voxel size is reached. Octomap also includes the occupancy information of each cell, allowing a user to perform queries with low computational cost.

POINT CLOUD MAPS These maps are usually created by concatenating a set of 3D point clouds. There are several methods for building a point cloud map that utilize different sensors such as cameras or LiDARs. Camera-based methods [79] reconstruct a 3D scene starting from a monocular vision system, stereo cameras or also with RGB-D cameras, *i.e.,* cameras that also provide depth information. However,

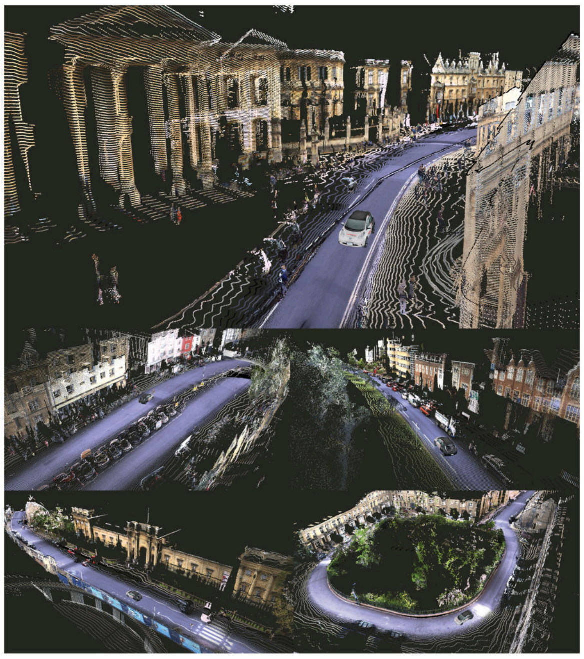


Figure 2.5: Example of a LiDAR point cloud map where RGB data are associated to 3D points. Image taken from [121]

the most accurate methods exploit LiDARs (Figure 2.5), which directly provide 3D point clouds [72]. This representation exploits a simple list of 3D points to store the navigation environment. In addition to the geometric information associated with points, such as their position, other kinds of information can be attached. For instance, it is not unusual to associate RGB camera data with point clouds or high-level geometric information such as surface normals.

HIGH DEFINITION MAPS These maps are built by including different data sources and providing both low-level and high-level knowledge of the navigation environment. In particular, a HD map comprises distinctive landmarks of the scene, that represents a reliable source of information for a perception system of a self-driving car. For instance, these markings provide information such as road geometry, lane attributes, vertical road signs or lane connectivity. In particular, HD maps aim to always provide centimeter accuracy and to be always up to date [115]. Figure 2.6 shows an example of an HD map.

2.1.3 *Perception problems in autonomous driving*

As mentioned in previous sections, perception systems typically available on-board autonomous vehicles utilise a variety of sensors to collect data from the surrounding environment. This data is then used by perception algorithms to retrieve higher level knowledge, a

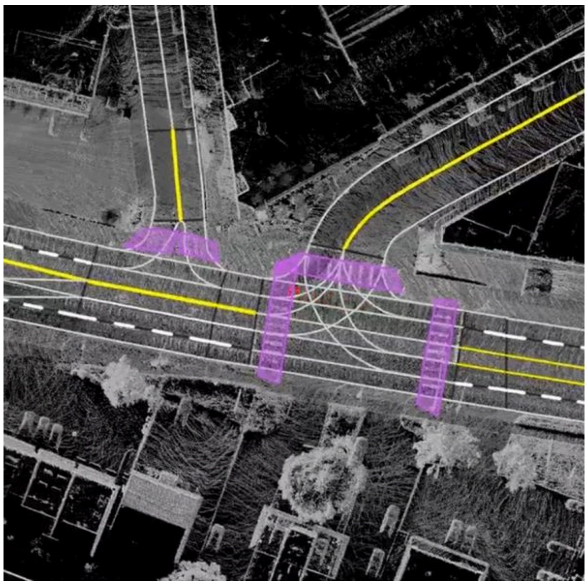


Figure 2.6: Example of a an HD map providing both geometric data and high level information of the navigation environment. Image taken from [187].

task commonly referred to as *scene understanding*. This is crucial for correct navigation and involves several sub-tasks that are handled by different components of the autonomous perception system. For instance, it is common to encounter a situation where one algorithm handles the detection of obstacles [11, 136] or estimates the trajectory of other road users, such as vehicles [74] and pedestrians [158], while another algorithm provides localization estimates by matching scene constituents observed from a sensor against a navigation map [27]. Moreover, it is possible the exploit the output of distinct models to enhance the reliability of an estimate in the same task [172]. Alternatively, components may be combined in a pipeline to iteratively refine an initial estimate [163].

2.1.4 *Deep Learning impact in robot perception*

As previously stated, the primary objective of self-driving cars is to operate in complex environments and perform a variety of tasks to ensure accurate and safe navigation. This is crucial as failures could result in catastrophic accidents. Providing reliable methods to address the challenges of urban areas is complex due to the variety of structures and scene configurations. For example, perception algorithms must demonstrate generalization capability and robustness despite factors such as illumination variability or view point variation, that usually negatively affect their performance.

In recent years, deep learning methods have proven to meet the aforementioned requirements. Starting from the outstanding classification results obtained by Convolutional Neural Networks (CNNs) on ImageNet [144], these approaches have become popular for addressing various perception problems, dealing with different data sources and outperforming previous state-of-the-art methods. For instance, YOLO [142], that is one of the first CNNs for visual object detection, demonstrated impressive results by processing images in real-time. Similar results were achieved also in 3D object detection with a 3D DNN named PointPillar [107]. In other relevant task, such as semantic segmentation, deep learning models became prominent and popular architectures can be found in the literature such as Mask R-CNN [64] for images and RangeNet++ [126] for point clouds data. Finally, also techniques for localization [80, 140] and 3D reconstruction [58] followed the deep learning approach.

2.1.5 *Methods categorization: a sensor type perspective*

As mentioned in section 2.1.1, sensors can provide heterogeneous data from which we can gain different types of knowledge about the current scene, *e.g.,* cameras provide visual data, while LiDARs captures the geometry of the environmental structure. Each sensor type has its own advantages and disadvantages, which usually depend on some external factors such as light or weather conditions.

In the literature, existing approaches can be divided in two main categories: methods that use a specific sensor, and techniques that exploit data from heterogeneous devices. The former can be further divided according to the sensor type used, such as *camera-based*, *LIDAR-based* or *radar-based* techniques. Instead, the latter is generally referred to *multi-modal* approaches, *i.e.,* methods providing an output from data with different nature.

Depending on the task and application scenario, some sensors may be more appropriate than others, making important to carefully consider the desired outcome before deciding which is the best sensor for addressing a task. For example, cameras provide rich visual information about a current scene, which can be particularly useful for detecting and classifying obstacles [200], but at the same time this kind of data does not allow for a direct estimation of objects position and the geometry of the surrounding world [58, 103]. Instead, LiDARs have the opposite issue: they can provide accurate distances of objects and allow for the retrieval of the environment geometry, but understanding the scene can be challenging due to the sparsity of point clouds [37, 155].

Generally, using a single sensor for addressing a task reduces the complexity of the vehicle perception system, not necessarily meaning that we cannot achieve accurate results. However, we should consider a set of appropriate predefined conditions. Nevertheless, in realistic scenarios, these conditions may not hold, leading to a decrement in performance. For instance, expecting a constant presence of an ideal lighting condition within the navigation environment does not represent a realistic application scenario for self-driving cars. Multi-modal approaches aim to overcome the previous limitations by employing a set sensors of different nature such that when a device fails in some adverse conditions the other components of the perception system are still able to provide reliable information. These techniques are mainly built according to one of the following strategies: by extracting local or global descriptors from separate data sources and then matching the results [27, 46], or by jointly exploiting heterogeneous data to directly provide a final estimate [29, 60]. Some approaches assume that all the sensors used are available on-board during the vehicle navigation, but other methods exist that exploit data recorded with a device during an offline stage and use another sensor type during online operations. Localization is a typical task in which the latter methods are used, *e.g.,* a map is built offline with LiDAR, while camera images are used to find matches with the map for localization, when navigating.

2.2 THE LOCALIZATION PIPELINE

Localization is the process of determining the position and orientation, *i.e.,* a *pose*, of a vehicle in relation to a known map. In particular, when the set of parameters that define the position and orientation of a rigid body describe a pose in the three-dimensional space, we usually talk about 6 Degrees of Freedom (*6DoF*) localization, that is typical for self-driving cars. However, in some scenarios, we can consider only three parameters: two for positioning and one for orientation. In these cases, we use the term *3DoF*. For example, in indoor localization, we can assume a flat navigation environment, and a simpler representation of the space is very often sufficient for a wheeled robot.

Correct localization is crucial in autonomous driving as it enables the vehicle to plan realistic actions to reach the desired destination. Additionally, localization is vital for safety, as high accuracy is necessary to avoid dangerous situations. For example, an error of just a few meters could cause a vehicle to enter the wrong lane or fail to stop correctly at a crosswalk.

In this research area, urban localization still is an open challenge due to the intrinsic complexity of the urban environment structure. In particular, although modern GNSSs generally provide very precise

Figure 2.7: Example of an urban canyon.

estimates, their accuracy decreases in urban scenes due to the presence of environmental factors such as urban canyons (fig. 2.7). However, there are other factors that make GNSSs estimates inaccurate, *e.g.*, cloudy weather, and in some cases the sensor measurement is not available at all when navigating in GNSS-denied environments such as tunnels.

Since GNSSs are not always reliable in urban areas, autonomous navigation using only these sensors is not currently feasible. Therefore, an autonomous vehicle must exploit other information gathered from the surrounding scene to increase the accuracy of a previous estimate or sometimes to perform localization without prior knowledge, as depicted in Figure 2.8. One should consider a localization system as a set of components that provide different estimates: some of them aim to provide a rough global pose of the vehicle, while others focus on the refinement of a previous guess.

Fig. 2.9 shows a possible localization pipeline. Localization is represented as a two-step process that aims to answer the following questions: given some clues about the current scene structures, i.e. the scene observed with a sensor, where is the vehicle roughly located within the map? After identifying a candidate location, can one exploit some discriminative local elements to find an accurate pose? For example, given an image of a well-known city square, such as Times Square, we can significantly narrow down the area where the camera device captured the picture if we are able to recognize that location, even without having any GNSS coordinates. This process involves identifying the global structure of the scene, which enables us to locate a specific area in the navigation environment, assuming we have prior

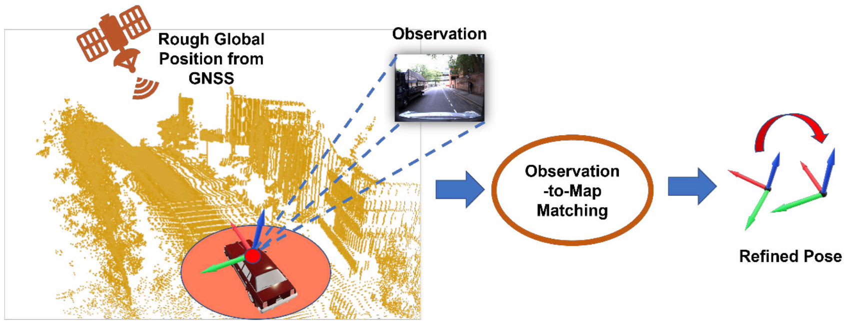


Figure 2.8: Since the GNSS estimate is not reliable for a safe navigation, accurate localization is achieved by exploiting a sensor observation to be matched against a local map.

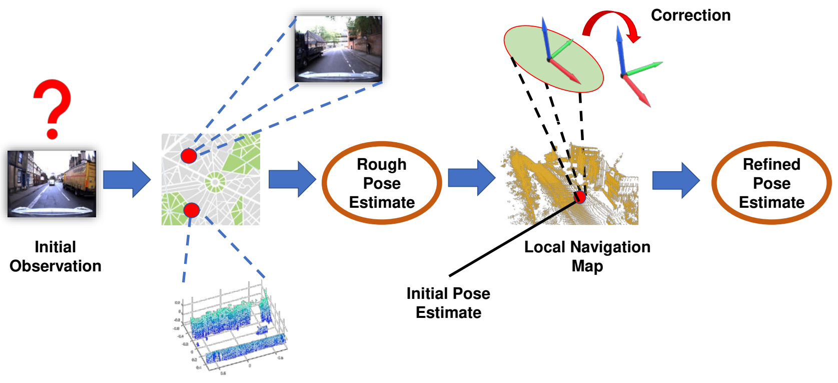


Figure 2.9: A possible Localization Pipeline without a GNSS. Firstly, an observation is used to provide a rough global vehicle pose, then another algorithm refines the initial localization by exploiting another observation.

knowledge of that location. Once a limited area is identified, one can use another observation or the same initial image to obtain a more precise pose estimate. The refinement task involves identifying local structures that provide important information about the observer's location, rather than focusing on the global scene structure. For example, the observer's point of view can be deduced by examining the position of buildings or the appearance of an intersection.

To summarize to overall process, a localization system initially receives an input observation, such as a camera image, to determine the corresponding location within the navigation map where the input observation was likely recorded. After providing a rough global pose, a local refinement component addresses the challenge of identifying more precise matches between the scene constituents depicted in another input observation and the scene elements represented in the candidate map location.

The proposed pipeline is only one of the possible representations of a localization system deployed on an autonomous vehicle. In fact, a localization system is a complex structure that relies on a number of components that cooperate to provide the most accurate pose estimate, and these components can vary under certain assumptions. For instance, by assuming that GNSS measurements are always available, one could use exteroceptive sensor observations only to perform a local refinement [28]. In other cases, odometry-based systems use a sensor observation to find previously visited locations and to correct position errors accumulated during navigation, as in Simultaneouos Localization And Mapping (SLAM) algorithms [36, 69, 186].

2.2.1 *Localization in urban areas with deep learning techniques*

In the pipeline recently described, localization is accomplished with two consecutive steps: *location recognition* and *local elements matching*. In the computer vision field, standard algorithms that deal with these problems employ techniques for the detection of region of interests within an input observation and extract a feature representation from key-points. Those representations also known as descriptors are then employed to directly match observations or to estimate poses by means of an external algorithm such as PnP or ICP [109, 199].

Scale Invariant Feature Transform (SIFT) [116] is a famous approach that accomplish both key-points detection and descriptors extraction in images. As the name suggests, SIFT provides key-points descriptors invariant to a set of transforms, that is scale and rotation, and due to its robustness it represented ground-breaking approach in the field of computer vision. Inspired by SIFT, two other relevant approaches emerged: Speeded-Up Robust Feature (SURF) [14] and Oriented FAST and Robust BRIEF (ORB) [143].

While the approaches presented so far are applied to images, other methods exist for other types of input. For example, Knopp *et al.* [93] proposed a key-point based approach that implements surf for 3D data. Zhong *et al.* [201] introduced a novel descriptor called Intrinsic Shape Signature (ISS) to characterise local parts of a point cloud.

Apart from the different types of data considered, all the mentioned approaches have a common feature, i.e. the procedure used for descriptor detection and extraction aims at finding certain low-level structures considered relevant by its inventor, and due to this aspect these methods are also called *handcrafted*. However, handcrafted techniques have a major disadvantage: one should expect such algorithms to work only in the presence of these particular scene structures. Therefore, these descriptors usually suffer in terms of generalization capability. However, before the advent of neural networks handcrafted feature-based approaches to address the localization problem were widely spread [27, 149, 195].

In the last decade, we could observe how the computer vision community was affected by the advent of neural networks, that outperformed standard approaches based on handcrafted descriptors. In fact, many DNNs and Deep Learning (DL)-based methods have been proposed in the literature across the years, which have had a significant impact in computer vision research. One of the most notable works that has transformed the field is that of Krizhevsky *et al.* [100], where they proposed a CNN that outperformed other approaches considered state-of-the-art at that time in the classification challenge on ImageNet. Since then, several architectures have been proposed across the years [65, 89] and neural networks have been used to tackle a great variety of tasks [57, 175].

The trend of DL models has impacted localization as well and several approaches have been proposed across the lest ten years. In the literature, we can distinguish approaches according how they are positioned within the pipeline described in section 2.2 [8, 29, 96, 140] and we can make a further classification by considering the type of data fed to DNN models, *e.g.*, camera-based [80] or LiDAR-based [4]. Regarding such distinctions, a deeper introspection will presented in the following sections.

Training neural networks for localization is not a straightforward task, especially when the application scenario is the automotive field. This problem is due to the characteristic of DNNs of being demanding in terms of training samples, since one of the condition required to ensure their robustness is to cover a large variety of inputs that well represent real-world scenarios. In particular, ensuring such coverage in the automotive field is essential due to the complexity of urban environments.

During the last decade, different datasets emerged in the automotive field [23, 55, 187] allowing research to develop several deep learning methods. However, those datasets are suitable for developing prototype applications and provide a very narrow representation of the world. Recording sufficiently large automotive datasets can be a very challenging task due to several reasons. Firstly, a variety of sensors are required on board a vehicle, which can also be very expensive, such as LiDARs. Second, the on-board sensors require calibration of intrinsic and extrinsic parameters. Thirdly, distinct sensors collect data at different frequencies and frames synchronization should be ensured if one want to compare observations from different devices. Another important issue is the noise management during the data acquisition process, *e.g.,* when the vehicle is moving at high speed LiDAR point clouds could be affected by distortion and camera acquisition could provide in blurred images. Finally, providing accurate groundtruth poses is challenging even with RTK GNSSs due to the issues described in section 2.1.1. This last problem is particularly relevant in localization, since without accurate groundtruth poses it is not possible to achieve accurate localization performance with deep learning models. Therefore, authors of automotive datasets often perform post-processing operation by applying algorithm such as SLAM [15, 112].

To overcome the problem of having labeled data to train DNN models, another trend in the deep learning community has emerged in recent years, namely the topic of unsupervised learning. Unsupervised learning aims to define training procedures that do not use labelled data, but instead use loss functions based on a consistency measure. An example is the left-right consistency loss used in unsupervised depth estimation and proposed by Godard *et al.* [57]. Unsupervised learning has also been proposed in localization approaches, in some cases achieving competitive results with respect to certain supervised methods [2, 194].

Since localization represents one of the main topics of this thesis, in the next sections a brief introduction to some of the most popular deep learning methods emerged in the last few years will be provided. In particular, in section 2.2.2 global localization approaches will be described, while in section 2.2.3 local refinement methods will be discussed. Note that, since in this thesis camera and LiDAR data were exploited, the discussion will be main focused on methods that exploit this kind of data.

2.2.2 *Global localization approaches*

Global localization approaches aim to provide a global pose corresponding to the point of view where an observation was acquired without using an initial guess, that is no prior knowledge of the vehicle is given. In particular, some deep learning methods directly estimate a pose, while others perform observations matching with the aim of matching a current observation with a previously collected one, giving a rough estimate of the observer position and orientation. This last task takes also the name of *place recognition* and consists in the matching of a current location, *e.g.,* observed within a camera image, to a previously visited place, *e.g.,* a city place image from a database. Observation localization can be categorized according to the type of input and one can distinguish methods with terms such as camera-based or LiDAR-based.

CAMERA-BASED Camera-based approaches has the characteristic of using a camera observation to estimate localization, that usually imply that such an estimate corresponds to the camera pose. PoseNet [80] is a popular camera localization CNN that performs 6DoF localization given an input image. The authors also proposed a bayesian version of PoseNet [83] that also provides uncertainty estimates of the different pose parameters. Unfortunately. although both methods achieve accurate localization, they are not scalable to new environments. In fact, they can perform localization only within environments depicted in the training set. However, Vödisch *et al.* [176] recently proposes a dual-model for adapting a camera-based approach to new environments while performing SLAM. Another popular localization CNN was proposed by Arandjelović *et al.* [8]. The approach, named NetVLAD, achieve localization by means of the place recognition task, that is the method's output is a correspondence with a previously visited location instead of a direct pose estimate. In particular, this technique integrates the well-known VLAD descriptor [9] within the training process of the network. The cheap cost of camera makes camera-based methods suitable for the mass production of intelligent vehicles. However, their performance typically decreases in presence of challenging lighting conditions or repetitive patterns present in the scene.

LIDAR-BASED LIDAR-based methods use DNNs to process input point clouds, which are sets of 3D points that provide a geometric representation of the scene. Processing point clouds with a DNN is not a straightforward process due to the sparsity of the representation, which makes it more difficult for a model to understand local structures, and DNNs recently proposed in the literature consider different data structures, *e.g.*, unordered lists [138], graphs [184] or voxels [204]. PointNetVLAD [4] represents one of the first deep learning-based methods for the 3D place recognition task. Taking inspiration from NetVLAD, the authors integrated a NetVLAD layer withing an existing 3D DNN architecture named PointNet [138]. In recent years, Komorowski [96, 97] *et al.* proposed a similar approach by providing a model which exploits sparse convolutions. The same authors also proposed a method that initially performs place recognition and then aligns an input point cloud and a matching candidate [98]. Although this last method cover the whole localization pipeline, one can consider it a global method since it does not require a prior localization guess. A similar method is PointLoc [180]: a 3D DNN for directly regressing a 6DoF pose given an input point cloud. LiDAR-based methods have a significant advantage than camera-based approaches: they achieve high localization accuracy. However, LiDAR sensors are usually very

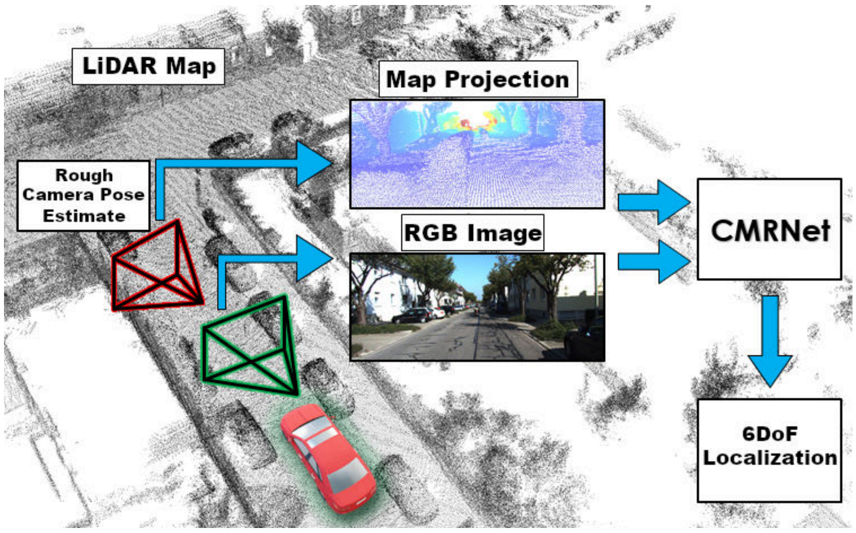


Figure 2.10: CMRNet: a local refinement approach. Image taken from [28]

expensive and their employment on-board autonomous vehicle is not suitable for the mass production.

MULTI-MODAL Multi-modal approaches correspond to methods that take advantage of different sensors, *e.g.*, by matching visual and LiDAR data. Cattaneo *et al.* [29] recently proposed a place recognition method that directly match image and LiDAR data. In particular, approach a 2D CNN and a 3D DNN learn a common feature space thanks to a knowledge distillation learning technique. Tang *et al.* [166] recently proposed an interesting technique. In particular, they introduced a DNN model called RSL-Net that enables a direct matching between radar data and satellite images by synthesizing a radar observation from an input satellite image, allowing for a direct matching with a radar scan. Similarly, Yin *et al.* [193] introduced a novel approach to perform cross-modal place recognition between radar and LiDAR data by exploiting a joint training strategy.

2.2.3 *Local methods*

The approaches that perform localization given a prior knowledge of their location are also referred as local methods. Such a prior knowledge does not necessarily represents an initial pose guess, but can also refer to an observation of the scene collected in a previous moment, *e.g.*, a camera image or a LiDAR point cloud, and these methods aim to provide the relative transform between the two observations.An example of a local method is reported in Figure 2.10.

CAMERA-BASED Sarlin *et al.* [148] proposed a camera-based deep learning models that estimates a pose between an input RGB image and another reference image. Instead of providing a direct pose estimate, this approach finds a match between a set of deep features and a 3D model representing a common view of the scene. Note that, in this work the regression part is addressed by an external alignment algorithm. Another interesting approach is DeepVo [179], that is a Recurrent Neural Network (RNN) for estimating visual odometry given a sequence of consecutive images. In particular, the transform estimated corresponds to the movement of the vehicle's camera observing the navigation scene. Taking inspiration from the previous technique, Li *et al.* [111] introduced UnDeepVo, a method that implements an unsupervised learning procedure to estimate visual odometry. More recently, Ban *et al.* [13] introduced a novel deep learning model for visual odometry estimation. In particular, this method utilizes the information obtained from the tasks of depth regression and optical flow estimation to obtain accurate relative pose estimates.

LIDAR-BASED A common strategy for refining a localization estimate with LiDAR data is to align a current LiDAR observation with a candidate point cloud that is likely to be part of the same map location. This task is called *point cloud registration* and involves estimating a rototranslation that allows the localization of a current LiDAR observation. Point cloud registration is a well-knonw problem in the literature and many registration algorithms have been developed in the past such as Iterative Closest Point (ICP) [135, 153, 191]. In the last decade, also the deep learning community tried to exploit DNN for addressing this task and started to propose models which incorporate popular feature extractors and computer vision algorithm [7] or take inspiration directly from ICP [182]. Although these methods often achieve accurate results when registering small point clouds, registration of large 3D scene still represents an open challenge. A popular deep learning-based registration method is HRegNet [117]: a hierarchical DNN model that performs fine registration between two input point clouds. In particular, the authors introduce an architecture that allows them to find correspondences between deep feature keypoints and to perform registration of the input point clouds. Another interesting DNN approach was proposed by Lu *et al.* [118]. In particular, the proposed model handles the noise introduced by dynamic objects to ensure a greater robustness of the registration procedure.

MULTI-MODAL A relevant multi-modal method recently proposed by Feng *et al.* [46] is 2D3DMatchNet: a DNN that addresses the camera-to-LiDAR map localization task by matching 2D-3D patches extracted from an image and a LiDAR point cloud respectively representing a portion of the navigation map. During a first step, patches are identified with an handcrafted detector while the computation the corresponding descriptors is delegated to a DNN trained to represent a common feature space between images and LiDAR point clouds. Given a set of image-to-LiDAR matches, an external algorithm named EPnP [109] estimates the final pose. Inspired from the previous work, a similar approach named DeepI2P [110] converts the matching task to a classification problem. Cattaneo *et al.* [28] introduced a multi-model CNN named CMRNet , that performs the camera-to-LiDAR map registration task. Given an initial pose estimate, their approach synthesizes a point of view from which observing a LiDAR map, that is they project the point cloud within a virtual image plane creating a *LiDAR image*. Then such a LiDAR image together with a camera image are fed to a CMRNet, that compares the two point of views and directly estimate a pose. Another version of the model named CMRNet++ [31] was recently proposed. Instead of directly providing a pose, this model estimates the *optical flow*, *i.e.,* a pixel-wise mapping, between camera and LiDAR images and delegate the pose estimation task to an external algorithm [109]. More recently, Chen *et al.* [34] introduced a similar method that jointly performs direct pose regression and optical flow estimation.

2.3 OPEN CHALLENGES IN DEEP LEARNING-BASED LOCALIZATION

As we have seen so far in this thesis, autonomous driving is a very complex topic and, despite the progress made in the last decade, we are still far from deploying fully autonomous systems in every day scenarios. In particular, current perception algorithms must face several tasks that become really challenging when addressed in urban areas, where complex structures are often encountered, *e.g.,* intersections and buildings, and human actors do not always act rationally.

Another indicator that suggests the complexity of the task is the lack of a standard between the autopilot software available on the market or research. In fact, there is not a shared view in the community regarding the suite of sensors to use on-board a vehicle, *e.g.,* Waymo Driver employs LiDARs while Tesla autopilot relies only on cameras and radars, or the type of technologies required to navigate, *e.g.,* some works question the necessity of expensive and hard to maintain maps [171].

DNNs have demonstrated their ability to tackle robot perception tasks, but some challenges still remain due to the lack of robustness and scalability of these techniques without a proper fine-tuning of the model parameters. Although deep learning models have led to significant improvements in the field, their reliability can be limited due to a lack of transparency, making it challenging to anticipate possible failures. Moreover, in the case of a system failure, precisely determining the root cause in a deep learning-based system is not straightforward. This aspect makes it less socially acceptable and more difficult to regulate from a legal perspective.

The perception system of an autonomous vehicle is a set of several components that handle a multitude of tasks such as obstacle detection or motion planning, and tackling all these aspects is not feasible within a PhD thesis. Accurate localization represents one of these crucial tasks to accomplish if one would like to ensure a correct and safe navigation, that also demonstrates its complexity within urban areas, where an error of few centimeters can endanger other road users. Due to the centrality of the problem, localization was the topic addressed in this thesis work, by focusing mainly on the problem of enhancing robustness and reliability of some DNN-based methods. From a technical perspective, two main issues were addressed: the *detection* of reverse loops with a LiDAR-based approach (Chapter 3) and the estimation of uncertainty within DNN-based local (Chapter 5) and global (Chapter 4) localization methods. There are of course strong connections between such issues, especially between loop detection and place recognition in 3D data. Moreover, the pattern followed in the development of this thesis stems from the consideration that the deeper the knowledge in localization, the stronger the urgence to integrate state of the art with methods allowing the inference output to be also endowed with a reliable measure of its reliability.

2.3.1 *The re-visiting problem in LiDAR-based methods*

When a robot navigates through an environment, a prior knowledge of the surrounding is often missing. In this case, without the possibility of matching current scene constituents with a previous map location, the robot can only relies on sensor measurements to estimate its own movement and then to localize itself. However after certain steps, the localization accuracy starts to decrease due to the intrinsic noise of sensor measurements, that leads to the accumulation of drift over time and makes impossible correctly navigating. This is a problem that affects several approaches such as odometry-based methods [67]. Nevertheless, if a robot re-visit a previously seen location, it is possible to refine the current localization estimate and to correct the vehicle trajectory.

The set of methods facing the previous problem are named Simultaneous Localization and Mapping (SLAM) algorithms and their main goal is to jointly provide a mapping of the unknown navigation environment and a correct pose estimate of a robot. Obviously, the quality of one task is strictly dependent from the other, *i.e.,* it is not possible to achieve accurate localization without an accurate mapping and vice-versa. Moreover, the mapping task usually involves the concatenation of multiple 3D scan [156] or reconstructed key-points from images [128].

GENERAL PROBLEM Generally speaking, a SLAM system involves three distinct tasks: (1) the accumulation of consecutive data acquisitions to build a map, by employing other sensors such as an Inertial Measurement Units (IMUs), (2) the identification of previously visited locations, that is the *loop closure detection* task, (3) Alignment of loop closures detected to correct the progressive drift accumulated during the first step. The identification of loops and their correct alignment with the map built is fundamental to achieve accurate localization, since serious errors in this part of the localization pipeline usually lead to failures of the overall algorithm. For instance, false loops represent one of the worst enemies of SLAM algorithms, considering that unrecoverable localization errors occur when a current observation is matched to a wrong map place. Also false negatives, *i.e.,* observations that have a match to a certain map location and are discarded by the algorithm, can produce the same effect, but usually over a longer period of time.

Correctly detecting a loop closure is far from being an easy task. In fact, when a vehicle navigates within an environment and revisits a same location, the sensor observations may depict significant structural differences due to changes in the point of view or to the presence of external factors that modify the scene representation, *e.g.,* changes in illumination between day and night. Under these conditions, visual-based methods often fails [40, 128, 198] making these set of approaches not particularly reliable for the task.

LIDAR-BASED SLAM As mentioned earlier, visual approaches are sensitive with respect to adverse environmental conditions that could arise during long-term navigation. Instead, LiDAR-based techniques show higher robustness due to the property of LiDAR of being invariant to light conditions, making them more reliable in SLAM than camera-based methods. Furthermore, the performance obtained by LiDARs are also due to their ability in providing accurate 3D geometric representation of a scene.

LiDAR-based loop closures detection with DNN-based methods has became another trend emerged in the deep learning community and several approaches have been proposed in the last few years [4, 36].

However, these techniques are not robust in presence of strong view point variations, such as reverse loops, and achieve worse performance than state-of-the-art handcrafted methods. Nevertheless, handcrafted approaches [86, 165] are not computationally efficient, making them slower than deep learning methods. Furthermore, due to the high computational time required for computing and matching descriptors, they do not easily meet the high frequency standard of autonomous navigation.

While loop closure detection cover the second part of a typical SLAM pipeline, the successive step is the accurate localization of a current observation with respect to a previously visited location. In this task, the main goal of LiDAR-based methods is to perform a registration of two point clouds, *i.e.,* to find a rigid transform that correctly aligns the two 3D observations. Iterative Closest Point ICP methods [1, 153, 199] generally provide an accurate point clouds alignment. However, they often achieve sub-optimal results in presence of challenging conditions, such as revers loops, by significantly reducing the benefit achieved with SLAM. In the last few years, several DNN-based approaches appeared in the literature addressing the point cloud registration task such as PointnetLK [7] and PCAM [25]. The former introduced the popular Lucas & Kanade algorithm [119] within a 3D DNN architecture, while the latter proposed to mix low-level and high-level geometry to improve the quality of the registration process. However, also these state-of-the-art methods achieve accurate results only in presence of small misalignment between point clouds, *i.e.,* they cannot perform a correct registration of reverse loops where the existing rotation between point clouds can be up to 180°.

SLAM has several practical applications, spacing from autonomous cars [160] to Unmanned Aerial Vehicles (UAVs) [146], up to self-driving marine robots [196], making this topic particularly pivotal in the robotics community, and ensuring a better accuracy and robustness of DNN methods against the current challenges constitutes a first step to improve deep learning-based SLAM reliability and safety. Therefore, considering the superiority of LiDAR in this field with respect to other sensors, the detection and registration of loops in LiDAR-based SLAM with a DNN was indeed taken as the first objective in this thesis work.

2.3.2 *Uncertainty estimation in DNNs*

While AI applications are becoming increasingly present in the modern society and are significantly changing working activities and the labor market, their limitations have been started to emerge, *e.g.,* it is challenging to understand whether one can trust a model output for exploiting it in a critical decision making process. In particular, when considering safety-critical domains, such as autonomous driving, the deployment of AI-based systems in real-world scenarios remains limited due to these emerging issues. Therefore, introducing AI within human activities should be done cautiously if we want to prevent disastrous consequences [16].

GENERAL PROBLEM DNNs have demonstrated their superiority in a large number of fields by achieving unprecedented performance in the last few years. Therefore, the research community continuously put a significant effort in realizing novel approaches with the aim of increasing the accuracy of deep learning systems. However, accuracy itself just describe the ability of a model to accomplish a specific task on a set of data, that provides a limited representation of the world, and it is unlikely to encounter the same conditions in every possible realistic scenario, making the behavior of a DNN unpredictable most of the time. Furthermore, a limited amount of data is also used to train models providing them a partial knowledge of the world and exposing them to biases [17, 78].

Unpredictability also results from a lack of transparency of DL models, since it is challenging to trace the possible reasons that caused a failure and is not clear how to define an explicit relation between certain inputs and model outputs considering the millions of parameters usually involved in the computation.

To overcome the previous issues, different approaches appeared in the literature with the aim of achieving robustness in Neural Networks (NNs) by ensuring invariant and equivariant properties with respect to possible input transforms [185], *e.g.,* subjects of interests can appear differently within an image according to the observer point of view. Moreover, other methods propose to reduce the lack of transparency of deep learning models aiming to better understand their behavior during inference or training. The former set of techniques lie under the topic named Geometric Deep Learning (GDL) [20], while the latter is also referred to Explainable Deep Learning (EDL) [141].

The previous methods aim to increase the generalization capability or to detect flaws of models originated from training. Although both GDL and EDL aim to address the mentioned problems in the deep learning field, they actually provide partial solutions, and how to act in presence of data not well represented in training datasets still remain an open challenge. Detecting out-of-distribution (OOD) data is a well-known issue in deep learning [22], since it as many practical application such as the detection of treacherous adversarial attacks [108].

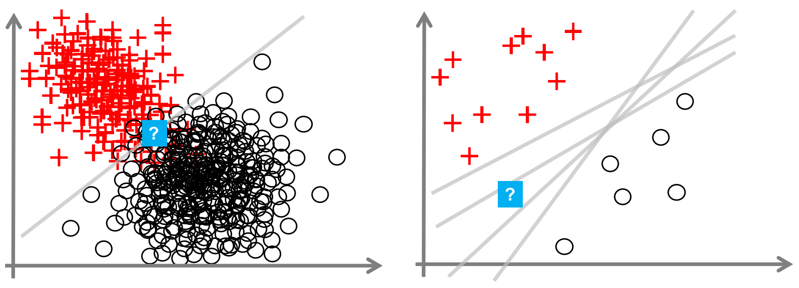


Figure 2.11: Aleatoric uncertainty (left) originates from the overlapping region between the two data classes (red crosses and black circles). Instead, epistemic uncertainty (right) is due to the lack of data that make impossible to chose the best model (grey lines). Images taken from [73].

UNCERTAINTY ESTIMATION Uncertainty estimation in deep learning models represents the task that aims to detect OOD data by also providing a quantitative measure. This measure can be interpreted as the probability that a DNN model is correct in its prediction. The term uncertainty typically refers to situations where a system operates with incomplete knowledge of the application environment, where the employment of uncertainty-aware systems can be useful for discriminating "good" and "bad" decisions when managing risk is crucial. However, uncertainty can originate from different sources and one can distinguish different typologies. In the literature, an accepted categorization includes two main classes: *aleatoric* and *epistemic* [91]. As described by Hüllermeier and Waegeman [73], *aleatoric* uncertainty is produced by random effects within the data generation process that cannot be measured and, consequently, is irreducible. This uncertainty is intrinsic to data and is also known as data or statistical uncertainty. Instead, *epistemic* uncertainty is the uncertainty originated from limited knowledge, *i.e.*, ignorance, that can be reduced by providing additional information. Training a DNN model on a limited amount of data constitutes a typical example where epistemic uncertainty arises in deep learning. In fig. 2.11 a schematic representation of both uncertainties is provided. On the left, aleatoric uncertainty arises due to the presence of an overlapping region between the two data classes. Even with the best possible model (grey line), it is not possible to certainly distinguish a sample lying in that region. On the right, the figure represents a situation where a limited amount of samples is available to fit our model. This cannot ensure complete coverage of the possible instances belonging to the two classes and several models can provide correct class separation according to data. However, it is uncertain which one provides the best fit for a given application scenario.

Uncertainty estimation in deep learning is an existing topic in the literature [52], which also finds a place in the area of computer vision. One of the most relevant study in the field was proposed by Kendall and Gal [83], where they tackle the problem of estimating both aleatoric and epistemic uncertainties for depth regression and semantic segmentation in images.

Regarding epistemic uncertainty estimation in DNNs, Bayesian deep learning [43, 129] represents one of first frameworks that faced the problem and from which Bayesian Neural Networks (BNNs) originated. In BNNs, model's parameters are not deterministic and a distribution is placed over the weights, *e.g.*, with a Gaussian prior distribution $W \sim N(0, 1)$. Therefore, given a dataset $D = \{d_1, ..., d_N\}$, the posterior over the weights is defined as $p(W|D)$ by simply applying Bayesian inference and define the set of plausible model's parameters given the data. However, the exact evaluation of this posterior is often unfeasible due to the complexity of deep learning models and forcing the usage of approximation techniques [53].

Sampling uncertainty is a popular method to perform posterior approximation and two relevant sampling-based approaches are surely Monte Carlo Dropout (MCD) [54] and Deep Ensembles (DEs) [105]. Given a single input, the MCD technique performs several inferences by applying dropout [164] each time to the model's weights to synthesize slightly different models, which provide their own response to the input. By aggregating the outputs, we finally obtain a probabilistic prediction, *e.g.*, by computing mean and variance of samples. Similarly, DEs also exploit multiple instances of a DNN by training several models starting from different weights initialization. DEs usually outperforms MCD in terms of accuracy and uncertainty quality. Fort *et al.* [50] provide an intuition regarding the superiority of DE methods, that is MCD provide samples by only exploiting a single local minimum, while DE members characterize multiple local minima describing a greater variety of patterns in data. Both the approaches assume that epistemic uncertainty can be described with a normal distribution $N(\mu, \sigma^2)$. Although DEs improve accuracy and provide a good representation of uncertainty, they are resource demanding due to the usage of multiple instances of a model by increasing memory usage and computational time. In general, without the proper hardware, sampling-based techniques should not be used in application domains requiring high frequencies, *e.g.*, autonomous driving. To mitigate such issues, techniques such us Multiple-Input Multiple-Output (MIMO) models offer a good trade off between resources demand and performance. Instead of employing different instances of a DNN, MIMO architecture comprises both independent sub-networks providing different outputs, *i.e.*, samples, and shared parts [63].

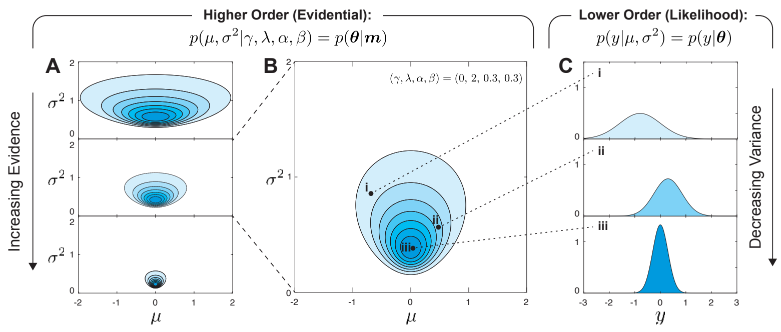


Figure 2.12: Depiction of the Normal Inverse Gamma and relation with evidence and uncertainty derivation. Image taken from [3]

To overcome the disadvantages of sampling-based methods, other approaches aim to directly provide uncertainty measures by training a model to estimate the parameters of a distribution. For instance, by assuming that we can model the uncertainty as a normal distribution $N(\mu, \sigma^2)$, a DNN can learn to produce μ and σ as outputs. Kendall and Gal [83] propose to incorporate *maximum likelihood estimation* (MLE) within the training process by defining the following loss function:

$$L(\Theta) = -\log p\left(y_i|\mu, \sigma^2\right) = \frac{1}{2}\log\left(2\mu\sigma^2\right) + \frac{y_i - \mu}{2\sigma^2} \tag{1}$$

where y_i is the target of inference. However, this method only estimates the noise present in the input, that is *it only deals with aleatoric uncertainty* and it is ignorant with respect to the epistemic uncertainty of the model used.

Other direct estimation approaches rely on *deep evidential learning*: a technique inspired by the Dempster–Shafer theory of evidence [41]. In this case, learning can be considered a data acquisition process that allows a model to learn the hyperparameters of an *evidential* distribution from which one can derive uncertainty. In particular, each sample fed to the network provide support (evidence) in favour of that distribution. Sensoy *et al.* [154] is one of the most relevant work in evidential deep learning for classification. In particular, their evidential-based model estimates the parameters of Dirichlet distribution over the possible softmax outputs. More recently, Amini *et al.* [3] propose a framework named Deep Evidential Regression (DER) to apply such a learning strategy within regression problems. In particular, their method estimates the parameters of a Normal Inverse Gamma (NIG) distribution $\left(x, \sigma^2\right) \sim \text{NIG}(\gamma, \nu, \alpha, \beta)$. Those parameters enable the computation of both aleatoric $E[\sigma^2] = \frac{\beta}{\alpha-1}$ and epistemic $Var[\mu] = \frac{\beta}{\nu(\alpha-1)}$ uncertainties together with the model prediction $E[\mu] = \gamma$. A schematic representation of the concepts at the base of DER is depicted in fig. 2.12.

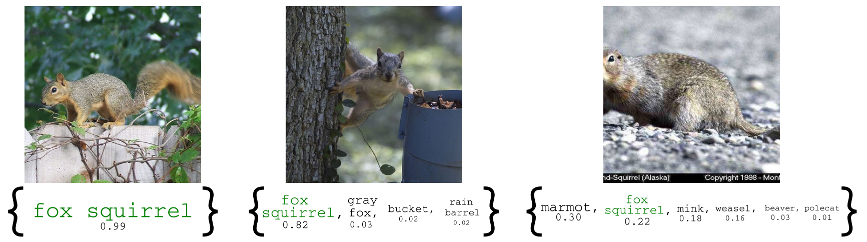

{ fox squirrel 0.99 }
{ fox squirrel 0.82, gray fox 0.03, bucket 0.02, rain barrel 0.02 }
{ marmot 0.30, fox squirrel 0.22, mink 0.18, weasel 0.16, beaver 0.03, polecat 0.01 }
nd-Squirrel (Alaska)
Copyright 1998 - Mon

Figure 2.13: Prediction sets produced by a conformal predictor. Image taken from [6].

Modeling uncertainty with a predefined distribution shape, *e.g.*, a Gaussian, can be a strong assumption, and we do not know if our choice represents a good approximation of the actual uncertainty distribution. To overcome this problem, distribution free approaches also started to emerge such as *Conformal Prediction* [5, 177]. Conformal Prediction is a statistical method that generates prediction sets from any model in a rigorous manner. The idea is to *calibrate* probabilities of a classifier, *e.g.*, softmax outputs of a neural network, by exploiting a limited amount of data X_{calib} never used during training, with the aim of providing more realistic confidence scores [6]. A prediction set $C(x_{test})$ includes a group of classes that are labeled by the conformal predictor as possibly correct. In particular, a "good" conformal predictor ensures that the true class is included within the prediction set $C(x_{test})$ with at least a probability of $1 - \alpha$, where α is an user defined error rate. This property also known as *coverage* holds only if the calibration set X_{calib} and the test set X_{test} are exchangeable, that is they are drawn from the same distribution. While some may view this aspect as a limitation, it is worth noting that it is possible to re-calibrate a deep learning model with minimal data. This could eliminate the need for re-training and allow for the production of meaningful class scores within a known application scenario. In Conformal Prediction, uncertainty is represented as the cardinality of the prediction set $C(x_{test})$, that is the uncertainty of a predictor increases when the number of possible true classes increases as well. In fig. 2.13, we can see that as the complexity of the scene increases, the same happens for the number of classes contained in the prediction set, which always includes the true class. There are also methods the apply Conformal Prediction in regression problems, taking the name of *Conformalized Quantile Regression* [94]. Conformal Prediction is not necessarily associated to DNN models, but it still finds some applications in deep learning and in very recent years some novel approaches emerged [6]. Although Conformal Prediction was not used in this thesis work, its basic concepts were reported to offer a more exhaustive overview.

UNCERTAINTY IN LOCALIZATION DNNS As mentioned in section 2.2, observer localization is a crucial task in autonomous driving, as inaccurate estimates could lead to navigation failures with disastrous consequences for road users. Therefore, enabling error detection in localization systems would improve their reliability and allow a better management of risky scenarios. Considering the trend of the last decade of equipping perception systems with deep learning methods, uncertainty estimation in DNNs could represent a fundamental tool even for localization systems.

One of the first relevant work in the field is Bayesian PoseNet [81]: a CNN that addresses the problem of camera pose regression taking an image as input. In particular, Kendall e Cipolla propose to exploit MCD sampling to estimate localization epistemic uncertainty. However, although their approach have had a large impact in the computer vision community, it presents two main disadvantages: camera-based localization methods suffer in terms of scalability, *i.e.*, they only work in the environment represented in the training set, and MCD generally produces overconfident uncertainty estimates [105]. In another work, the same authors also proposed a novel loss function to capture data uncertainty [82]. Another interesting uncertainty-aware approach for camera pose regression is proposed by Sarlin *et al.* [148]. The model does not directly estimate the camera's 6DoF. Instead, it matches deep image features with a 3D model of the scene, and the regression part is delegated to an external algorithm. The proposed CNN explicitly estimates aleatoric uncertainty maps that allow the authors to discard confounding image regions that are not useful for localization, such as dynamic objects. Peretoukhin *et al.* [133] propose a CNN named HydraNet to estimate epistemic uncertainty for $SO(3)$ rotations expressed with unit quaternions from a sequence of images. In particular, HydraNet architecture comprises two branches: one to perform visual odometry estimation and one for estimating camera rotations. The rotation branch involves multiple heads providing different rotation guesses, that one can consider as samples to use for estimating epistemic uncertainty. Recently, Petek *et al.* [134] propose a DNN model based on DER [3]. The model does not estimate the uncertainty of the predicted camera pose directly. Instead, localization is achieved through an external module. However, the approach identifies objects within the scene and associates both aleatoric and epistemic uncertainties with the final prediction. These uncertainties are then used in the localization module to match the detected objects with an HD map. An interesting application of uncertainty-aware camera localization was recently proposed by Moreau *et al.* [127]. Similarly to [81], the authors propose a camera-based localization model that also provide uncertainty of predicted pose components. Then, they integrate their approach within an Extended Kalman Filter (EKF). However, their method only estimates aleatoric uncertainty. As it can be seen, most of the mentioned approaches are camera-based and mainly offer the estimation of aleatoric uncertainty, while model uncertainty is often neglected or utilize overconfident methods such as MCD. Therefore, in this thesis work a multi-modal camera pose regression method is proposed chapter 5 by utilizing both sampling-based and direct uncertainty estimation approaches.

Other approaches localize an observer by performing place recognition, *i.e.,* a rough localization estimate is provided by matching an input observation with an already visited location. Place recognition is a well-discussed topic in the literature, but approaches that estimate uncertainty within deep learning-based place recognition only emerged recently and are still limited. A relevant work in the field is STUN [24]: the first work that estimates aleatoric uncertainty within image descriptors by also exploiting a knowledge-distillation technique. In particular, they train a neural network also named *teacher* to perform standard place recognition, then another model named *student* is trained to imitate the descriptors of the teacher. They provide a loss function that allows the student network to learn how to estimate uncertainty in data. Recently, Latoje *et al.* [104] propose a visual place recognition technique that estimate aleatoric descriptors uncertainty, which is utilized within a SLAM algorithm. The deep learning model learn to estimate aleatoric uncertainty as an isotropic Normal distribution. Finally, uncertainty estimation in place recognition was also recently tackled in LiDAR-based approaches [122]. In particular, Mason *et al.* propose an epistemic uncertainty estimation approach that relies on DEs. However, instead of computing feature-wise uncertainty in point cloud descriptors, they consider uncertainty as a degree of accordance of ensemble members with respect to a similarity measure.Uncertainty-aware deep learning based methods are quite novel in place recognition. Also for this problem, mainly camera-based approaches are present in the literature and mainly focus on aleatoric uncertainty, while a lack of methods that deal with other kind of observations, other type of uncertainty and DNN architecture is encountered. Therefore, during the PhD it was proposed an in-depth study of sampling-based method for epistamic uncertainty in LiDAR-based place recognition by focusing on DEs.

3 DNN-BASED LIDAR LOOP CLOSURE DETECTION AND REGISTRATION

So far in this thesis, the discussion was focused on a general introduction of the autonomous driving field together with the currently open challenges to be addressed for achieving fully-autonomous navigation. In particular, the main topic of this thesis is to improve the reliability of deep learning localization techniques with the final aim of deploying these systems in urban areas, where the presence of complex scene structures and several other users makes this task particularly challenging. Furthermore, since the configuration of urban areas usually undermines the performance of Global Navigation Satellite Systems (GNSSs) (Section 2.1.1), accurate localization is usually achieved by exploiting other sensors that are used to gather knowledge from the current scene. An example is the localization pipeline presented in Section 2.2, where we could observe that accurate localization is usually achieved in a series of consecutive steps, where we firstly aim to obtain a rough global initial estimate that is usually refined in a successive moment. The main scope of this thesis is to tackle the different parts of this pipeline by considering both the global and local refinement modules.

This chapter introduces a novel 3D Deep Neural Network (DNN) that faces both the tasks of place recognition, *i.e.,* global localization, and Light Detection And Ranging (LiDAR) point clouds registration, *i.e.,* local refinement. In particular, one the of primary application of such a model is a LiDAR-based Simultaneouos Localization And Mapping (SLAM) system, since the place recognition module can be used for detecting loop closures and the point cloud registration module can provide accurate localization estimates after the detection of loops. Therefore, the approach proposed in this chapter covers the whole localization pipeline by also addressing the re-visiting problem described previously (section 2.3.1).

A typical SLAM algorithm operates to ensure an accurate navigation and aims to reduce the accumulated drift affecting the localization estimates obtained from sensor measurements, which typically leads to failures during long-term navigation. Due to this reasons, SLAM has become a very popular method to overcome the flaws of sensors-based localization systems. However, the operations performed by the SLAM pipeline are critical, since failures usually have a negative impact in the subsequent activities of the vehicle. For instance, incorrect loop detections or inaccurate registrations of point clouds can degrade the localization quality due to the propagation of the pose error during the update of the previous robot trajectory. In particular, false loops, *i.e.,* wrong matches, often lead to unrecoverable localization errors, making loop closure detection a critical task.

Employing LiDARs within a SLAM algorithm has several advantages, since the performance of these sensors do not suffer from the impact of external factors such as the presence of poor illumination. Furthermore, since they provide accurate 3D reconstruction of the surrounding scene, they demonstrate to be particularly effective for mapping a navigation environment. Due to these reasons, LiDAR-based techniques are preferred over visual-based methods [40, 128], that often fail in challenging conditions. However, despite the high performance achieved by LiDAR-based approaches, the accuracy of these methods drastically decreases in presence of reverse loops, that is observations depicting a previously visited location from a different point of view that makes its recognition particularly challenging. Also registration methods suffer of the same problem. Therefore, the model proposed in this chapter deals with these two specific issues.

The contributions reported in this chapter are the following. A novel LCDNet for loop closure detection which performs both loop detection and point cloud registration is proposed (see Figure 3.1). This method combines the ability of DNNs to extract distinctive features from point clouds, with algorithms from the transport theory for feature matching. LCDNet is composed of a shared backbone that extracts point features, followed by the place recognition head that extracts global descriptors and the relative pose head that estimates the transformation between two point clouds. One of the core components of the proposed LCDNet is the Unbalanced Optimal Transport (UOT) algorithm that was implemented in a differentiable manner. UOT allows us to effectively match the features extracted from the two point clouds, reject outliers, and handle occluded points, while still being able to train the network in an end-to-end manner. As opposed to existing loop closure detection methods that estimate the relative yaw rotation between two point clouds, the proposed LCDNet estimates the full 6-DoF relative transformation under driving conditions between them which significantly helps the subsequent Iterative Closest Point (ICP) refinement to converge faster.

The proposed LCDNet is trained on sequences from the KITTI odometry [55] and KITTI-360 [189] datasets, and evaluate it on the unseen sequences on both datasets. Moreover, following a deep analysis of the existing literature, a lack of a standard protocol for evaluating loop closure detection was found. Different works evaluated their approaches using different metrics such as precision-recall curve, average

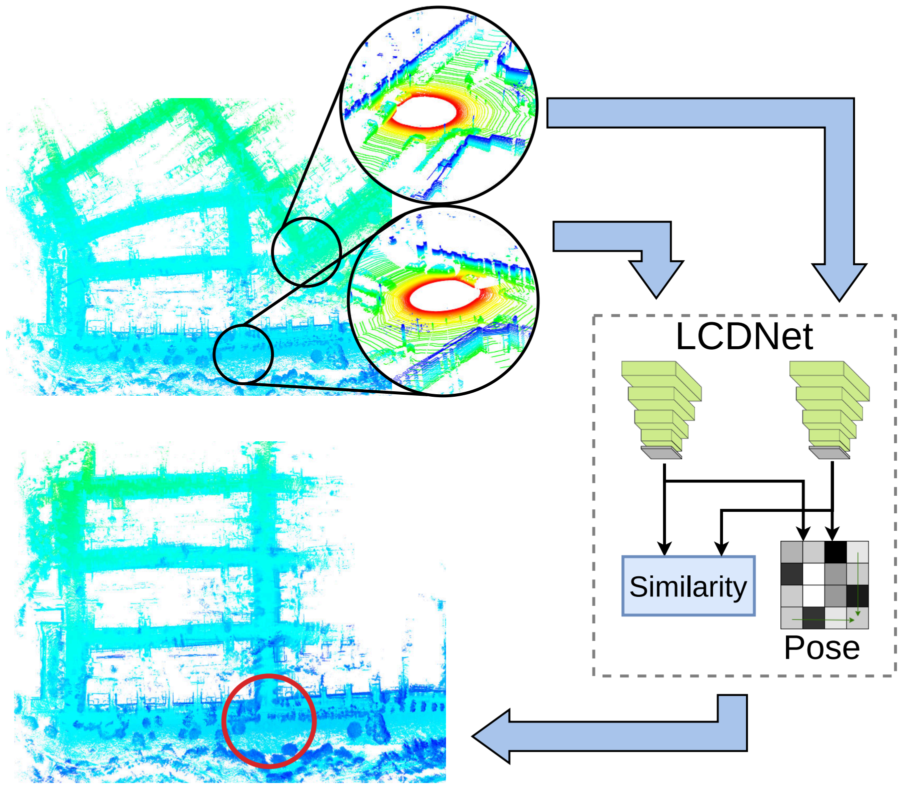


Figure 3.1: The proposed LCDNet detects loops by computing the similarity between two point clouds and predicting the relative pose between them. This is a crucial component of any SLAM system, as it reduces the drift accumulated over time.

precision, Receiver Operating Characteristic (ROC) curve, recall@k, and maximum F1-score. Even among the methods that use the same metric for evaluation, there are still substantial differences in the other parameters chosen for computing the metrics which makes the performance of existing methods not directly comparable. For example, the definition of a true loop can span from scans within three meters [99] up to scans within 15 meters [205]. Therefore, in this work, existing state-of-the-art approaches are evaluated according a uniform evaluation protocol to provide a fair comparison. Exhaustive comparisons demonstrate that the proposed LCDNet outperforms both handcrafted methods as well as DNN-based methods and achieves state-of-the-art performance on both loop closure detection and point cloud registration tasks. Furthermore, a detailed ablation studies on the architectural topology of LCDNet is reported together with the results obtained from the integration of LCDNet into a recent LiDAR SLAM library [157]. Additionally, an assessment of the generalization ability of the proposed approach is provided by performing experiments with a different sensor setup from an autonomous driving scenario in a completely different city.

3.1 RELATED WORK

In this section, an overview of the state-of-the-art techniques for vision-based and LiDAR-based loop closure detection is provided, followed by methods for point cloud registration.

Loop Closure Detection: Techniques for loop closure detection can primarily be categorized into visual and LiDAR-based methods. Traditionally, vision-based techniques for loop closure detection rely on handcrafted features for identifying and representing relevant parts of scenes depicted within images, and exploit a Bag-of-Words model to combine them [40, 128]. In the last few years, deep learning approaches [8, 198] have been proposed that achieve successful results. These techniques employ DNN for computing global descriptors to provide a compact representation of images and perform direct comparisons between descriptors for searching matches between similar places. Recently, [114] proposed a novel approach that employs DNNs for extracting local features from intermediate layers and organizes them in a word-pairs model. Although vision-based methods achieve impressive performance, they are not robust against adverse environmental situations such as challenging light conditions and appearance variations that can arise during long-term navigation. As loop closure detection is a critical task within SLAM systems, in this work, LiDARs are used for the sensing modality since they provide more reliable information even in challenging conditions in which visual systems fail.

3D LiDAR-based techniques have gained significant interest in the last decade, as LiDARs provide rich 3D information of the environment with high accuracy and their performance is not affected by illumination changes. Similar to vision-based approaches, LiDAR-based techniques also exploit local features. Most methods use 3D keypoints [93, 201] that are organized in a bag-of-words model for matching point clouds [165]. [18] propose a keypoint based approach in which a nearest neighbor voting paradigm is employed to determine if a set of keypoints represent a previously visited location. Recently, [162] propose a voxel-based method that divides a 3D scan into voxels and extracts multiple features from them through different modalities, followed by learning the importance of voxels and types of features.

Another category of techniques represents point clouds through global descriptors. [66] propose an approach that directly produces point clouds fingerprints. In particular, this method relies on density signatures extracted from multiple projections of 3D point clouds on different 2D planes. [86] introduces a novel global descriptor called Scan Context that exploits bird-eye-view representation of a point cloud together with a space partitioning procedure to encode the 2.5D information within an image. In a similar approach, [181] propose a method to extract binary signature images from 3D point clouds by employing LoG-Gabor filtering with thresholding operations to obtain a descriptor. The main drawback of these approaches is that they require an ad-hoc function to compare the global descriptor of two point clouds which drastically impacts the runtime when the number of scans to compare increases.

Recently, DNN-based techniques have also been proposed for computing descriptors from 3D point clouds. [4] propose PointNetVLAD which is composed of PointNet [138] with a NetVLAD layer [8] and yields compact descriptors. [150] propose OREOS which computes 2D projection of point clouds on cylindrical planes and is subsequently fed into a DNN that computes global descriptors and estimate their yaw discrepancy. More recently, the OverlapNet [36] architecture was introduced, which estimates the overlap and relative yaw angle between a pair of point clouds. The overlap estimate is then used for detecting loop closures while the yaw angle estimation is provided to the Iterative Closest Point (ICP) algorithm as the initial guess for the point clouds alignment. While DNN-based methods are generally faster than classical techniques, and show promising results in sequences that contain loops only in the same direction, their performance drastically decreases when they are faced with reverse loops.

Recently, techniques that exploit graph structures by matching semantic graphs have been proposed [99, 205]. These approaches first extract semantic information and perform instance retrieval, followed by defining graph vertices on the object centroids. Subsequently, features are extracted by considering handcrafted descriptors or by processing nodes through a Dynamic Graph CNN [183]. Finally, loop closures are identified by comparing vertices between graphs. However, computing the exact correspondences between two graphs is still an open problem and existing methods are only suitable when a few vertices are considered or they can only provide an approximated solution [12]. In this chapter, the proposed DNN-based approach exploits the recent advancements in deep learning to detect loop closures by combining high-level voxel features with fine-grained point features. This method effectively detects loops in challenging scenarios such as reverse loops and outperforms state-of-the-art handcrafted and learning-based techniques.

Point Cloud Registration: Point clouds registration represents the task of finding a rigid transformation to accurately align a pair of point clouds. The ICP algorithm [199] is one standard method that is often employed to tackle this task. Although ICP is one of the most popular methods, the main drawback concern the initial rough alignment of point clouds which is required to reach an acceptable solution, and

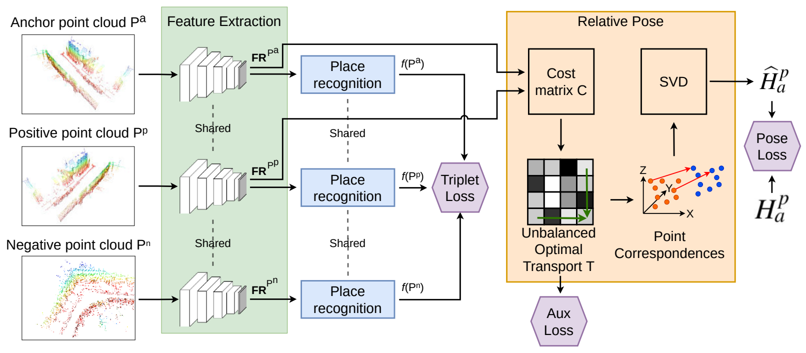


Figure 3.2: Overview of the proposed LCDNet which is composed of a shared feature extractor (green), a place recognition head (blue) that generates global descriptors, and a relative pose head (orange) that estimate the transformation between two point clouds. Three loss functions are used to train LCDNet (triplet loss, aux loss, and pose loss) which are depicted in purple. The topology of the feature extractor is further illustrated in Figure 3.3.

the algorithm complexity which increases drastically with the number of points. Other methods tackle the registration problem globally without requiring a rough initial alignment. Traditionally, these techniques exploit local features [145] for finding matches between point clouds and employ algorithms such as RANdom SAmple Consensus (RANSAC) [48] for estimating the final transformation. However, the presence of noise in the input data and outliers generated from incorrect matches can lead to an inaccurate result. To address these problems, [202] proposes a global registration approach that ensures fast and accurate alignment, even in the presence of many outliers.

Recent years have also seen the introduction of deep learning methods that tackle the registration problem. A typical approach is to employ a DNN for extracting features which are then used in the later stages to perform point clouds alignment. [7] propose such an approach known as PointNetLK, which exploits the PointNet [138] architecture for feature extraction and employs a variation of the Lucas and Kanade algorithm [119] to perform registration. Deep Closest Point (DCP) [182] is another approach that employs a Siamese architecture, attention modules, and differentiable Singular Value Decomposition (SVD) to regress a rigid transform for aligning two input point clouds. Recently, [192] proposes a DNN-based method called RPM-Net which is inspired by Robust Point Matching (RPM). RPM-Net employs two different neural networks to extract features and predict annealing parameters that are required for RPM. However, these methods are only capable of aligning point clouds that are relatively close to each other (up to 45° rotation misalignment), and completely fail to register point clouds that are more than 120° apart [7]. In contrast to the aforementioned methods, the approach proposed in this chapter does not require any initial guess as input and can handle both outliers and occluded points. Moreover, unlike existing DNN-based methods, the approach effectively aligns point clouds with arbitrary initial rotation misalignment.

3.2 PROPOSED APPROACH

This section provides the details of the proposed LCDNet for loop closure detection and point cloud registration from LiDAR point clouds. An overview of the proposed approach is depicted in Figure 3.2. The network consists of three main components: feature extraction, global descriptor head, and 6-DoF relative pose estimation head. The following discussion is organized by providing a description for each of the aforementioned components and the associated loss functions for training, followed by the approach for integrating LCDNet into the SLAM system.

3.2.1 *Feature Extraction*

The feature extractor stream of the proposed network is based on the PV-RCNN [159] architecture that was proposed for 3D object detection. PV-RCNN effectively combines the ability of voxel-based methods for extracting high-level features, with fine-grained features provided by PointNet-type architectures. However, LCDNetpresents several differences with respect to the standard architecture to adapt it to the task addressed in this chapter. An illustration of the topology of our adapted PV-RCNN in Figure 3.3.

The input to the network is a point cloud $P \in \mathbb{R}^{J x 4}$ (J points with 4 values each: x, y, z, and intensity). The output of the proposed feature extractor network is a set of N keypoints' feature $\mathbf{FR}^P = \{fr_1^P, \dots, fr_N^P\}$, where $fr_i^P \in \mathbb{R}^D$ is the D-dimensional feature vector for the i-th keypoint. Since the backbone architecture is used for extracting features, and not for detecting objects, LCDNetonly retains the 3D voxel DNN and the Voxel Set Abstraction (VSA) module, discarding the region proposal network, the ROI-grid pooling, and the fully connected layers towards the end of the architecture. The 3D voxel DNN first converts the point cloud into a voxel grid of size $L \times W \times H$, where voxel features are averaged across all the points that lay within the same voxel. Subsequently, a sequence of sparse 3D convolutions and downsampling operators extract a feature pyramid. From a technical perspective, LCDNetemploys four pyramid blocks composed of 3D sparse convolutions, with downsampling rates of $1\times$, $2\times$, $4\times$, and $8\times$, respectively. Finally, the coarsest feature map is

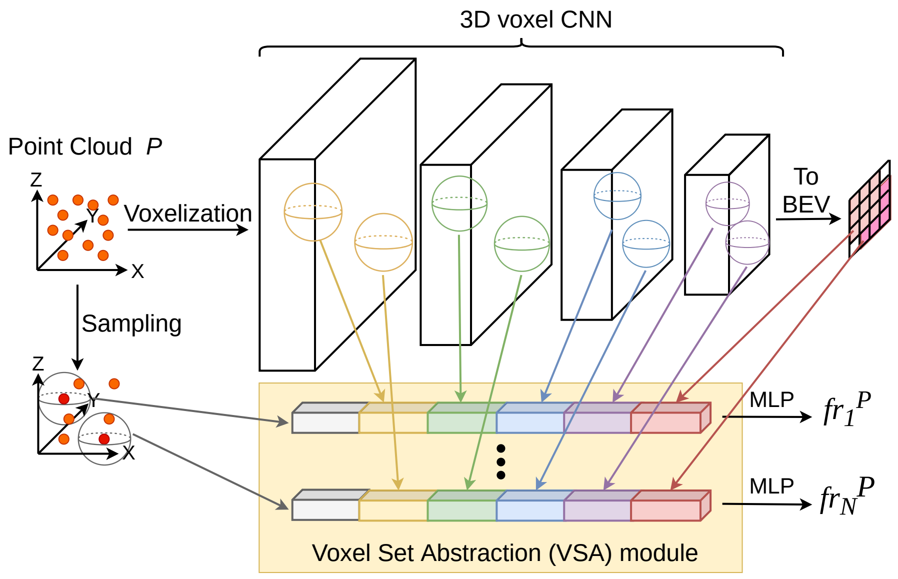


Figure 3.3: Network topology of the PV-RCNN architecture upon which LCDNetis based for the feature extractor component of the proposed proposed LCDNet.

converted into a 2D Bird's-Eye-View (BEV) feature map by stacking the features along the Z axis.

The VSA module, on the other hand, aggregates all the pyramid feature maps together with the BEV feature map and the input point cloud into a small set of N keypoints features. To do so, first the Farthest Point Sampling (FPS) algorithm [59] performs downsampling of the point cloud by selecting N uniformly distributed keypoints. The VSA module is an extension of the Set Abstraction (SA) level [139]. The standard SA aggregate neighbors point features in the raw point cloud, whereas, the VSA aggregate neighbors voxel features in the 3D sparse feature map. For every selected keypoint kp_i, and every layer l of the pyramid feature map, the keypoint features f_i^l are computed as

$$f_i^l = \mathrm{MP}(\mathrm{MLP}(\mathcal{M}(S_i^l))), \tag{2}$$

where MP is the max-pooling operation, MLP denotes a MultiLayer Perceptron (MLP), and $\mathcal{M}$ randomly samples the set of neighbor voxel features S_i^l, which is computed as

$$S_i^l = \left\{ \left[fvox_j^l ; v_j^l - kp_i \right] ; \text{s. t. } \left\| v_j^l - kp_i \right\|^2 < r \right\}, \tag{3}$$

where $fvox_j^l$ is the feature of the voxel j at level l, v_j^l denotes the coordinates of the voxel j at level l, and r is the neighbor radius. This operation is performed at every level of the pyramid to yield

$$f_i^{pv} = \left[f_i^1, f_i^2, f_i^3, f_i^4 \right]. \tag{4}$$

We perform a similar operation for the input raw point cloud, as well as the BEV feature map, yielding the aggregated keypoint features

$$f_i^{3D} = \left[f_i^{pv}, f_i^{raw}, f_i^{bev}\right]. \tag{5}$$

Lastly, a MLP computes the final keypoint feature vectors strating from the aggregated keypoint features:

$$fr_i = \text{MLP}(f_i^{3D}). \tag{6}$$

As opposed to the original PV-RCNN that processes only the points that lay in the camera Field Of View (FOV), the task addressed by LCD-Netrequires the full 360° surrounding view. Therefore, the proposed model employs a voxel grid size of $\pm 70.4\,\text{m}$, $\pm 70.4\,\text{m}$ and $[-1\,\text{m}, 3\,\text{m}]$ in the x, y and z dimensions, respectively, with a voxel resolution size of $0.1\,\text{m} \times 0.1\,\text{m} \times 0.1\,\text{m}$. A demonstration of the discriminative power of the proposed feature extractor will be provided in Section 3.3.6, by comparing it with different state-of-the-art backbones. Moreover, we also investigate the best choice for the dimensionality D of the keypoint features.

3.2.2 *Global Descriptor*

In order to generate a global descriptor for a given point cloud, the keypoints' feature set $\mathbf{FR}^P$ obtained from the feature extractor are aggregated into a compact G-dimensional vector. To do so, LCD-Netfirst employs the NetVLAD layer [8] which converts the (N x D)-dimensional $\mathbf{FR}^P$ set into a (K x D)-dimensional vector $\mathbf{V}(\mathbf{FR}^P)$ by learning a set of K cluster centers $\{c_1, \ldots, c_K\}$, $c_k \in \mathbb{R}^D$. NetVLAD mimics the original Vector of Locally Aggregated Descriptors (VLAD) [75] using differentiable operations. It replaces the k-means clustering with learnable clusters and replacing the hard assignment with a soft assignment defined as

$$a_k(fr_i^P) = \frac{e^{\mathbf{w}_k^\top fr_i^P + b_k}}{\sum_{k'=1}^{K} e^{\mathbf{w}_{k'}^\top fr_i^P + b_{k'}}}, \tag{7}$$

where $\mathbf{w}_k \in \mathbb{R}^D$ and $b_k \in \mathbb{R}$ are the learnable weights and bias. In practice, $a_k(fr_i^P)$ represents the probability of assigning the feature vector fr_i^P to the cluster center c_k. The final NetVLAD descriptor $\mathbf{V}(\mathbf{FR}^P) = [\mathbf{V}_1(\mathbf{FR}^P), \ldots \mathbf{V}_K(\mathbf{FR}^P)]$ is computed by combining the original VLAD formulation with the soft assignment defined in eq. (7) as

$$\mathbf{V}_k(\mathbf{FR}^P) = \sum_{i=1}^{N} a_k(fr_i^P)(fr_i^P - c_k). \tag{8}$$

Since the NetVLAD layer has demonstrated superior performance for point cloud retrieval [4] than the max-pooling employed in PointNet [138], it is also utilized in the proposed model. To further reduce the dimensionality of the final global descriptor, the global descriptor head employs a simple MLP that compresses the $(K \times D)$-dimensional vector $\mathbf{V}(\mathbf{FR}^P)$ into a G-dimensional compact descriptor. We then obtain the final global descriptor $f(P) \in \mathbb{R}^G$ by employing the Context Gating (CG) module [125] on the output of the MLP. The CG module re-weights the output of the MLP using a self-attention mechanism as

$$Y(X) = \sigma(WX + b) \odot X, \tag{9}$$

where X is the MLP output, σ is the element-wise sigmoid operation, $\odot$ is the element-wise multiplication, W and b are the weights and bias of the MLP. The CG module captures dependencies among features by down-weighting or up-weighting features based on the *context* while considering the full set of features as a whole, thus focusing the attention on more discriminative features.

3.2.3 *Relative Pose Estimation*

Given two point clouds P and S, the third component of the proposed architecture estimates the 6-DoF transformation to align the source point cloud P with the target point cloud S under driving conditions. This task is performed by matching the keypoints' features $\mathbf{FR}^P$ and $\mathbf{FR}^S$ computed using the adapted feature extractor from section 3.2.1. Due to the sparse nature of LiDAR point clouds and the keypoint sampling step which is performed in the feature extractor, a point in P might not have a single matching point in S, but it can lay in between two or more points in S. Therefore, a one-to-one mapping is not desirable in our task.

In order to address this problem, the proposed mehtod employs the Sinkhorn algorithm [161], which can be used to approximate the *optimal transport* (OT) theory in a fast, highly parallelizable and differentiable manner. Recent work has shown benefits of using the Sinkhorn algorithm with DNNs for several tasks such as feature matching [147], scene flow [137], shape correspondence [44], and style transfer [95]. The discrete Kantorovich formulation of the optimal transport is defined as

$$T = \underset{A \in \mathbb{R}^{N \times N}}{\arg\min} \left\{ \sum_{i,j} C_{ij} A_{ij};\ \text{s. t. } A \text{ is doubly stochastic} \right\}, \tag{10}$$

where C_{ij} is the cost of matching the i-th point in P to the j-th point in S. In order to employ the Sinkhorn algorithm, an entropic regularization term is added to the formula:

$$T = \underset{A \in \mathbb{R}^{N \times N}}{\arg\min} \left\{ \sum_{i,j} C_{ij} A_{ij} + \lambda A_{ij} \left(\log A_{ij} - 1 \right) \right\}, \tag{11}$$

where λ is a parameter that controls the sparseness of the mapping (as $\lambda \to 0$, T converges to a one-to-one mapping). However, both eqs. (10) and (11) are subject to A being a doubly stochastic matrix (mass preservation constraint), *i.e.*, every point in $\mathbf{FR}^P$ has to be matched to one or more points in $\mathbf{FR}^S$, and vice versa. In our point cloud matching task, some points in $\mathbf{FR}^P$ might not have a matching in $\mathbf{FR}^S$, for example when a car is present in one point cloud but is absent in the other, or in the case of occlusions. Therefore, we need to relax the mass prevention constraint. One common approach to overcome this problem is by adding a dummy point in both P and S (*i.e.*, add a dummy row and column to A). Another way is to reformulate the problem as *unbalanced optimal transport* (UOT) which allows mass creation and destruction, and is defined as

$$\begin{aligned} T = \underset{A \in \mathbb{R}^{N \times N}}{\arg\min} \Bigg\{ \left(\sum_{i,j} C_{ij} A_{ij} + \lambda A_{ij} \left(\log A_{ij} - 1 \right) \right) + \\ \rho \left(\mathrm{KL} \left(\sum_{i} A_{ij} | U(1,N) \right) + \mathrm{KL} \left(\sum_{j} A_{ij} | U(1,N) \right) \right) \Bigg\}, \end{aligned} \tag{12}$$

where KL is the Kullback–Leibler divergence, U is the discrete uniform distribution, and ρ is a parameter that controls how much mass is preserved. The UOT formulation, compared to the standard OT, reduces the negative effect caused by incorrect point matching and is more robust to the stochasticity induced by keypoint sampling [45]. A recent extension to the Sinkhorn algorithm [38] that approximates the unbalanced optimal transport is shown in algorithm 1. In this work, the cost matrix C is set as the cosine distance between the keypoints' features $C_{ij} = 1 - {}^{FR_i^P \cdot FR_j^S}/_{\|FR_i^P\| \|FR_j^S\|}$. Instead of setting λ and ρ manually, LCDNetlearns them using back propagation.

After the estimation of the unbalanced optimal transport T, which represents the set of soft correspondence between keypoints' features $\mathbf{FR}^P$ and $\mathbf{FR}^S$, together with their respective 3D keypoints' coordinates P and S, for every keypoint $p_j \in P$ its projected coordinates in S is computed as

$$\hat{s}_j = \frac{\sum_{k=1}^{K} T_{jk} s_k}{\sum_{k=1}^{K} T_{jk}}. \tag{13}$$

Finally, the weighted SVD estimates the rigid body transformation between the original point cloud P and its projection $\hat{S}$ in S. Since both

Algorithm 1: Unbalanced Optimal Transport

```
Data: Cost matrix $C$, number of iterations $L$, parameters $\lambda$ and $\rho$
Result: Unbalanced Optimal Transport $T$
begin
    $K \leftarrow e^{-C/\lambda}$
    $a \leftarrow \mathbb{1}_N/N$
    $b \leftarrow \mathbb{1}_N/N$
    $v \leftarrow \mathbb{1}_N/N$
    for $i \leftarrow 1$ to $L$ do
        $u \leftarrow [a \oslash (Kv)]^{\rho/(\rho+\lambda)}$
        $v \leftarrow [b \oslash (K^T u)]^{\rho/(\rho+\lambda)}$
    end
    $T \leftarrow u \odot K \odot v^T$
end
```

where $\oslash$ is the element wise division, and $\odot$ is the element-wise multiplication.

algorithm 1 and SVD are differentiable, the proposed relative pose head is trained in an end-to-end manner by comparing the predicted transformation $\widehat{H}_P^S$ with the groundtruth transformation H_P^S.

Once the network has been trained, the UOT-based relative position head is replaced with a RANSAC-based registration method that exploits the features extracted by LCDNetto find correspondences. In this way, we can train the network in an end-to-end manner, and at the same time estimate accurate relative poses using the robust RANSAC estimator during inference.

3.2.4 *Loss Function*

Regarding the global descriptor head, LCDNetutilizes the triplet loss [152] during training. Given an anchor point cloud P^a, a positive sample P^p (point cloud of the same place), and a negative sample P^n (point cloud of a different place), the triplet loss enforce the distance between the descriptors of positive samples to be smaller than the distance between negative samples descriptors. More formally, the triplet loss is defined as

$$\mathcal{L}_{trp} = [d(f(P^a), f(P^p)) - d(f(P^a), f(P^n)) + m]_+, \tag{14}$$

where $d(\cdot)$ is a distance function, m is the desired separation margin, and $[x]_+$ means $max(0, x)$.

Instead of selecting the triplets in advance (offline mining) for every anchor in the batch, the proposed training procedure randomly selects a positive sample, and then randomly selects the corresponding negative sample from all the batch elements that depict a different place (online negative mining). The relative pose transformation is only computed for positive pairs, and the model is trained by comparing the anchor point cloud $P^a = \{p_1^a, \ldots, p_J^a\}$ transformed using the predicted transformation $\widehat{H}_a^p$ and the groundtruth transformation H_a^p as

$$\mathcal{L}_{pose} = \frac{1}{J}\sum_{j=1}^{J}\left|\widehat{H}_a^p p_j^a - H_a^p p_j^a\right|. \tag{15}$$

In addition, the proposed method employs an auxiliary loss on the matches estimated by the unbalanced optimal transport T as

$$\mathcal{L}_{OT} = \frac{1}{J}\sum_{j=1}^{J}\left|\frac{\sum_{k=1}^{K} T_{jk} p_k^p}{\sum_{k=1}^{K} T_{jk}} - H_a^p p_j^a\right|. \tag{16}$$

The final loss function is a linear combination of the three aforementioned components:

$$\mathcal{L}_{total} = \mathcal{L}_{trp} + \mathcal{L}_{pose} + \beta\mathcal{L}_{OT}, \tag{17}$$

where β is a loss balancing term which is empirically set to 0.05. Consequently, due to the combination of triplet loss, UOT, and data augmentation, the shared feature extractor learns to yield distinctive, rotation and translation invariant keypoints' features through backpropagation.

3.2.5 *SLAM System*

We integrate the proposed LCDNet into a recently proposed SLAM system, namely LIO-SAM [157] which achieves state-of-the-art performance on large-scale outdoor environments. LIO-SAM is a tightly coupled LiDAR inertial odometry framework built atop a factor graph. The framework takes a LiDAR point cloud and Inertial Measurement Unit (IMU) measurements as input. It includes four types of constraints that are added to the factor graph: IMU preintegration, LiDAR odometry, Global Positioning System (GPS) measurements (optional), and loop closure. In order to reduce the computational complexity, LIO-SAM selectively chooses LiDAR scans as keyframes only when the robot moves more than a predefined threshold since the last saved keyframe. The scans in between two keyframes are then discarded. Moreover, LCDNetis used for detecting loop closures instead of the Euclidean distance-based loop closure detection module provided in LIO-SAM. From a technical perspective, for every keyframe $\mathbb{F}_i$ added to the LIO-SAM factor graph, the proposed model computes and stores the respective global descriptor $f(\mathbb{F}_i)$ in a database. When a new keyframe $\mathbb{F}_{i+1}$ is added to the graph, the point cloud with the most similar descriptor (excluding the past M keyframes) is retrieved from the database:

$$W = \underset{j \in \{1,\dots,i-M\}}{\arg\min} \left\| f(\mathbb{F}_{i+1}) - f(\mathbb{F}_j) \right\| . \tag{18}$$

If the distance between the two descriptors is below a certain threshold th, $\mathbb{F}_W$ is labeled as a loop candidate, and the estimation of the 6-DoF transformation between the two point clouds $\widehat{H}^W_{i+1}$ is provided by the relative pose head as described in section 3.2.3. Finally, ICP performs a further refinement of the transformation by employing $\widehat{H}^W_{i+1}$ as initial guess, and another threshold th_{icp} determines if the loop closure factor can be added to the pose graph according to the ICP fitness score. By using this additional geometric consistency check, we can discard the few remaining false positive detection. It is important to note that no IMU nor GPS measurements are used in the loop detection step.

3.3 EXPERIMENTAL EVALUATION

The experimental activity presented in this section will initially describe the datasets involved in the evaluation, the implementation details and the training protocol. Then, the presented quantitative and qualitative results from experiments will demonstrate that the proposed LCDNet can (i) effectively detect loop closures even in challenging condition such as loops in the reverse direction, (ii) align two point clouds without any prior initial guess, (iii) robustly align point clouds that only partly overlap, (iv) provide an accurate initial guess for further ICP alignment, (v) integrate with an existing SLAM system to provide a fully featured localization and mapping framework, (vi) generalize to unseen environments.

3.3.1 *Datasets*

The evaluation of the proposed approach considers three different autonomous driving datasets. A detailed list of sequences that were used for training and testing, together with the respective number of loop closures and route direction of revisited places are reported in table 3.1. Note that this list does not include the sequences without loops.

KITTI: The KITTI odometry dataset [55] contains 11 sequences with LiDAR point clouds and groundtruth poses, six of which contain loops. However, the groundtruth for some of these sequences is not aligned to nearby loop closures. Therefore, in this work the groundtruth provided with the SemanticKITTI dataset [15] is used, which is consistent for

Table 3.1: Statistics of evaluation datasets.

	KITTI					
	00	05	06	07	08	09
Num. of scans	4541	2761	1101	1101	4071	1591
Num. of loops	790	492	69	97	334	18
Num. of pairs	10 499	6534	2138	2497	2960	252
Route direction	*Same*	*Same*	*Same*	*Same*	*Reverse*	*Same*
% Reverse Loops	3%	5%	0%	0%	100%	0%

	KITTI-360						Freiburg
	00	02	04	05	06	09	-
Num. of scans	10 514	18 235	11 052	6291	9186	13 247	25 612
Num. of loops	2452	4690	2218	2008	2433	4670	13 851
Num. of pairs	24 499	43 894	21 165	20 361	22 822	53 858	~ 411M
Route direction	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*
% Reverse Loops	67%	87%	92%	88%	61%	46%	20%

all the sequences. Most of the KITTI odometry sequences contain loop closures from the same driving direction, except for sequence 08 which contains reverse loop closures. The proposed approach is evaluated on sequences 00 and 08 as they contain the highest number of loops and reverse loops, respectively.

KITTI-360: The recently released KITTI-360 dataset [189] consists of nine sequences, six of which contain loops. KITTI-360 contains more loops and reverse loops than the standard KITTI dataset (see table 3.1). The proposed approach is evaluated on two of the sequences in KITTI-360 that contain the highest number of loop closures: sequence 02 and sequence 09.

Freiburg: We recorded our own dataset by driving around the city of Freiburg, Germany, across different days. We used a car equipped with a Velodyne HDL-64E LiDAR sensor and an Applanix POS LV positioning system. The resulting dataset includes many loops, both from the same and reverse directions. Moreover, differently from the KITTI and KITTI-360 datasets, our Freiburg dataset includes many dynamic objects. The Freiburg dataset is thus used to evaluate the generalization ability of our approach to a different city, different sensor setup, and across different days by training the models on KITTI and KITTI-360, and evaluating them on our own dataset collected in Freiburg, without any re-training or fine-tuning.

3.3.2 *Implementation and Training Details*

Following [86], in this work two point clouds represents a real loop if the distance between the groundtruth poses is less than four meters. Moreover, loop candidates are not searched in the past 50 scans to avoid detecting loops in nearby scans. LCDNet is trained on sequences 05, 06, 07, and 09 of the KITTI dataset, validate it on sequences 00 and 08, and test it on the KITTI-360 dataset. A second model, denoted as **LCDNet**$_\dagger$, is trained on sequences 00, 04, 05, and 06 of the KITTI-360 dataset, validated on sequences 02 and 09, and tested on the KITTI dataset.

All models are trained for 150 epochs on a server with 4 NVIDIA TITAN RTX GPUs, using a batch size of 24 positive pairs. The ADAM optimizer is used to update the weights of the network, with an initial learning rate of 0.004 which is halved after epochs 40 and 80, and a weight decay of $5 \cdot 10^{-6}$. In all the experiments, if not otherwise specified, the number of considered keypoints is $N = 4096$, the intermediate feature dimension $D = 640$, the output feature dimension $G = 256$, the number of NetVLAD clusters $K = 64$, the triplet margin $m = 0.5$, and the distance function in eq. (14) as the L2 distance. The number of iterations for the Sinkhorn algorithm is set to $L = 5$.

In order to help the network to learn viewpoint-invariant features, a random rigid body transformation is applied to each point cloud, with a maximum translation of $[\pm 1.5\,\text{m}]$ on the x and y axes, and $[\pm 0.25\,\text{m}]$ on the z axis; the maximum rotation of $[\pm 180°]$ for the yaw (to simulate loop closures from different directions), and $[\pm 3°]$ for roll and pitch.

3.3.3 *Evaluation of Loop Closure Detection*

To evaluate the loop closure detection performance of LCDNet, precision-recall curves and the Average Precision (AP) metric are proposed by considering two different evaluation protocols.

Protocol 1: In the first protocol, the proposed assessment of LCDNetconsiders a real loop closure setting. For each scan i of the sequence, the similarity between the global descriptor $f(P^i)$ and the descriptor of all the previous scans is computed, excluding the nearby scan as detailed in section 3.3.1. The scan j with the highest similarity is selected as the loop candidate, and if the similarity between the two descriptors is higher than a threshold th, then the pair (i, j) is considered as a loop. In such a case, a further check of the distance of the groundtruth poses between the two scans is performed: if the distance is less than four meters, then the the matched pair is a true positive, and a false positive otherwise. On the other hand, if the similarity is lower than

Table 3.2: Comparison with the state of the art in terms of the average precision evaluated on the KITTI and KITTI-360 datasets.

	Method	Protocol 1			
		KITTI		KITTI-360	
		00	08	02	09
Handcrafted	M2DP [66]	0.93	0.05	0.15	0.66
	Scan Context [86]	0.96	0.65	0.81	0.90
	ISC [178]	0.83	0.31	0.41	0.65
	LiDAR-Iris [181]	0.96	0.64	0.83	0.91
DNN-based	OverlapNet [36]	0.95	0.32	0.14	0.70
	SG_PR [99]	0.49	0.13	-	-
	LCDNet	0.97	0.94	0.95	0.98
	LCDNet$_{\dagger}$	**0.998**	**0.96**	**0.97**	**0.99**
	Method	Protocol 2			
		KITTI		KITTI-360	
		00	08	02	09
Handcrafted	M2DP [66]	0.31	0.01	0.03	0.17
	Scan Context [86]	0.47	0.21	0.32	0.31
	ISC [178]	0.14	0.05	0.03	0.04
	LiDAR-Iris [181]	0.42	0.17	0.25	0.26
DNN-based	OverlapNet [36]	0.60	0.20	0.05	0.33
	SG_PR [99]	0.23	0.13	-	-
	LCDNet	0.62	0.73	0.69	0.79
	LCDNet$_{\dagger}$	**0.89**	**0.76**	**0.73**	**0.80**

the threshold, but if a scan within four meters around the current scan i exists, then it is considered as a false negative.

Protocol 2: For each scan, the second protocol takes into account all the previous scans, not only the one with the highest similarity. For every pair of scans, if the similarity between the two descriptors is higher than the threshold, the pair is labeled as loop closure, and a comparison against the groundtruth is performed to compute precision and recall. Although in a real-world loop closure application only the most similar scan matters, if an approach is able to detect loops when the scans are very similar, but fails in more challenging scenarios (such as occlusions), this will not be reflected in the protocol 1 results. In protocol 2, on the other hand, all pairs of scans are considered, and thus approaches that better deal with challenging situations will achieve better results. Also in this protocol, nearby scans are ignored to avoid matching consecutive scans.

In both protocols, the variation of the threshold th produces different set of pairs (precision, recall), that are used to generate the precision-recall curve and to compute the AP.

The proposed approach is compared with state-of-the-art hand-crafted methods: M2DP [66], Scan-Context [86], Intensity Scan-Context (ISC) [178], and LiDAR-IRIS [181], as well as DNN-based methods OverlapNet [36], and Semantic Graph Place Recognition (SG_PR) [99]. For all these approaches, the official code published by the respective authors and the pretrained models provided for DNN-based methods are used. OverlapNet only provides the model trained with geometric information, we refer to this model as OverlapNet (*Geo*). All the DNN-based methods except for LCDNet$_{\dagger}$ are trained on the KITTI dataset as described in Section 3.3.1, and evaluated individually on sequences from both KITTI and KITTI-360 datasets.

Table 3.2 reports the results with the AP metric for protocol 1 and protocol 2. The best method is highlighted in bold, and the second best is underlined. Moreover, Figure 3.4 presents the precision-recall curves for both protocols. As reported, while most approaches achieve satisfactory results in detecting loop closures in the same direction (fig. 3.4 (a)), this is not the case for reverse loops as shown in fig. 3.4 (b). M2DP and SG_PR completely fail on the KITTI sequence 08; Scan Context, OverlapNet (*Geo*) and LiDAR-Iris also show a strong decrease in performance when dealing with reverse loops. For instance, the previous state-of-the-art method Scan Context achieved an AP of 0.96 in sequence 00 of the KITTI dataset (which contains only same direction loops), and 0.65 in sequence 08. The proposed LCDNet, on the other hand, performs equally well for both reverse and same direction loops, achieving an AP of 0.94 and 0.97 respectively. This is even more noticeable in the results using protocol 2 where all the other approaches

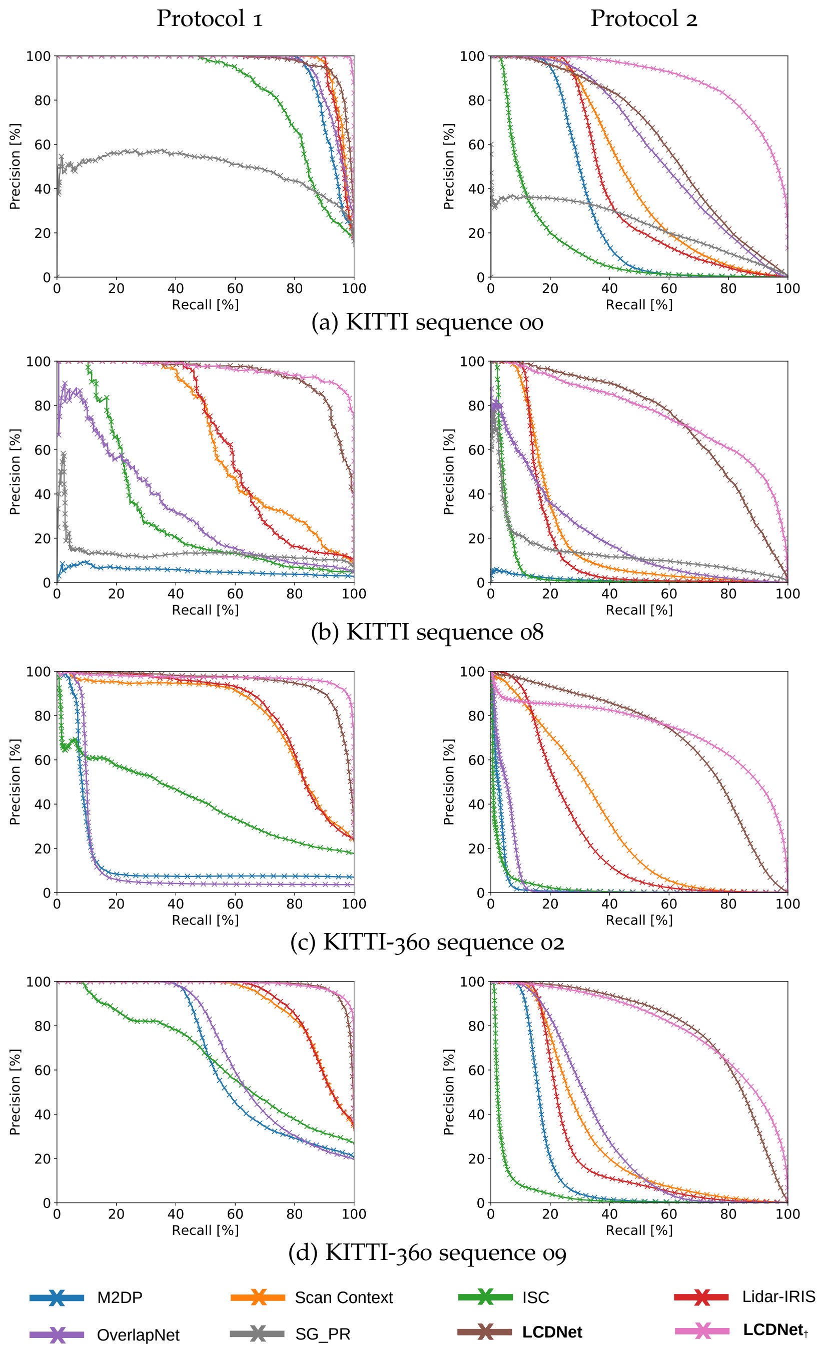


Figure 3.4: Comparison of loop closure detection precision-recall curves on KITTI (a-b) and on KITTI-360 (c-d) datasets evaluated using both protocols. The proposed LCDNet† achieves the best performance in all the experiments, followed by LCDNet as second best method. The improvement over previous state-of-the-art approaches is even more prominent when dealing with reverse direction loops, as observed in (b).

Table 3.3: Comparison of relative pose errors (rotation and translation) between positive pairs on the KITTI dataset (Seq. 00).

	Approach	Seq. 00		
		Success	TE [m] (succ. / all)	RE [deg] (succ. / all)
Handcrafted	Scan Context* [86]	97.66%	- / -	1.34 / 1.92
	ISC* [178]	32.07%	- / -	1.39 / 2.13
	LiDAR-Iris* [181]	98.83%	- / -	0.65 / 1.69
	ICP (P2p) [199]	35.57%	0.97 / 2.08	1.36 / 8.98
	ICP (P2pl) [199]	35.54%	1.00 / 2.11	1.39 / 8.99
	RANSAC [145]	33.95%	0.98 / 2.75	1.37 / 12.01
	FGR [202]	34.54%	0.98 / 5972.31	1.2 / 12.79
	TEASER++ [190]	34.06%	0.98 / 2.72	1.33 / 15.85
DNN-based	OverlapNet* [36]	83.86%	- / -	1.28 / 3.89
	RPMNet [192]	47.31%	1.05 / 2.07	0.60 / 1.88
	DCP [182]	50.71%	0.98 / 1.83	1.14 / 6.61
	PCAM [25]	99.68%	**0.07 / 0.08**	0.35 / 0.74
Ours	LCDNet (fast)	93.03%	0.65 / 0.77	0.86 / 1.07
	LCDNet	**100%**	0.11 / 0.11	**0.12 / 0.12**
	LCDNet$_\dagger$ (fast)	99.79%	0.28 / 0.29	0.30 / 0.30
	LCDNet$_\dagger$	**100%**	0.14 / 0.14	0.14 / 0.14
	LCDNet + ICP	100%	0.04 / 0.04	0.09 / 0.09
	LCDNet$_\dagger$ + ICP	100%	0.04 / 0.04	0.08 / 0.08
	LCDNet + TEASER	94.39%	0.66 / 0.77	0.09 / 0.10
	LCDNet$_\dagger$ + TEASER	99.78%	0.28 / 0.29	0.09 / 0.09

* these approaches only estimate the rotation between two point clouds, therefore are not directly comparable with the other approaches which estimate the full 6-DoF transformation under driving conditions.

show a substantial decrease in performance, while LCDNet achieves an AP score that is even better for detecting reverse loops than the same direction loops. Finally, the model trained on the KITTI-360 dataset (LCDNet$_\dagger$) achieves the best performance on all the sequences, thereby setting the new state-of-the-art on both KITTI and KITTI-360.

3.3.4 *Evaluation of Relative Pose Estimation*

This section reports the evaluation of the relative pose estimation between two point clouds. The proposed LCDNet provides a full 6-DoF transformation under driving conditions between two points clouds. However, Scan Context, ISC, LiDAR-Iris, and OverlapNet only provide an estimation of the yaw angle. As M2DP, and SG_PR do not provide any information about the relative pose, the results presented

Table 3.4: Comparison of relative pose errors (rotation and translation) between positive pairs on the KITTI dataset (seq. 08).

	Approach	Seq. 08		
		Success	TE [m] (succ. / all)	RE [deg] (succ. / all)
Handcrafted	Scan Context* [86]	98.21%	- / -	1.71 / 3.11
	ISC* [178]	81.28%	- / -	2.07 / 6.27
	LiDAR-Iris* [181]	<u>99.29%</u>	- / -	0.93 / 1.84
	ICP (P2p) [199]	0%	- / 2.43	- / 160.46
	ICP (P2pl) [199]	0%	- / 2.44	- / 160.45
	RANSAC [145]	15.61%	1.33 / 4.57	1.79 / 37.31
	FGR [202]	17.16%	1.32 / 35109.13	1.76 / 28.98
	TEASER++ [190]	17.13%	1.34 / 3.83	1.93 / 29.19
DNN-based	OverlapNet* [36]	0.10%	- / -	2.03 / 65.45
	RPMNet [192]	27.80%	1.28 / 2.42	1.77 / 13.13
	DCP [182]	0%	- / 4.01	- / 161.24
	PCAM [25]	94.90%	0.19 / 0.41	0.51 / 6.01
Ours	LCDNet (fast)	60.71%	1.02 / 1.62	1.65 / 3.13
	LCDNet	**100%**	**0.15 / 0.15**	**0.34 / 0.34**
	LCDNet$_\dagger$ (fast)	88.51%	0.66 / 0.93	1.00 / 1.31
	LCDNet$_\dagger$	**100%**	<u>0.18 / 0.18</u>	<u>0.36 / 0.36</u>
	LCDNet + ICP	100%	0.09 / 0.09	0.33 / 0.33
	LCDNet$_\dagger$ + ICP	100%	0.07 / 0.07	0.32 / 0.32
	LCDNet + TEASER	94.39%	1.05 / 1.62	0.33 / 0.35
	LCDNet$_\dagger$ + TEASER	89.39%	0.67 / 0.93	0.33 / 0.34

* these approaches only estimate the rotation between two point clouds, therefore are not directly comparable with the other approaches which estimate the full 6-DoF transformation under driving conditions.

Table 3.5: Comparison of relative pose errors (rotation and translation) between positive pairs on the KITTI-360 dataset (Seq. 02).

	Approach	Seq. 02		
		Success	TE [m] (succ. / all)	RE [deg] (succ. / all)
Handcrafted	Scan Context* [86]	92.31%	- / -	1.60 / 5.49
	ISC* [178]	83.15%	- / -	1.71 / 3.44
	LiDAR-Iris* [181]	96.54%	- / -	1.07 / 2.24
	ICP (P2p) [199]	4.19%	1.10 / 2.26	1.74 / 149.76
	ICP (P2pl) [199]	4.19%	1.11 / 2.30	1.18 / 149.39
	RANSAC [145]	24.78%	1.24 / 3.67	1.83 / 32.22
	FGR [202]	27.92%	1.23 / 6758.87	1.85 / 18.16
	TEASER++ [190]	27.02%	1.25 / 3.16	1.83 / 19.16
DNN-based	OverlapNet* [36]	11.42%	- / -	1.79 / 76.74
	RPMNet [192]	37.99%	1.18 / 2.26	1.30 / 5.97
	DCP [182]	5.62%	1.09 / 3.14	1.36 / 149.27
	PCAM [25]	97.46%	**0.20 / 0.30**	0.75 / 1.36
Ours	LCDNet (fast)	83.92%	0.84 / 1.10	1.28 / 1.67
	LCDNet	**98.62%**	0.28 / 0.32	0.32 / 0.35
	LCDNet$_\dagger$ (fast)	89.07%	0.40 / 0.45	0.57 / 0.62
	LCDNet$_\dagger$	98.55%	0.27 / 0.32	**0.32 / 0.34**
	LCDNet + ICP	98.51%	0.20 / 0.25	0.24 / 0.27
	LCDNet$_\dagger$ + ICP	98.51%	0.20 / 0.25	0.24 / 0.27
	LCDNet + TEASER	86.63%	0.85 / 1.10	0.40 / 0.52
	LCDNet$_\dagger$ + TEASER	98.06%	0.40 / 0.45	0.37 / 0.45

* these approaches only estimate the rotation between two point clouds, therefore are not directly comparable with the other approaches which estimate the full 6-DoF transformation under driving conditions.

Table 3.6: Comparison of relative pose errors (rotation and translation) between positive pairs on the KITTI-360 dataset (seq 09).

	Approach	Seq. 09		
		Success	TE [m] (succ. / all)	RE [deg] (succ. / all)
Handcrafted	Scan Context* [86]	95.25%	- / -	1.40 / 6.80
	ISC* [178]	86.26%	- / -	1.51 / 7.08
	LiDAR-Iris* [181]	97.63%	- / -	0.72 / 3.80
	ICP (P2p) [199]	21.24%	1.06 / 2.22	1.34 / 66.34
	ICP (P2pl) [199]	21.29%	1.07 / 2.24	1.38 / 66.23
	RANSAC [145]	29.69%	1.12 / 3.14	1.48 / 23.42
	FGR [202]	30.46%	1.12 / 6011.39	1.44 / 17.35
	TEASER++ [190]	30.32%	1.14 / 2.91	1.46 / 19.22
DNN-based	OverlapNet* [36]	54.33%	- / -	1.38 / 33.62
	RPMNet [192]	41.42%	1.13 / 2.21	1.02 / 3.95
	DCP [182]	30.10%	1.04 / 2.30	1.06 / 64.86
	PCAM [25]	99.78%	**0.12 / 0.13**	0.51 / 0.64
Ours	LCDNet (fast)	89.49%	0.76 / 0.94	0.99 / 1.19
	LCDNet	**100%**	0.18 / 0.18	**0.20 / 0.20**
	LCDNet$_{\dagger}$ (fast)	98.87%	0.43 / 0.44	0.59 / 0.63
	LCDNet$_{\dagger}$	**100%**	0.20 / 0.20	0.22 / 0.22
	LCDNet + ICP	100%	0.10 / 0.10	0.15 / 0.15
	LCDNet$_{\dagger}$ + ICP	100%	0.11 / 0.11	0.15 / 0.15
	LCDNet + TEASER	90.57%	0.76 / 0.94	0.22 / 0.25
	LCDNet$_{\dagger}$ + TEASER	99.10%	0.43 / 0.44	0.22 / 0.23

* these approaches only estimate the rotation between two point clouds, therefore are not directly comparable with the other approaches which estimate the full 6-DoF transformation under driving conditions.

in this section do not include them. Moreover, the proposed evaluation compares LCDNetwith state-of-the-art handcrafted methods for point cloud registration: ICP [199] using point-to-point and point-to-plane distances, RANSAC with FPFH features [145] and Fast Global Registration (FGR) [202], all implemented in the Open3D library [203], and TEASER++ [190] using the official implementation. A comparison with DNN-based methods RPMNet [192], Deep Closest Point (DCP) [182] and Product of Cross-Attention Matrices (PCAM) [25] is also provided. To provide a fair evaluation, all the latter DNN-based approaches are trained on the same data, following the same protocol, and using the same number of keypoints used to train the proposed LCDNet. Following [39], for the aforementioned handcrafted methods a first downsample is applied to the point clouds using a voxel size of 0.3 meter, while the latter DNN-based methods and LCDNet perform point cloud registration using 4096 sampled points, which is a much sparser representation. Scan-Context, LiDAR-Iris, ISC, and Overlap-Net, on the other hand, operate on spherical projections of the points, and thus they process almost all the points in the original cloud. We evaluate two versions of the method described in this chapter. The first one, denoted as *LCDNet (fast)*, leverages the output of the UOT-based relative position head to estimate the transformation. In the second version, denoted as *LCDNet*, the UOT-based head is replaced with a RANSAC estimator, as described in section 3.2.3. The models trained on KITTI-360 are denoted as *LCDNet*$_\dagger$ *(fast)* and *LCDNet*$_\dagger$, respectively. The performance of LCDNet followed by a further ICP registration are also evaluated. The latter evaluation is only reported as a reference to show the best alignment achievable. Finally, a further investigation is conducted to understand whether TEASER++ is a better pose estimator by replacing RANSAC in LCDNet.

All the methods are evaluated in terms of success rate (percentage of successfully aligned pairs), translation error (TE), and rotation error (RE) averaged over successful pairs as well as over all the positive pairs. Two pairs are successfully aligned if the final rotation and translation error is below five degrees and two meters, respectively. The results on the KITTI and KITTI-360 datasets are reported in tables 3.3 to 3.6. We observe that LiDAR-Iris achieves the best performance among the handcrafted methods and PCAM demonstrates superior results compared to existing DNN-based approaches when dealing with same and reverse direction pairs. However, as opposed to the other methods, PCAM only performs point cloud registration and do not provide any information regarding loop closure detection. Whereas, the proposed LCDNet and LCDNet$_\dagger$ achieve the highest success rates and lowest rotation errors compared to all the methods, with a success rate of 100% in three out of four sequences. PCAM, on the other hand, achieves the lowest translation errors in most sequences, but is not robust to registration under partial overlap, as discussed in section 3.3.5. The fast versions of the proposed method achieve results comparable with, and in some sequences even better than existing approaches, while being much faster than most point cloud registration methods, as reported in section 3.3.8. By replacing RANSAC in LCDNet and LCDNet$_{\dagger}$ with TEASER++ the success rates decrease and the translation errors significantly increase, while the rotation errors remain similar. During the proposed experimental evaluations, we also observed that while the rotation and translation invariance obtained by LCDNet primarily arise from the proposed data augmentation scheme, many existing loop closure detection approaches (not reported in the comparison) did not converge at all when trained with the same scheme. Therefore, we argue that data augmentation by itself is not sufficient, and a well-designed architecture and loss function is necessary to achieve invariance.

3.3.5 *Partial Overlap*

In this section, the proposed evaluation aims to demonstrate the ability of LCDNet in detecting loops and regressing the relative pose between point clouds that only overlap partially. In particular, the evaluation follows the protocol used in section 3.3.3 (protocol 1) and section 3.3.4. To do so,partial overlapping pairs are simulated by removing a random section of each point cloud. The proposed comparison aims to observe the performance of LCDNet against state-of-the-art approaches on the sequence 08 of the KITTI dataset under two settings: by removing a random 45° and 90° sector, respectively. Table 3.7 reports the results of this experiment in terms of average precision (AP), success rate, mean translation error and mean rotation error. Although the AP of LCDNet drops moderately when a 90° section is removed, LCDNet$_{\dagger}$ still achieves an AP higher than all the existing approaches evaluated on the complete overlap test (table 3.2). Not that PCAM, which achieves remarkable results in the full overlap registration test, struggles when dealing with partial overlapping point clouds with a success rate that drops from 95% to 56%, a translation error that increases from 0.41 m to 3.32 m, and a rotation error that raises from 6.01° to 34.64°. LCDNet and LCDNet$_{\dagger}$, on the other hand, retain an almost perfect success rate and slightly lower translation and rotation errors.

Also an investigation on the MulRan dataset [87] was performed for this experiment, as the LiDAR mounted on their vehicle is obstructed by the radar sensor for approximately 70° rear FOV. Therefore, in reverse direction scenarios, the scans share only a very limited overlap.

In preliminary evaluations, all the considered approaches failed in detecting reverse loops. We can argue that this is a limitation of all scan-to-scan methods, and that scan-to-map approaches should be considered in these scenarios.

3.3.6 *Ablation Studies*

This section presents ablation studies on the different architectural components of the proposed LCDNet. All the models presented in this section are trained on the KITTI dataset, and evaluated on the sequence 08 using the AP, mean rotation error (RE) and mean translation error (TE) metrics. Sequence 08 is selected as the validation set since it is the most challenging sequence, containing only reverse direction loops. Since RANSAC does not influence the training of the network, in this section the rotation and translation errors are computed using the *LCDNet (fast)* version.

The first comparison considers the proposed feature extractor built upon PVRCNN presented in Section 3.2.1 with three different backbones: the widely adopted feature extractor PointNet [138], the dynamic graph CNN EdgeConv [183], and the recent state-of-the-art semantic segmentation network RandLA-Net [71]. All backbones were modified in order to output a feature vector of size $D = 640$ for $N = 4096$ points, similar to our backbone. Table 3.8 reports the results of this evaluation. The ability of the adapted feature extractor presented in Section 3.2.1 to combine high-level features from the 3D voxel DNN with fine-grained details provided by the PointNet-based voxel set abstraction layer is demonstrated by the superior performance compared to other backbones, outperforming them in every metric by a large margin. The proposed backbone built upon PV-RCNN achieves an average precision of 0.94 compared to 0.67 achieved by the second best backbone. For relative pose estimation, PV-RCNN achieves a mean rotation error of 3.13° and a mean translation error of 1.62 m compared to 16.85° achieved by EdgeConv and 3.55 m achieved by RandLA-Net.

Table 3.9 presents ablation studies on the architecture of the relative pose head, the dimensionality of the extracted point features, the effect of the auxiliary optimal transport loss presented in Equation (16), and the number of keypoints. First, a comparison is proposed between the proposed UOT-based relative pose head presented in Section 3.2.3 and a MLP that directly regresses the rotation and translation, similar to [150]. In particular, three models are trained using different rotation representations. The first model, *MLP(sin-cos)* uses two parameters to represent the rotation: the sine and cosine of the yaw angle. *MLP(quat)* represents the rotation as unit quaternions, and *MLP(bingham)* uses

Table 3.7: Comparison of loop closure detection (AP) and relative pose errors (rotation and translation) under partial overlap on the sequence 08 of the KITTI dataset.

	Approach	45°			
		AP	Success	TE [m] (all)	RE [deg] (all)
Handcrafted	Scan Context* [86]	0.52	27.33%	-	57.70
	LiDAR-Iris* [181]	0.43	97.84%	-	2.78
	ICP (P2p) [199]	-	0%	2.42	160.46
	ICP (P2pl) [199]	-	0%	2.45	160.46
	RANSAC [145]	-	15.51%	4.88	43.77
	FGR [202]	-	16.55%	44 439.37	30.30
	TEASER++ [190]	-	16.42%	4.03	30.32
DNN-based	OverlapNet* [36]	0.09	1.11%	-	70.69
	PCAM [25]	-	84.67%	1.04	11.80
Ours	LCDNet	0.79	**100%**	0.20	0.38
	LCDNet$_{\dagger}$	**0.83**	**100%**	**0.19**	**0.36**
	Approach	**90°**			
		AP	Success	TE [m] (all)	RE [deg] (all)
Handcrafted	Scan Context* [86]	0.40	17.40%	-	72.05
	LiDAR-Iris* [181]	0.22	96.28%	-	5.13
	ICP (P2p) [199]	-	0%	2.42	160.46
	ICP (P2pl) [199]	-	0%	2.45	160.42
	RANSAC [145]	-	13.78%	5.50	48.74
	FGR [202]	-	14.49%	235 332.54	34.20
	TEASER++ [190]	-	15.98%	4.37	34.99
DNN-based	OverlapNet* [36]	0.01	0.68%	-	85.68
	PCAM [25]	-	55.62%	3.32	34.64
Ours	LCDNet	0.59	99.93%	0.24	0.46
	LCDNet$_{\dagger}$	**0.70**	**100%**	**0.21**	**0.37**

* these approaches only estimate the rotation between two point clouds, therefore are not directly comparable with the other approaches which estimate the full 6-DoF transformation under driving conditions.

Table 3.8: Ablation study on the backbone network architecture.

Backbone	AP	TE [m]	RE [deg]
PointNet [138]	0.67	5.15	34.14
EdgeConv [183]	0.52	5.44	16.85
RandLA-Net [71]	0.55	3.55	20.08
PVRCNN [159]	**0.94**	**1.62**	**3.13**

Table 3.9: Ablation study on the different architectural components of the proposed LCDNet evaluated on sequence 08 of the KITTI dataset.

Relative Pose Head	Feature Size D	Auxiliary Loss	Num Keypoints	AP	TE [m]	RE [deg]
UOT	640	✓	4096	**0.94**	1.62	3.13
MLP (sin-cos)				0.75	2.14	21.05
MLP (quat)				0.78	2.43	35.16
MLP (bingham)				0.75	2.27	22.69
UOT	640	✓	4096	**0.94**	1.62	3.13
	128			0.92	1.85	3.19
	64			0.92	1.99	3.33
	32			0.86	2.23	4.09
UOT	640	✓	4096	**0.94**	1.62	3.13
		✗		0.83	6.00	4.71
UOT	640	✓	8192	**0.94**	**1.28**	**1.99**
			4096	**0.94**	1.62	3.13
			2048	0.85	4.68	3.73
			1024	0.69	5.17	4.75
			512	0.50	4.79	4.85

the Bingham representation proposed in [132]. From the first set of rows in Table 3.9, one can observe that the proposed relative pose head significantly outperforms the MLP-based heads, especially in the rotation estimation. The proposed relative pose head achieves a mean rotation error of 3.13° compared to 21.05° achieved by the best MLP model. Moreover, the UOT-based head favors keypoint features that are rotation and translation invariant, thus enabling the backbone to learn more discriminative features, consequently also improving the loop closure detection performance. The MLP based heads, on the other hand, require rotation specific features in order to predict the transformation, which hinders the performance of the place recognition head, which can be observed from the lower average precision achieved by these models.

Subsequently, to study the influence of the dimensionality of point features on the performance of the proposed approach, four models are trained by varying dimensionality D as 640, 128, 64, and 32. From the results shown in the second set of rows in Table 3.9, we observe that the performances decrease with lowering the dimensionality D.

Another evaluation is performed to test the performance of LCDNet without the auxiliary loss presented in Equation (16). From the results shown in the third set of rows of Table 3.9, it can be seen that when training without the optimal transport loss, the performance in terms

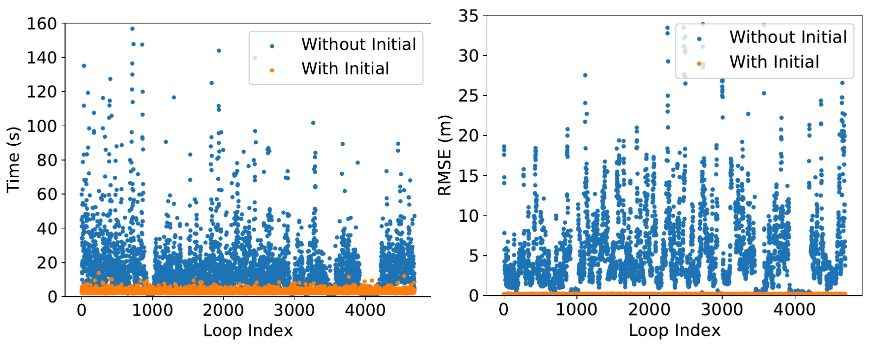


Figure 3.5: Comparison of time (left) and RMSE (right) between ICP without initial guess and ICP with the LCDNet prediction as the initial guess on the sequence 02 of the KITTI-360 dataset. Results on other sequences show similar behaviour, and are thus not reported for brevity. The initial guess provided by LCDNet significantly reduces both runtime and final error on sequences containing reverse loops.

of average precision and relative transformation decreases significantly. This demonstrates that the auxiliary optimal transport loss enables the network to learn more distinctive features, which benefits the performance of both loop closure detection and relative transformation estimation.

Finally, the last set of rows of Table 3.9 reports the performance variation of LCDNet with respect to changes in the number of selected keypoints N. Predictably, the performances increase with adding more keypoints. However, the average precision does not improve when increasing the number of keypoints to 8192. Therefore, due to the higher memory and computation required, the final proposed model uses 4096 keypoints.

3.3.7 *ICP with Initial Guess*

In this experiment, we evaluate the performance of employing LCDNet as an initial guess for further refinement using ICP. A comparison of the runtime and the final Root Mean Squared Error (RMSE) of ICP without any initial guess is provided together with the results obtained by ICP that utilizes LCDNet relative pose estimate as an initial guess. The time of ICP with initial guess also includes the network inference time. Results from this experiment are presented in fig. 3.5 and two qualitative results are shown in fig. 3.6. While only dealing with the same direction loops, ICP achieves satisfactory results and the initial guess does not improve the performance significantly. However, when reverse loops are present, ICP often fails in accurately registering the two point clouds. In this case, the initial guess from LCDNet greatly reduces both the runtime and final errors of ICP as observed in fig. 3.5.

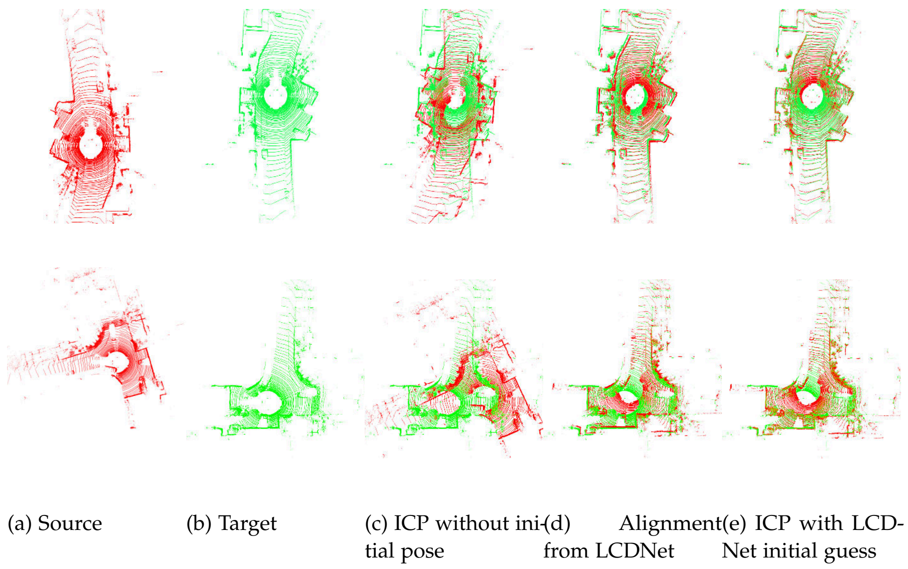

(a) Source
(b) Target
(c) ICP without initial pose
(d) Alignment from LCDNet
(e) ICP with LCDNet initial guess

Figure 3.6: Qualitative comparison of ICP alignment with and without using the LCDNet prediction as an initial guess. ICP alone (c) is not able to register the source (a) and the target (b) when the initial rotation misalignment is high. Whereas, LCDNet effectively aligns them (d). The final ICP alignment with the prediction of the proposed LCDNet as the initial guess further improve the results (e).

From fig. 3.6, we can see that ICP fails when the rotation misalignment between the two point clouds is significant. On the other hand, LCDNet accurately aligns these two point clouds and it improves the results even further while using ICP with LCDNet prediction as initial guess. On average, ICP with LCDNet initial guess is 4 times faster than ICP without any initial guess and achieves an RMSE which is 22 times lower. Note that in the results presented in fig. 3.5, the whole point clouds is used to perform the registration with ICP.

3.3.8 *Runtime Analysis*

In this section, we compare the runtime of LCDNet with existing state-of-the-art approaches for loop detection. All experiments were performed on a system with an Intel i7-6850K CPU and an NVIDIA GTX 1080 ti GPU. We use the official implementation of existing approaches as described in Sections 3.3.3 and 3.3.4. Results from this experiment are presented in Table 3.10 in which the descriptor extraction time also includes the preprocessing required by the respective method. The pairwise comparison represents the time required to compare the descriptors of two point clouds. In the map querying column, we report the time for comparing the descriptor of one scan with that of all the previous scans in the KITTI-360 sequence 02, which amounts to 18 235 comparisons in total. For methods that do not re-

Table 3.10: Comparison of runtime analysis for the loop closure task.

Method	Descriptor Extraction [ms]	Pairwise Comparison [ms]	Map Querying [ms]	GPU Required
M2DP [66]	169.28	0.01	5	✗
SC [86]	3.66	0.11	2000	✗
SC-50 [86]	3.66	0.11	6.96	✗
ISC [178]	1.97	0.53	9000	✗
LiDAR-Iris [181]	8.13	5.39	98 000	✗
OverlapNet [36]	16.00	6.00	109 000	✓
LCDNet	94.60	0.01	5	✓

Table 3.11: Comparison of runtime analysis for the point cloud registration task.

	Method	Descriptor Extract. [ms]	Pairwise Reg. [ms]	Total [ms]	GPU
Handcrafted	SC [86]	3.66	0.11	7.43	✗
	ISC [178]	1.97	0.53	4.47	✗
	LiDAR-Iris [181]	8.13	5.39	21.65	✗
	ICP (P2p) [199]	-	25.53	25.53	✗
	ICP (P2pl) [199]	8.16	35.83	52.15	✗
	RANSAC [145]	24.99	299.66	349.64	✗
	FGR [202]	24.99	188.74	238.72	✗
	TEASER++ [190]	24.99	94.89	144.87	✗
DNN-based	OverlapNet [36]	16.00	6.00	38.00	✓
	RPMNet [192]	366.75	121.29	854.79	✓
	DCP [182]	19.56	78.76	117.88	✓
	PCAM [25]	187.71	80.77	456.18	✓
Ours	**LCDNet (fast)**	94.60	4.70	193.9	✓
	LCDNet	94.60	1135	1324.2	✓

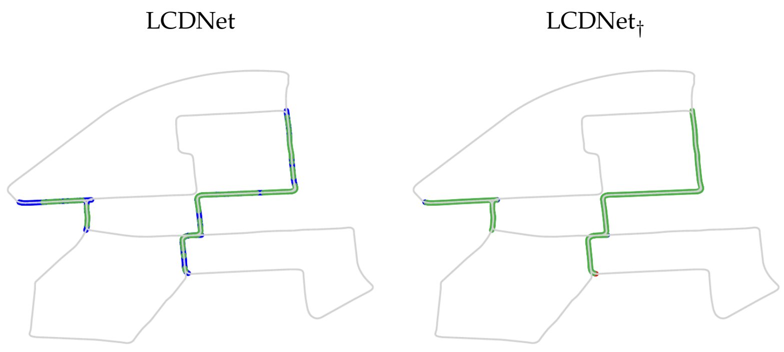
quire an ad-hoc function to compare descriptors (LCDNet and M2DP), we use the efficient FAISS library [76] for similarity search in order to build and query the map. Scan Context also introduces the *ring key* descriptors which enable fast search for finding loop candidates, at the expense of detection performances. We also report the runtime of scan context using the ring key, denoted as *Scan Context-50*. However, it is important to note that the results reported in 3.3.3 were computed without the ring key.

As shown in Table 3.10, the methods that require an ad-hoc comparison function (ISC, LiDAR-Iris, and OverlapNet) are not suited for real-time applications, since they require up to 100 seconds to perform a single query. Whereas, LCDNet queries more than 18 000 scans in five milliseconds. Although it is possible to further reduce the time required to query the map when integrating the loop closure approaches in a SLAM system, such as using the covariance-based radius search [36], in this experiment we evaluate the runtime in the case where no prior information about the current pose is available.

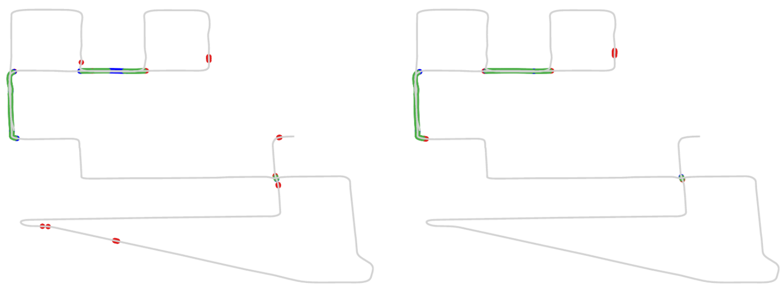
We report the runtime for aligning two point clouds by LCDNet and existing approaches in table 3.11. For methods that perform both loop closure and point cloud alignment (SC, ISC, LiDAR-IRIS, OverlapNet, and LCDNet), the descriptor extraction time is shared between the two tasks. While LCDNet (fast) is faster than most DNN-based approaches for point cloud registration (RPMNet and PCAM), LCDNet is slightly slower than RPMNet. On the other hand, some approaches are much faster than both LCDNet and LCDNet (fast); however, they either only estimate a 1-DoF transformation (SC, ISC, and LiDAR-Iris), or achieve unsatisfactory performances (ICP, RANSAC, FGR, TEASER++, and DCP). It is important to note that the point cloud registration task does not need to run in realtime, since it is only required after a loop closure is detected. Moreover, LCDNet is the only method that performs both loop closure detection and 6-DoF point cloud registration under driving conditions.

3.3.9 *Qualitative Results*

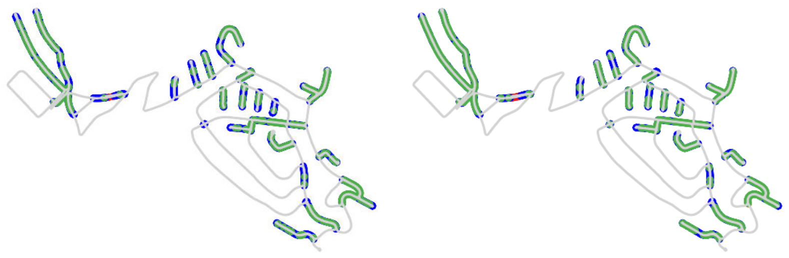
Figure 3.7 reports the qualitative results from LCDNet and LCDNet$_\dagger$ on sequences from both the KITTI and KITTI-360 datasets. The figure shows the true positive, false positive, and false negative scans overlaid with the respective groundtruth trajectories. We observe that while LCDNet effectively detects same direction and reverse direction loops, it also fails to detect some loops (false negative) and detects some loops where there should be no loops (false positive). LCDNet$_\dagger$ further improves the performance by reducing the number of false positives and false negatives, while still maintaining accurate true positive



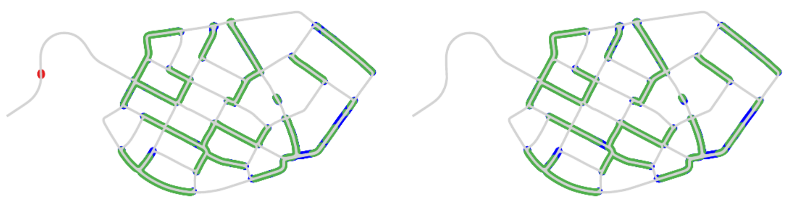
(a) KITTI sequence 00



(b) KITTI sequence 08



(c) KITTI-360 sequence 02



(d) KITTI-360 sequence 09

Figure 3.7: Qualitative loop closure detection results of LCDNet on KITTI (a-b) and KITTI-360 (c-d) datasets. Green points ● are true positive detections, red points ● are false positive, and blue points ● are false negative. The left column shows results of LCDNet trained on the KITTI dataset, while the right columns shows results of LCDNet$_\dagger$ trained on the KITTI-360 dataset. While both LCDNet and LCDNet$_\dagger$ effectively detects loops in all the sequences, LCDNet$_\dagger$ further reduces the number of false positive and false negative detections.

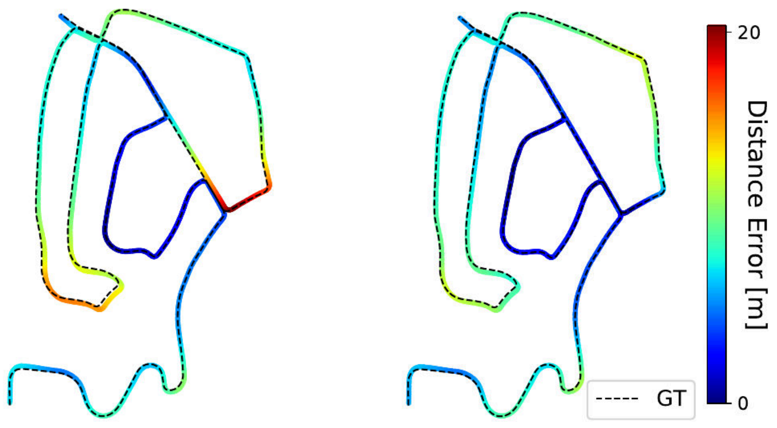


Figure 3.8: Performance of LIO-SAM with the original loop closure detection method (left) compared to our approach (right) on sequence 02 of the KITTI dataset.

detections. On the KITTI sequence 08, LCDNet yields some false negative detections that are almost completely eliminated by LCDNet$_\dagger$, although few false positive scans are still detected. On sequence 02 of the KITTI-360 dataset, LCDNet presents a large amount of false negatives which are significantly reduced by LCDNet$_\dagger$. Similarly, on the KITTI-360 sequence 09, LCDNet presents a few false positive detections that are completely eliminated by LCDNet$_\dagger$.

3.3.10 *Evaluation of Complete SLAM System*

The proposed approach is integrated into LIO-SAM [157], which is a recent state-of-the-art LiDAR SLAM system, by replacing its loop closure detection pipeline with LCDNet. The entire SLAM system is evaluated on the sequence 02 of the KITTI dataset. In particular, the evaluation includes a comparison between the original implementation of LIO-SAM and LIO-SAM integrated with LCDNetL̇IO-SAM with LCDNetdetects loop closures where the original LIO-SAM fails to do so due to the presence of the accumulated drift. Figure 3.8 reports the results obtained with both SLAM systems and shows the distance error between LIO-SAM keyframes and groundtruth poses. We can observe that the high error (red) associated with the path of the original LIO-SAM is caused by the failed loop closure detection, since the system drifts significantly along the z-dimension. Conversely, LCDNet detects such loops, perform the closure and improve the overall performance of LIO-SAM.

A public release of the proposed LCDNet is available at `http://rl.uni-freiburg.de/research/lidar-slam-lc`.

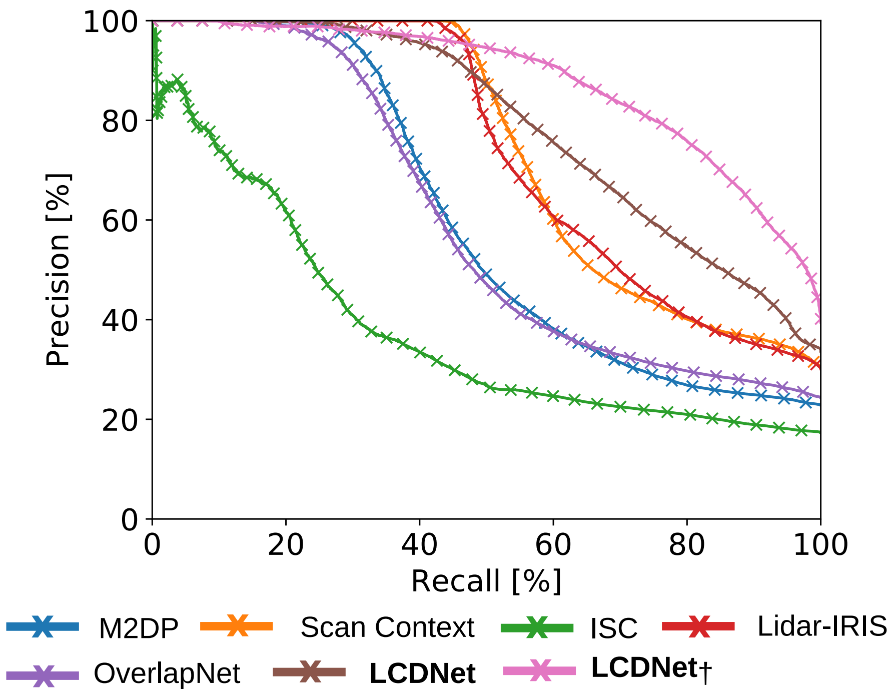


Figure 3.9: Comparison of precision-recall curves evaluated using protocol 1 on data from the generalization experiments in Freiburg.

3.3.11 *Generalization Analysis*

Finally, in this section, the generalization ability of the proposed LCDNet is assessed by analyzing the performance in unseen environments and on different robot platforms. Both LCDNet and LCDNet$_\dagger$ are evaluated in real-world experiments in Freiburg using a car with a rack of LiDAR sensors mounted on the roof as shown in fig. 3.10 (bottom right). Note that in these experiments, LCDNetand LCDNet$_\dagger$ are neither retrained nor fine-tuned on any data from Freiburg. The KITTI and KITTI-360 datasets on which these models were on, were primarily recorded in narrow roads, but the streets of Freiburg also include dual carriageways, therefore we increase the range at which two scans are considered to be a real loop from 4 to 10 meters. Figure 3.9 shows a comparison of the precision-recall curves of the proposed approach with handcrafted and DNN-based methods using protocol 1 (section 3.3.3). Results using protocol 2 are not reported for this experiment as there are more than 400 million positive pairs in this trajectory.

As shown in Table 3.12, Scan Context achieves the best performance among the handcrafted methods, with an AP of 0.74. Nevertheless, the proposed LCDNet and LCDNet$_\dagger$ outperform all the other approaches achieving an AP of 0.79 and 0.88, respectively. Since the data from Freiburg consists of many reverse loops, existing approaches often fail

Table 3.12: Comparison with the state of the art on data from the generalization experiments in Freiburg.

	Method	AP
Handcrafted	M2DP [66]	0.60
	Scan Context [86]	0.74
	ISC [178]	0.38
	LiDAR-Iris [181]	0.73
DNN-based	OverlapNet [36]	0.59
	LCDNet	0.79
	LCDNet$_\dagger$	**0.88**

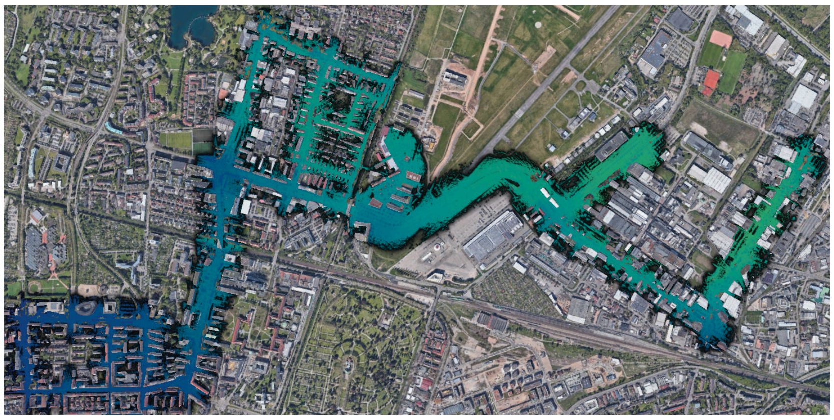
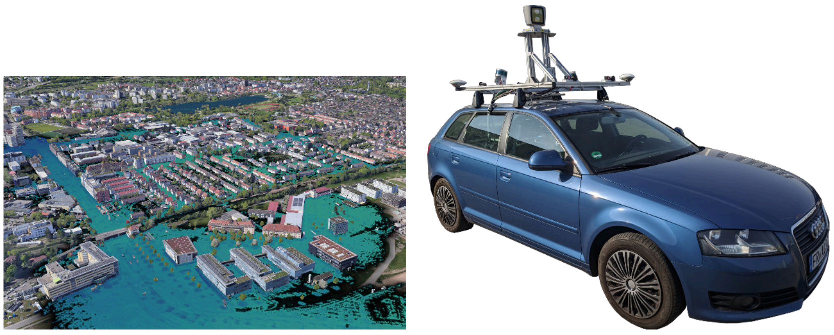



Figure 3.10: Qualitative results of the proposed approach on data from the generalization experiments in Freiburg. The final map generated from LIO-SAM integrated with LCDNet is overlaid on the geo-referenced aerial images. Image on the top shows the entire map, while the images in the bottom show zoomed in segments and the car used to collect the dataset. The color of the point cloud is based on the Z-coordinates of the points from lowest (blue) to highest (green).

to detect them, leading to a decrease in their performance. The approach proposed in this work demonstrates exceptional performance even though it has never seen scans from Freiburg during training. Moreover, we employ the proposed modified version of LIO-SAM to generate the trajectory and the map of the experimental runs in Freiburg. Figure 3.10, shows the resulting map overlaid on the aerial image. The results show that the map is well aligned with the aerial image and there is no evidence of any drift. This demonstrates that LCDNet effectively corrects the accumulated drift. It is important to note that the precision-recall curve and the AP of the proposed LCDNet are computed based only on the global descriptor extracted by the place recognition head. However, in the modified SLAM system, an additional consistency check (section 3.2.5) based on the transformation predicted by the relative pose head allow to further discard the remaining false positive detections.

Finally, the Freiburg dataset is also used to demonstrate the point cloud alignment ability of the proposed approach in new environments. Since an accurate pose is not available for each LiDAR frame, groundtruth transformations are generated using the GPS poses and ICP, and discarding pairs that produce an inaccurate alignment. First, for each frame the identification of possible pairs is made by considering its neighbors within a distance of $10m$. In order to avoid the pairs that are composed of consecutive frames, given a point cloud the previous and the following $n = 100$ frames are discarded. Secondly, for each pair the yaw angle difference Δ_{yaw} is computed and three difficulty levels are defined. Then, for each frame a selection of random pair is performed for every category whenever possible. Moreover, the maximum number of ICP iterations is set to $n_{icp} = 1000$, and only pairs with a fitness score $fit >= 0.6$ and an inlier correspondences $rmse <= 0.3m$ are considered. Finally, we randomly sample the resulting pairs to have about the same number of same direction and reverse direction loops. The resulting number of pairs amounts to 4246, of which 2106 are reverse loops.

The results reported in table 3.13 show that LCDNet effectively generalizes to new environments for the point cloud registration task. LCDNet and LCDNet$_{\dagger}$ achieve a success rate of 98.94% and 99.81%, respectively, compared to the second best method that achieves 92.49%. The overall mean translation error of LCDNet$_{\dagger}$ is more than two times smaller, and the rotation error is an order of magnitude lower than PCAM.

Table 3.13: Comparison of relative pose errors (rotation and translation) between positive pairs on the Freiburg dataset.

	Approach	Success	TE [m] (succ. / all)	RE [deg] (succ. / all)
Handcrafted	Scan Context [86]	59.30%	- / -	1.36 / 52.70
	ISC [178]	55.51%	- / -	1.52 / 51.02
	LiDAR-Iris [181]	69.95%	- / -	1.52 / 51.02
	ICP (P2p) [199]	29.06%	0.83 / 2.60	1.21 / 89.79
	ICP (P2pl) [199]	28.73%	0.88 / 2.62	1.22 / 89.83
	RANSAC [145]	29.96%	1.01 / 3.54	1.34 / 31.29
	FGR [202]	27.72%	0.97 / 313 258	1.27 / 13.46
	TEASER++ [190]	29.49%	0.99 / 3.37	1.31 / 11.94
DNN-based	OverlapNet [36]	42.79%	- / -	1.31 / 70.91
	RPMNet [192]	32.05%	0.87 / 2.57	1.09 / 46.99
	DCP [182]	12.25%	1.26 / 5.22	1.19 / 87.04
	PCAM [25]	92.49%	0.40 / 0.67	0.50 / 4.28
Ours	LCDNet	98.94%	0.39 / 0.42	0.32 / 0.37
	LCDNet$_\dagger$	**99.81%**	**0.28 / 0.28**	**0.18 / 0.18**
	LCDNet + ICP	99.79%	0.22 / 0.23	0.16 / 0.16
	LCDNet$_\dagger$ + ICP	99.86%	0.21 / 0.22	0.14 / 0.14
	LCDNet + TEASER	33.61%	1.09 / 3.74	0.17 / 0.42
	LCDNet$_\dagger$ + TEASER	33.37%	1.15 / 4.45	0.13 / 0.27

3.4 FINAL DISCUSSION

In this chapter, the novel LCDNet architecture for loop closure detection and point cloud registration was presented. LCDNet is composed of a shared feature extractor built upon the PV-RCNN network, a place recognition head that captures discriminative global descriptors, and a novel differentiable relative pose head based on the unbalanced optimal transport theory which effectively aligns two point clouds without any prior information regarding their initial misalignment.

An extensive experimental activity was performed on the KITTI odometry and KITTI-360 datasets, demonstrating that LCDNetsets the new state-of-the-art and successfully detects loops even in challenging conditions such as reverse direction loops, where existing methods fail. The proposed LCDNet$_{\dagger}$ achieves an average precision of 0.96 on the sequence 08 of the KITTI dataset which contains only reverse direction loops, compared to 0.65 AP of the previous state-of-the-art method. The proposed relative pose head demonstrates impressive results, outperforming existing approaches for point clouds registration and loop closure detection as well as different heads based on the standard MLPs. LCDNet aligns opposite direction point clouds with an average rotation error of 0.34°, and 0.15 m for the translation components, compared to 1.84° and 0.41 m achieved by LiDAR-IRIS and PCAM, respectively. Moreover, LCDNet is robust to partial overlapping point cloud, retaining a 100% success rate when removing a 90° sector from each point cloud, while the second best method drop from 95% to 55%. In this chapter, we could observe that the relative pose prediction from the proposed approach can further be refined using ICP for accurate registration. Moreover, the integration of LCDNet with LIO-SAM was proposed to provide a complete SLAM system which can detect loops even in presence of strong drift. Additionally, LCDNet demonstrated its generalization ability by deploying such a model on a different robotic platform and in an unseen city from that which was used for training.

4 SAMPLING UNCERTAINTY IN LIDAR-BASED PLACE RECOGNITION

In Chapter 3, a Simultaneouos Localization And Mapping (SLAM) approach exploiting Light Detection And Ranging (LiDAR) data was introduced, tackling the localization problem both from a global and local perspective. In particular, the proposed 3D Deep Neural Network (DNN) named LCDNet jointly identifies loop closures and performs the registration of a LiDAR input with a candidate point cloud, representing a previously visited location. Despite the proposed method achieves accurate results and demonstrates its robustness against adverse conditions, such as reverse loops, it remains challenging to understand whether this model is likely to fail.

As reported in Section 2.3.1, false loops detection is a serious issue, since a single wrong match can lead to an unrecoverable failure of the localization system. Therefore, on one hand LCDNet shows impressive recall and precision in LiDAR-based place recognition, on the other hand this model experience the lack of an explicit tool for detecting failures, that could be crucial for the critical task of loop closure detection. The same flaw is generally experienced in other existing approaches [62, 96].

A naive and commonly used solution to the underlying problem is to fix a threshold on a similarity score computed over the feature representations of the loop candidates, assuming that the considered DNN enables the computation of global descriptors. However, this threshold becomes an hyperparameter that one should tune according to the deployment scenario. For example, if distinct locations of the navigation environment are perceptually similar, the descriptors produced by the model could lie in a very close region of the feature space, implying an higher similarity score together with a larger probability to be wrongfully matched.

Perceptual aliasing is not the unique reason behind hypothetical failures, since other uncontrolled environmental factors could affect the quality of an input observation. For instance, dynamic entities, *e.g.*, pedestrians and cars, often occlude other scene elements that an algorithm could utilize to provide a more accurate localization estimate. Moreover, due to the lack of transparency of deep learning models, the presence of these objects could lead to undesired and unpredictable effects within the feature representation extracted from a model input.

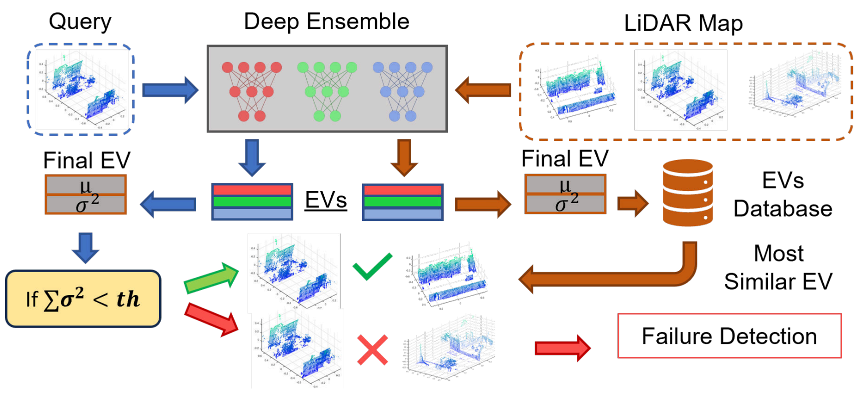


Figure 4.1: The main goal of this work is to provide an ensemble-based method to address the LiDAR-based place recognition task. In particular, different models are trained by means of a knowledge distillation learning strategy to produce similar feature spaces among ensemble members and to enable uncertainty estimation. In a later step, this uncertainty can be used to detect localization failures by improving the accuracy of the overall system. Finally, we compare our approach with other ensemble-based strategies by exploring the advantages and flaws of each method.

In recent years, several deep learning methods for LiDAR-based place recognition [4, 8, 62, 96] have demonstrated outstanding performance, making them valuable tools also for SLAM systems [72]. As mentioned earlier, using models' estimates without considering their potential uncertainty can lead to treacherous situations and have serious consequences when making decisions during navigation, endangering the safety of both passengers and other road users. Therefore, the ability to accurately estimate the uncertainty associated to the output of DNN models is of paramount importance for mitigating the risks inherent in the deployment of these technologies. This will allows autonomous systems to make more informed decisions during navigation, reducing potential hazards. For instance, human drivers constantly take decisions without a complete knowledge of the surrounding environment and, nevertheless, they are still able to manage very complex situations without negative consequences most of the time. This phenomenon happens since the ability of measuring "what we do not know" can improve the driving experience, making human drivers more cautious when they do not fully understand what is happening within the current scene. From a technical perspective, uncertainty refers to situations where information is incomplete or unavailable, often arising due to the inherent complexity of an environment or the limitations of a model in accurately representing a specific aspect of reality.

In order to overcome the previous problems and to deal with ambiguous scenes, this chapter presents an approach for estimating *uncertainty* in a DNN for LiDAR-based place recognition. In particular, the proposed method is based on the popular technique of Deep Ensembles (DEs) [105], with the aim of sampling model uncertainty, *i.e.*, *epistemic* uncertainty, directly on feature elements composing the final point cloud descriptor. In terms of performance, DEs usually provide high uncertainty quality, making them a solid choice among the existing uncertainty estimation techniques [50]. However, as described in this chapter, the application of these technique in the context of place recognition is not straightforward. Furthermore, in the place recognition field, there have been relatively few methods to estimate uncertainty, with the majority of these approaches focusing primarily on aleatoric uncertainty [24].

To summarize, a novel approach for accurately estimating epistemic uncertainty during the retrieval process is introduced. In particular, the proposed strategy incorporates multiple models that share a common feature space obtained with the application of knowledge distillation method, that allows for a direct comparison of the distinct models output. Finally, an extensive experimental activity is proposed to observe the recall capability of the uncertainty-aware method and to assess the ability of the proposed uncertainty-aware model in eliminating unreliable predictions. Figure 4.1 depicts the place recognition approach described in this chapter, where the proposed global localization pipeline can be mainly divided in two successive steps. Firstly, multiple models generates distinct feature-based representations of input point clouds, from which we derive feature-wise uncertainty. Secondly, localization is performed only if the total uncertainty falls below a specific threshold.

Note that, the proposed technique is not directly related to LCDNet, since the main goal is to provide a more general insight of sampling-based uncertainty estimation methods deployed in the place recognition field. However, one of the most intuitive application still remain the detection of loop closures.

4.1 RELATED WORK

Before introducing the proposed approach, this section will briefly discuss the state-of-the-art topics that have been addressed. In particular, this section will begin with an overview of DNNs for LIDAR-based place recognition. Following this, a brief summary of DNN-based uncertainty estimation approaches will be provided together with methods applied in the context of place recognition.

4.1.1 *LiDAR-based place recognition with DNNs*

In the last decade, many deep learning-based approaches emerged to deal with the place recognition problem by exploiting observations gathered from different sensors. In the literature, we can identify techniques that use only image data [8], 3D data such as LIDAR [4], and multi-modal based methods [30].

Deep learning-based LiDAR place recognition demonstrated superior performance than other modalities by showing robustness across challenging scenarios [4, 96, 97, 197]. Due to such promising results, in this work we decided to focus on LIDAR-based approaches.

One of the groundbreaking works in this field is PointNetVLAD [8], a DNN that combines PointNet [138] for extracting features and VLAD [77] for computing the final descriptor from a point cloud. In the meantime, several other methods appeared in the literature [150], aiming also to address the problem in crowded areas [33] or todefine yaw invariant architectures [120]. More recently, Komorowski *et al.* [96] proposed MinkLoc3D, a DNN that exploits sparse convolutions to extract discriminative descriptors.

In this work, we decided to consider PointNetVLAD and MinkLoc3D as the reference architectures in our in-depth study. On the one hand, PointNetVLAD is a very popular model for place recognition, making it suitable for representing a baseline approach. On the other hand, MinkLoc3D is a novel approach that demonstrates to outperform several state-of-the-art approaches, making it another good baseline candidate. Moreover, we argue that measuring the effect of deep ensembles respecting a traditional and a modern architecture represents an interesting case.

4.1.2 *Uncertainty estimation methods for DNNs*

Nowadays, epistemic uncertainty estimation in neural network approaches represents a particularly discussed topic. One of the first attempts at modeling uncertainty is through Bayesian Neural Networks (BNNs) [43, 129], where a prior is associated with each model's weight. However, their application is limited to light-weighted architectures due to the intractability of the posterior distribution.

To overcome the previous issue, Kingma et al. [90] and Lakshminarayanan et al. [105] proposed two sample-based strategies for estimating epistemic uncertainty: the former exploits dropout sampling, the latter uses a set of predictions provided by a deep ensemble, *i.e.,* a set of instances of a specific model architecture. In general, deep ensembles achieve higher prediction accuracy together with a better representation of uncertainty among the sampling-based approaches.

Nevertheless, sampling techniques have the disadvantage of being expensive from a computational perspective.

More recently, Amini et al. [3] proposed an approach for a direct estimation of uncertainty in DNN models named Deep Evidential Regression (DER). In particular, this approach can estimate both aleatoric and epistemic uncertainty with a single step. However, the approach relies on a large number of hyperparameters, which tuning is challenging by employing a "trial and error" approach during training.

4.1.3 *Uncertainty estimation in place recognition*

In recent years, localization approaches that deal with the uncertainty estimation problem emerged [42, 81, 173] and more recently also in the place recognition field. For instance, Cai *et al.* [24] proposed STUN: a DNN-based approach that exploits a knowledge-distillation technique for estimating aleatoric uncertainty in a visual place recognition task. Another notable work is proposed by Latoje et al. [104] by integrating an aleatoric uncertainty-aware visual place recognition model within a SLAM system. Please note, the previous approaches estimate aleatoric uncertainty, while we aim to model the epistemic component.

Regarding LiDAR based place recognition, Mason et al. [122] recently proposed an ensemble-based strategy for estimating epistemic uncertainty. In particular, they train multiple DNNs with different parameters initialization to extract discriminative point-cloud global descriptors. Then, they represent a navigation map as a set of n database copies, where n corresponds to the number of models used. For each model, they compute the similarity score between an input query and the corresponding database entries. Considering all models, the final match is the location with the highest average similarity score over the different database copies.

To the best of our knowledge, only their work exploits an ensemble-based method for place recognition. However, such an approach presents some flaws that we aim to overcome. For instance, such a method requires different database replicas, and epistemic uncertainty corresponds to similarity variance rather than features variance, making more arduous to obtain a model introspection in presence of challenging inputs. Moreover, they claim that the negative average similarity of the candidate match represents another "natural" way to quantify the model uncertainty. However, we argue that this assumption is flawed for reasons reported in Section 4.2.3. As we will describe later, our method exploits a knowledge-distillation learning technique to allow us a direct comparison between feature spaces created by each model.

4.2 PROPOSED APPROACH

In this section, the proposed uncertainty-aware place recognition pipeline will be presented. The first part of the discussion will be focused on the possible implementations of DEs in place recognition, bringing out the issues that one can encounter when creating a common feature space between ensemble members. Moreover, it will be explained how to represent feature-wise uncertainty by aggregating multiple predictions and how similar methods in the literature works, by highlighting the differences respecting this work.

4.2.1 *Ensemble-based LiDAR place recognition*

In the context of place recognition, we achieve the main goal by generating a compact and discriminative representation of an input observation, such as an image or point cloud. This representation, commonly referred to as Embedding Vector (EV), is then used to find previously visited locations that have been encoded using the same strategy. From a technical perspective, an EV is a vector of m real numbers denoted as $v = (f_1, ..., f_m)$ designed to represent an input observation, enabling the computation of a distance metric respecting other vectors. The final goal is to encode distinct map places with distinct vectors $V = v_1, ..., v_n$, such that observations belonging to the same location are positioned in close proximity within the feature space, while observations from different locations are more distant to each other. By generating accurate and robust EVs, it is possible to effectively match and retrieve previously visited locations, *i.e.*, v_{input} and $v_{retrieved}$ are the most similar according to a predefined measure, thereby facilitating effective navigation and localization.

DNN models designed for place recognition are typically trained with a metric learning approach [26]. This involves the use of a loss function that aims to create a discriminative feature space by comparing positive and negative pairs of samples. Positive examples consist of observations corresponding to the same place, while negative ones consist of samples belonging to different locations. This comparison aims to reduce or increase the distances between EVs, that is the training loss penalize the model whether those expectations are not met. In an ensemble-based approach, where multiple networks are trained independently, the output of the system is generated by aggregating the predictions of these individual networks. In particular, a typical strategy is to compute metrics such as the mean and variance over the ensemble predictions, where the mean corresponds to the final inference and the variance corresponds to the model uncertainty. However, since the output of each model is an EV v comprising several elements,

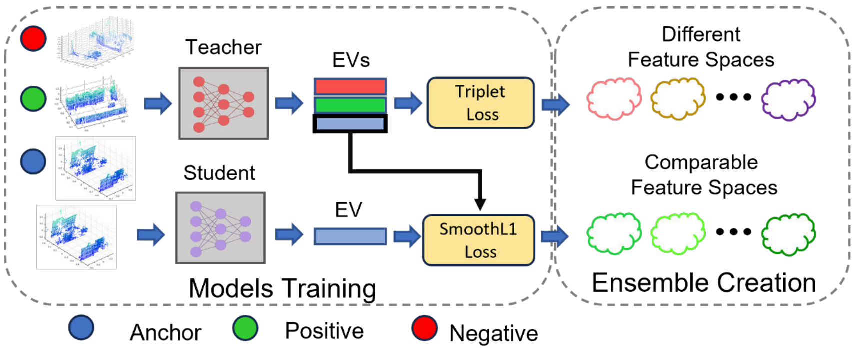


Figure 4.2: Pipeline followed by our method to create ensemble members. Firstly, we train a teacher DNN with a standard metric learning approach (top-left), then we train a student model to imitate the teacher feature space (bottom-left). By doing that, we are able to train different ensemble members to which correspond similar feature spaces (bottom-right). As we will show later, when training different models with a standard triplet loss, we obtain unique feature spaces (top-right) where a direct aggregation of results leads to undesired effects.

in place recognition we could perform different aggregation steps over the i-th features of the output samples V. For example, given a set of EVs $V = \{v_1, ..., v_n\}$, the output of an ensemble could be the mean and variance of each feature computed as follows:

$$E[\mu_v] = \frac{1}{n} \cdot \sum_{i=1}^{n} v_i, \quad Var[\mu_v] = \frac{1}{n} \cdot \sum_{i=1}^{n} (v_i - E[\mu_v])^2 \tag{19}$$

where i corresponds to the i-th member of the ensemble.

While the application of an ensemble-based approach may appear straightforward in theory, the process of averaging features can sometimes lead to undesired results in practice in the context of EVs aggregation. This is because a direct comparison of the vectors within the set V may not always be meaningful, and can result in inaccurate or unreliable estimates. One potential issue arises from the randomization processes that occur during a standard training procedure. Since metric learning is generally an unsupervised process, it is difficult to ensure that each member of the ensemble produces the same feature space and, although a ground-truth is used to build positive and negative sample pairs, it is not possible to explicitly control how each model learns that space. As a result, even if two independent models are able to extract effective EVs from the same input observation, the actual values of the vectors may differ. As another example, two models could learn the same feature space, but the vector generated by the former may be an unknown permutation of the latter.

To address the issue of variability in the feature spaces generated by different members of an ensemble, we propose the use of a popular method known as knowledge distillation. This approach allows us to create an ensemble in which each member is capable of representing similar feature spaces for LiDAR-based place recognition. This is achieved through a two-step process. First, a DNN model, referred to as the *teacher*, is trained using the triplet loss function in Equation (20), with the aim of producing a target feature space:

$$\mathcal{L}^{\mathtt{triplet}} = \sum_{i=1}^{n} [d(\nu_p, \nu_a) - d(\nu_n, \nu_a) + m]_+ \qquad (20)$$

In the equation, ν is a generic EV, a, p and n refer to *anchor*, *positive* and *negative* samples respectively, d is a distance metric and m represents a margin. More specifically, anchor a represents a generic input point cloud, positive sample p is another observation referring to a the same place depicted in a, and negative sample e is a point cloud recorded in a distinct location respecting a and p.

In knowledge distillation, the target feature space serves as a reference for the other members of the ensemble, known as the *students*. Put differently, we train additional models with the goal of imitating the EVs generated by the teacher. To achieve this, we use the *teacher* network to produce ground-truth EVs of the training set. These ground-truth EVs are then used to train the different student models that will become part of the ensemble. This straining strategy takes also the name of *teacher-student* approach, that employs the loss function in Equation (21) to train each member of the ensemble.

$$\mathcal{L}^{\mathtt{ts}} = d(f(p_i), g(p_i)) \qquad (21)$$

This loss function is designed to optimize the performance of the ensemble members, *i.e.*, they achieve accurate place recognition, and ensure that they are able to accurately imitate the EVs generated by the teacher model. In the equation, f and g are the teacher and student model respectively, d is a distance measure computed over the output EVs, *e.g.*, smoothL1 in our case, and p_i is an input point cloud. Please note, only the student model $g(\cdot)$ is trained in this case, while the $f(\cdot)$ weights do not change since we aim to keep the target feature space fixed. In Fig. 4.2, we show the difference between creating an ensemble where members are trained with a triplet loss and a knowledge-distillation approach.

4.2.2 *Exploiting ensemble predictions*

Once our ensemble is trained, we are able to extract features mean and variance from a generic set of predictions V by using Equation (19).

As desired from an ensemble-based approach, we expect to obtain an increment of the overall performance, that in the case of a retrieval system should result in a better recall, that is we are are able to achieve an optimal rate of correct matches. In particular, by training several models, we are able to cover a wider range of patterns within data, that is we are able to provide better point cloud EVs. If one model provides a poor quality embedding, the other ensemble members can mitigate the issue. To match an input query with a map location, we simply compute the similarity between query and database EVs and choose the match with the highest similarity score:

$$s(\nu_q, \nu_{db}) = \frac{\nu_q \cdot \nu_{db}}{||\nu_q||\,||\nu_{db}||} \tag{22}$$

Moreover, we can take advantage of ensemble uncertainty to detect possible failures and decide to not exploit a particular prediction for localization. In this work, we decided to represent the model uncertainty as the total amount of uncertainty associated to the final descriptor $\nu_e = (\nu_\mu, \nu_{\sigma^2})$ generated by the ensemble:

$$u_{EV} = \sum_{i=1}^{m} f_i^{\sigma^2} \tag{23}$$

where $f_i^{\sigma^2}$ represents the variance associated to feature i obtained from ν_{σ^2}, that is $\nu_{\sigma^2} = (f_1^{\sigma^2}, ..., f_n^{\sigma^2})$. Once obtained u_{EV}, we can decide to discard or not the associated predictions according to a predefined threshold.

4.2.3 *Comparison with existing approaches*

To the best of our knowledge, only Mason et al. [122] recently proposed a deep ensemble method for addressing the task of LiDAR-based place recognition. However, there are some differences between their work and the approach proposed in this thesis. In particular, we propose a method to produce feature-wise uncertainty estimates, while they represent uncertainty as a degree of accordance between ensemble members. From a technical perspective, the members of their ensemble-based approach describe distinct feature spaces, that are not directly comparable. However, they calculate separated similarity scores across m replicas of a query ν_q and the map database entries $V_D = (\nu_{db,1}, ..., \nu_{db,n})$, where m corresponds to the number of ensemble members. Then, after obtaining different similarity scores $S = (s_1, ..., s_m)$ for each database entry, the match candidate is the location with the highest average similarity score s_μ. In their case, a possible representation of uncertainty is the similarity variance s_{sigma^2} of EV copies. The authors also affirm that a natural representation of uncertainty can be the negative similarity score obtained from the ensemble, that is $u_e = -s_\mu$. They also state that the usage of this representation leads to better results. However, we argue that with this last design choice one cannot capture an actual representation of model uncertainty. For instance, given an average similarity of $s_\mu = 0.5$, there is a huge difference between the case where all the ensemble members provided a similarity score of $s = 0.5$ and the case where half of the members provided $s = 0.$ and the other half $s = 1.$. The main flaw is that we loose information regarding the actual dispersion of samples and we embed the concept of uncertainty to the measure that the model optimize during training, *i.e.,* s_μ. Due to previous consideration, we compare our approach respecting the similarity variance formulation.

As we will describe in Section 4.3, the previous method has the advantage to considerably improve the overall recall capability at the expense of maintaining distinct database instances. Instead, our approach has the ability to encode ensemble predictions with a single representation and to better detect localization failures in order to improve the recall capability of the proposed system.

4.2.4 *Training Details*

A DNN architecture for place recognition comprises two main components: a feature extractor followed by a feature aggregation layer. Our experimental setup is based on the implementation of two widely used DNNs for 3D place recognition: MinkLoc3D [96] and PointNetVLAD [4]. MinkLoc3D requires a sparse voxelized representation of the scene as input and employs a Feature Pyramid Network architecture [113] to extract relevant features. On the other hand, PointNetVLAD [4] extracts features directly from a point cloud using PointNet as its feature extractor. For feature aggregation, MinkLoc3D uses a Generalized Mean Pooling (GeM) layer, while PointNetVLAD employs a VLAD layer [9]. No modifications were made to the architecture of either neural network considered in this study, since our main objective is to observe the effect generated by the employment of deep ensembles within standard 3D DNNs.

From a technical point of view, and inspired by the training configuration of MinkLoc3D, we have fixed the configuration settings for both the teacher-student and triplet-loss training methods, as well as the two DNNs. Specifically, the batch size expansion described by Komorowski et al. [96] was disabled and the batch size was fixed to 64. Moreover, due to limited memory space available on the GPU, the batch size for training MinkLoc3D and PointNetVLAD using the Knowledge-Distillation pipeline was reduced to 32 and 8 respectively. Initially, we trained models for 40 epochs, but we observed that halving the epochs improved the uncertainty quality for the teacher-student approach without particularly worsening the accuracy of teacher models. A possible explanation is that an extended training leads to a reduction of ensemble members diversity, which is an important property to ensure if we want to obtain high quality uncertainty.

With regard to the training methods, for the triplet-loss function [68] we trivially used Equation (20). However, the teacher-student method involves comparing the feature vector extracted by the student network with the corresponding ground-truth vector extracted by the teacher network. To evaluate the distance between these vectors, we used *smoothL1* distance as the cost function.

4.3 EXPERIMENTAL ACTIVITY

In the following section, we report our extensive experimental activity to compare the ensemble-based methods previously discussed. In particular, we compare our teacher-student ensemble with [122] and with a *naive ensemble* approach that follows a similar idea to our method, but without using a knowledge-distillation technique. In particular, we evaluate the recall gain obtained by each ensemble modality and the capability of detecting localization failures according to a threshold-based strategy that exploits system uncertainty generated by the ensemble. Such an assessment is performed across different datasets never used during the training stage of ensemble members.

4.3.1 *Datasets*

To train DNN models and to validate our approach, we choose three popular automotive datasets: Oxford Robotcar [121], NUS InHouse [4] and MulRan [88]. Please note, for all the methods considered we used only the Oxford Robotcar dataset for training, while others were only used to validate our system.

Oxford and NUS InHouse datasets are characterized by urban scenes collected during multiple traversals of the city of Oxford and Singapore, respectively. Scenes depicted in these datasets can present tricky examples for a place recognition system due to the presence of external actors, *e.g.,* cars, bicycles, and pedestrians, that may occlude relevant scene constituents. Furthermore, the presence of different light and weather conditions causes an increasing recognition difficulty. Similarly, MulRan also comprises tricky environments characterized by repetitive scenarios, inducing perceptual aliasing.

Regarding Oxford Robotcar and NUS InHouse datasets, we exploit the benchmark provided by Uy et al. [4]. For the MulRan dataset, we considered DCC and Riverside locations by selecting two runs for each: one for building a database set and the other for a query set. In particular, we exploited the pre-processed point clouds provided by Knights *et al.* [92].

When training models with a triplet-loss, we consider positive correspondences as point clouds with a maximum distance of 10 meters, whereas negative matches are point clouds situated more than 50 meters apart. Differently, during evaluation, a match is positive if the retrieved database sample is at a maximum distance of 25 meters from the query.

Point clouds belonging to each dataset are pre-processed by applying a set of transformations following the pipeline described by Uy et al.[4]. In particular, such transforms comprehend ground plane removal and 3D downsampling to obtain point clouds with 4096 points. Finally, points coordinates are re-scaled within a range of $[-1; 1]$.

4.3.2 *Evaluation Strategy and Metrics*

For the evaluation, we follow a similar method reported in [4, 96, 122]. We consider a dataset $D = \{d_1, d_2, ..., d_n\}$, where each element d_i corresponds to the $i-th$ traversal, *i.e.*, a specific run performed by the vehicle during recordings. We represent a query set as $Q = \{q_1, q_2, ..., q_n\}$, where q_j refers to the $j-th$ traversal. We compare each d_i with all the queries in Q by excluding q_j when $i = j$. The goal is to evaluate only queries recorded during a different traversal than d. We label a match as *correct* if the spatial distance between the query point cloud and the most similar database point cloud is within $25m$. To assess the retrieval capability of the approaches considered, we employ the recall R@1, R@5, and R@10. We use these recall measures also to highlight the feature spaces differences in the exchangeability test shown in the following subsection.

Finally, to understand if a method can detect localization failures, we observe the recall trend when progressively removing queries according to a dynamic threshold λ applied on u_{EV}, that is we discard an EV if $u_{EV} > \lambda$. Ideally, higher uncertainties should correspond to a higher likelihood of prediction errors. For each step, the recall is computed only considering the number of queries labeled as reliable.

4.3.3 *Feature Space Interchangeability: Triplet vs TS ensemble*

In the following experiment, we compare the feature spaces produced by each trained model. In particular, we aim to demonstrate that only using student models allows us to exchange feature spaces without altering recognition performance, that is features from dif-

		QUERY				
		m1	m2	m3	m4	m5
DATABASE	m1	73,29	1,42	1,11	1,11	1,37
	m2	0,85	72,13	0,68	0,84	1,04
	m3	0,98	0,83	73,19	0,94	1,11
	m4	1,20	0,88	1,41	73,40	1,24
	m5	1,46	0,84	1,17	0,78	72,69

		QUERY				
		m1	m2	m3	m4	m5
DATABASE	m1	71,21	68,25	68,74	69,34	69,32
	m2	67,24	71,63	66,88	67,34	68,50
	m3	68,14	67,19	71,38	68,28	68,82
	m4	69,60	68,28	68,96	71,17	69,48
	m5	69,96	70,06	69,81	69,99	71,34

Table 4.1: Triplet vs knowledge-distillation feature space interchangeability: from top to bottom, we represent the recall@1 of *PointNetVLAD* trained using triplet loss (top) and knowledge-distillation (bottom) loss functions. Only the knowledge-distillation method allows us to exchange database and query sets across different student models.

		QUERY				
		m1	m2	m3	m4	m5
DATABASE	m1	92.37	0.75	0.25	0.28	1.40
	m2	0.87	92.20	0.49	0.48	0.21
	m3	0.58	1.31	91.86	0.74	0.49
	m4	1.34	0.76	1.25	92.33	0.93
	m5	0.98	0.55	0.77	0.14	92.07

		QUERY				
		m1	m2	m3	m4	m5
DATABASE	m1	93.44	92.49	92.94	92.83	92.99
	m2	92.93	93.36	93.09	93.07	93.14
	m3	92.85	92.66	93.60	93.14	93.07
	m4	93.05	92.89	93.30	93.48	93.23
	m5	93.24	92.82	93.15	93.11	93.63

Table 4.2: Triplet vs knowledge-distillation feature space interchangeability: from top to bottom, we represent the recall@1 of *MinkLoc3D* trained using triplet loss (top) and knowledge-distillation (bottom) loss functions. Only the knowledge-distillation method allows us to exchange database and query sets across different student models.

ferent models are directly comparable. Considering a set of models $M = \{m_1, ..., m_n\}$, we employ the following protocol: we use one model m_k to create a set of EVs representing a database D_k, the remaining members to produce different query sets from Q_j and, finally, we observe the R@1 following the protocol described in Section 4.3.2. In Table 4.1 and Table 4.2, we demonstrate the comparability between the database built with a model and the query set obtained from another ensemble member when using a knowledge-distillation approach. In particular, in the former table we conducted the experiments with PointNetVLAD and with MinkLoc3D in the latter case. As can be seen, we can achieve similar performance no matter how we build a pair (D_k, Q_j), and such property holds independently from the architecture used. Please note, experiments were performed on the test set of the Oxford dataset.

4.3.4 *Recall capability*

In Table 4.3, we report the recalls R@1, R@5, R@10 of the baseline approaches [4, 96] and the Δ recalls obtain by the other methods. Our evaluation scheme comprises a variation of the backbone architecture and the training method. In particular, columns indicated with T-S refers to the results obtained with a single network trained in a knowledge-distillation strategy, Naive Ensemble is an ensemble where members were trained with a standard triplet loss, and T-S Ensemble is the method proposed in this PhD work. Finally, we also report the performance obtained with an ensemble that relies on different query and database replicas [122]. At first glance, we can see a difference between the performance achieved with our method across the two different architectures. As explained in Section 4.2.4, we trained the teacher-student models by reducing the overall number of epochs since we noticed that model over-training leads to better recall in general but also to a lower uncertainty quality. With MinkLoc3D, we found the optimal trade-off between the number of training epochs and the recall performance. Unfortunately, we could not achieve similar results with PointNetVlad. A possible explanation is that ensuring both ensemble diversity and discriminative features is more challenging for this task with a less accurate architecture. In general, the approach of Mason et al. [122] slightly outperforms the other methods. However, with this technique we need to maintain separated replicas of database and queries implying n simultaneous searches to extract the most likely match, where n represents the number of ensemble members. Furthermore, as we will report in the next section, while this method achieves higher recall, it produces lower quality uncertainty.

PNV	Baseline			T-S			Naive Ens.			T-S Ens. (ours)			Mason et al. [122]		
Data	**R@1**	**R@5**	**R@10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**
O.	71.1	86.6	91.7	-2.9	-1.7	-1	0.7	0.3	0.3	-1.9	-1.1	-0.6	1.7	1.2	0.9
U.S.	66.8	84.3	90.2	0.3	-0.5	-0.6	1.2	-0.3	-0.1	1.2	0.4	0.5	2.1	0.2	-0.2
R.A.	62.7	80.5	85.6	-2.8	-1.8	-0.2	1.5	0	-0.1	-1.1	-1.2	-0.9	2	1.1	0.9
B.D.	63.8	81.7	87.2	-1.5	-3	-2	1.8	-0.2	0.2	-0.7	-1.1	-0.7	2.6	0.5	0.8
DCC	62.8	65.9	67.8	0	0.4	0.6	1.9	1.4	1.5	0.1	0.7	0.6	1.8	1.4	1.2
R.	54.9	67.4	72.3	-1.9	-1.4	-1.3	1.9	0.7	-0.1	-0.6	-0.1	-0.2	2.2	0.9	0.4

(a)

ML3D	Baseline			T-S			Naive Ens.			T-S Ens. (ours)			Mason et al. [122]		
Data	**R@1**	**R@5**	**R@10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**	**Δ1**	**Δ5**	**Δ10**
O.	92.4	97.9	98.8	0.9	0.2	0.1	-0.3	-0.3	-0.3	1.3	0.3	0.1	2.2	0.5	0.2
U.S.	84	94.5	96.4	0.7	0	0.3	1.9	0	-0.1	3.1	1.3	0.9	4.9	2	1.4
R.A.	75.9	89.8	93.1	4.2	3.5	2.1	4.5	3	2.1	6.6	4.4	2.8	9.5	5	3.4
B.D.	80	93.3	96.3	1.6	-0.2	-0.3	0.1	-0.6	-0.5	3.1	0.7	0.2	5.7	1.7	0.9
DCC	76.9	86.4	89.5	2.1	-0.6	-1.3	3.5	1.8	1	2.3	-0.4	-1.2	4.5	2	1.2
R.	56.4	69.1	74.9	-0.1	-1.9	-2.4	0.4	0.5	-0.5	0.8	-0.7	-2.1	2.1	1	0.7

(b)

Table 4.3: In this table, we show different recalls r@1, r@5 and r@10 achieved by the baseline model [96] across different datasets. For the other approaches, we report the Δ recall with respect to the baseline method. All the methods reported were trained only on the Oxford Robotcar dataset

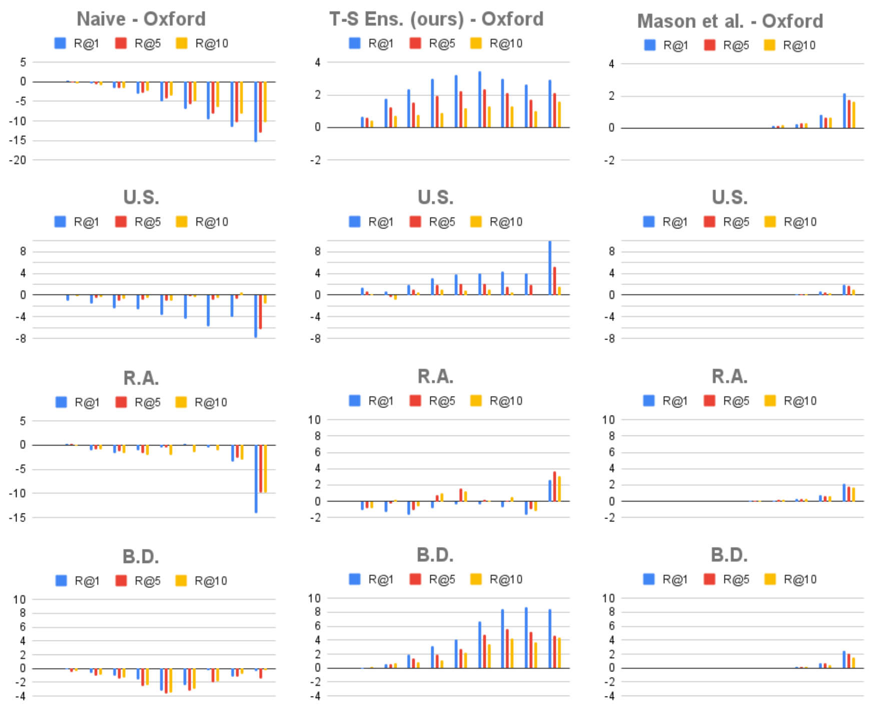


Figure 4.3: Uncertain queries removal test on *PointNetVLAD* model considering Oxford [121] and InHouse [4] datasets. x-axis represents the percentage of query removed in the interval of $[0\%, 90\%]$, while y-axis reports the $\Delta\%$ R@1, R@5 and R@10. We compare Naive Ensemble (Left), Ours (Centre) and [122] (Right) approaches.

4.3.5 *Localization Failures Detection*

A desirable behavior from an uncertainty-aware system is to provide the possibility of detecting possible failures. In the context of place recognition, we would like to decide whether a prediction is reliable or not to perform localization, *i.e.*, by fixing a threshold on the total feature uncertainty and discard EVs accordingly.

We use a similar evaluation protocol of Section 4.3. However, for each dataset and each traversal instance d_i , we order queries q_j in descending order according to the total uncertainty computed with Equation (19), and we progressively discard queries to observe the presence of an increasing trend in terms of R@1, R@5, R@10. Since we compute a new ordering for each run, the results show the average recall obtained by simulating multiple traversals of a vehicle within the navigation map. Note that the mentioned assessment pipeline has the following goal: to observe the recall capability of our system on a subset of queries that we label as reliable. We report the results obtained with PointNetVLAD in Figure 4.3 and Figure 4.5, while in Figure 4.4 and Figure 4.6 we can find the results obtained with MinkLoc3D. We decided to split the datasets in two different groups,

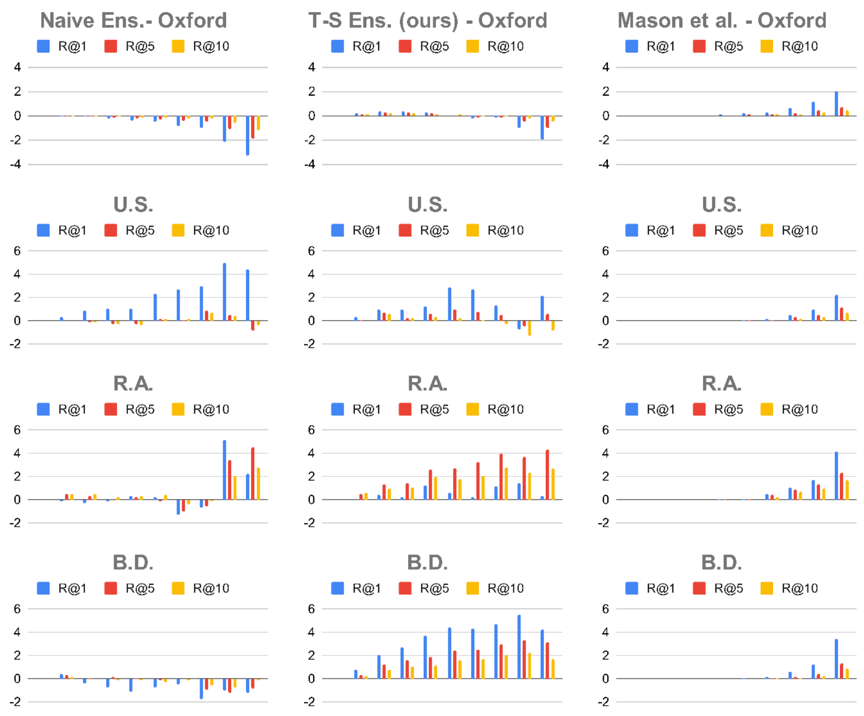


Figure 4.4: Uncertain queries removal test on *MinkLoc3D* considering Oxford [121] and InHouse [4] datasets. x-axis represents the percentage of query removed in the interval of $[0\%, 90\%]$, while y-axis reports the $\Delta\%$ R@1, R@5 and R@10. We compare Naive Ensemble (Left), Ours (Centre) and [122] (Right) approaches.

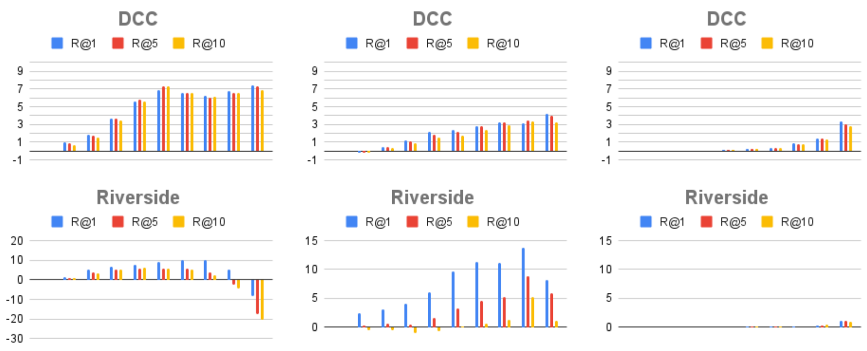


Figure 4.5: Uncertain queries removal test on *PointNetVLAD* considering MulRan [88] dataset. x-axis represents the percentage of query removed in the interval of $[0\%, 90\%]$, while y-axis reports the $\Delta\%$ R@1, R@5 and R@10. We compare Naive Ensemble (Left), Ours (Centre) and [122] (Right) approaches. Due to the worse recall obtained by all the reported methods, we consider this dataset more challenging.

since both models struggle to achieve high recall measures. Therefore, we label the former group as *trivial* and the latter group as *challenging*.

Our goal is to observe the uncertainty quality of the considered approaches in the presence of those two different scenarios. From the results obtained with our approach (center column), it is possible to determine a strong relation between high uncertainty and wrong matches. In fact, by gradually removing uncertain queries we are able to ensure a better recall capability of our system. Furthermore, our method shows excellent results within challenging scenarios as reported in Figure 4.6. The variance-based similarity approach of Mason et al. also shows improvements but after discarding a considerable amount of queries. This suggests that feature-wise variance is more powerful than similarity variance to represent the model's uncertainty. As expected, uncertainty extracted with a Naive Ensemble does not produce the desired results.

4.4 FINAL DISCUSSION

In this chapter, we provide an in-depth study of the deep ensemble effect in LiDAR-based place recognition. In particular, we propose a deep ensemble implementation that exploits a knowledge-distillation approach to approximate a unique feature space between members, and we compare it with the method of Mason *et al.* [122], which instead relies on different feature space replicas.

From our extensive experimental activity, we can conclude that both strategies increase the overall system recall. In particular, the approach of Mason *et al.* achieves slightly better performance, but

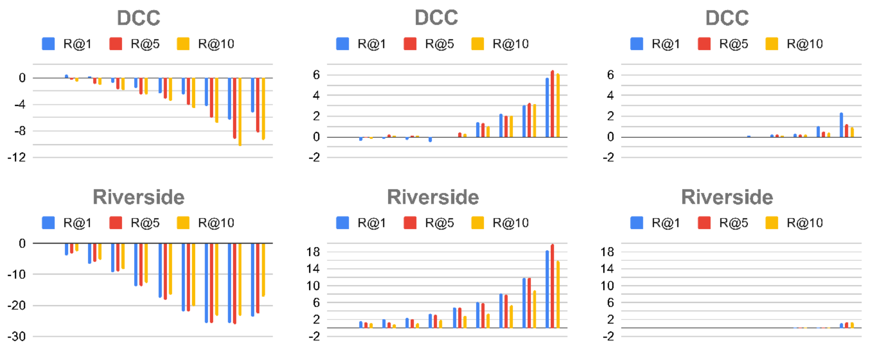


Figure 4.6: Uncertain queries removal test on *MinkLoc3D* considering MulRan [88] dataset. x-axis represents the percentage of query removed in the interval of $[0\%, 90\%]$, while y-axis reports the $\Delta\%$ R@1, R@5 and R@10. We compare Naive Ensemble (Left), Ours (Centre) and [122] (Right) approaches. Due to the worse recall obtained by all the reported methods, we consider this dataset more challenging.

our method better associates wrong predictions to high uncertainty estimates. Furthermore, our feature-level uncertainty demonstrates its effectiveness especially in challenging scenarios never represented in the training set, and we perform query-to-database search only a single time, since we exploit a unified feature space.

Furthermore, since our method estimates uncertainty directly on EV, it enables the discovery of features that generate anomalies giving a better introspection of the neural network model. This aspect put the basis for future works to improve the explainability of such systems.

Finally, the approach proposed in this chapter does not directly address the problem of loop closure detection, but it remains one of its primary applications.

5 UNCERTAINTY-AWARE DNN FOR MULTI-MODAL POSE REGRESSION

So far in this thesis, an approach that covers the entire localization pipeline was presented (Chapter 3) together with a method to address the uncertainty estimation problem in global localization (Chapter 4). Instead, in this chapter the main focus is the local refinement part of the localization pipeline and we integrate three different uncertainty estimation techniques within an existing pose regression Convolutional Neural Network (CNN), with the aim of providing uncertainty estimates with respect to the predicted output pose components. Similarly to Chapter 4, the following approach deals with the task of estimating *epistemic* uncertainty. However, in contrast with the previous chapter where uncertainty was estimated directly on the features extracted by a Deep Ensemble-based model, this work quantifies the uncertainty of the individual pose components provided by a regression CNN. This allows for the determination of an area in which the vehicle is likely to be located, since the proposed approach provides a variance measure of both translation and rotation components, that can be also used to detect and discard localization failures. Providing an uncertainty-aware localization Deep Neural Network (DNN) can be particularly useful to understand the degree of confidence that such a model has with respect to the predicted position and orientation, allowing us to explicitly use this knowledge in the decision making process. For instance, accurate positioning can be more relevant when the goal is to prevent the vehicle from encroaching the opposite lane.

Estimating epistemic uncertainty in deep learning models for pose regression has other several advantages. Firstly, reliable pose uncertainties are easier to understand from the human perspective, increasing the transparency of the deep learning model used. Secondly, since this uncertainty can be geometrically interpreted, it can be potentially used within a filtering algorithm such as *Kalman filter* [35]. Similarly to Simultaneouos Localization And Mapping (SLAM), Kalman filter aims to compensate the robot drift resulting from the intrinsic noise that affects sensors such as odometry and Inertial Measurement Units (IMUs). However, differently from SLAM, this algorithm performs localization by integrating the estimates of the so-called *motion* and *observation* models. Generally, the motion model estimate the vehicle movement, while the observation model evaluate the estimated robot location by means of the expected observation of the current environment.

Kalman filtering explicitly define the uncertainty of both models and exploit it to compute an optimal weighted average over the motion and observation estimates. Therefore, in this chapter we aim to provide a DNN for localization that can be potentially exploited in a filtering algorithm.

Regarding the proposed approach, we consider a multi-modal camera localization method, that exploits camera images, a Light Detection And Ranging (LiDAR) point cloud map and rough initial poses (Figure 5.1). In particular, the proposed technique leverages on an existing CNN model named *CMRNet* [28], that we make uncertainty-aware. In particular, this approach synthesizes a virtual point of view from which to observe the LIDAR map, given an initial rough camera pose estimate. Then, by comparing this point of view with the one provided by the actual camera image, CMRNet predicts the refined camera pose. CMRNet has several desirable properties that one would desire from a localization CNN: it achieves accurate localization, it is scalable to scenarios never seen during the training stage, and it is cost effective, since LiDAR data are matched from a pre-built 3D map of the navigation environment and not to data from expensive sensors mounted on-board a vehicle. Due to these practical reasons, this technique represented an interesting local refinement approach to be extended during this thesis work.

In the literature, only few DNN-based camera pose regression approaches exist that also estimate uncertainty [42, 80], and often they do not provide a direct pose uncertainty [134] or measure only its aleatoric part [148]. To bridge the gap within uncertainty estimation techniques for DNN-based camera localization, in this chapter we propose the integration of sampling-based and direct state-of-the-art methods for epistemic uncertainty estimation within a multi-modal camera localization CNN, and show that they can provide calibrated uncertainties and that some of them can also be used to detect localization failures. Moreover, one of the main contributions is to have developed a version of a camera localization DNN model that is able to estimate uncertainty by using Deep Evidential Regression (DER) [3]. The performance of the proposed uncertainty-aware model is evaluated in terms of localization accuracy and uncertainty quality. In particular, the assessment is performed on three different datasets: KITTI [55], KITTI 360 [112] and Argoverse2 [187]. Finally, to further validate the proposed approaches in a realistic scenario, we employ the different uncertainty-version of CMRNet as the observation model of an Extended Kalman Filter (EKF) tested on KITTI and KITTI360.

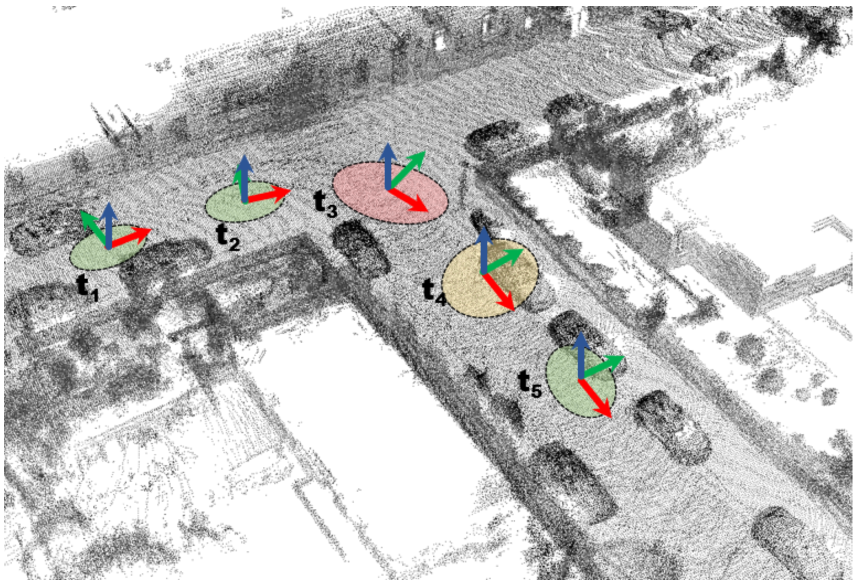


Figure 5.1: In this chapter, an uncertainty-aware localization DNN is presented. Our approach refines an initial pose estimate by comparing an image with a given LiDAR map and associate uncertainty measures to the final prediciton.

5.1 RELATED WORK

With the advent of neural networks, several DNN-based approaches for camera localization emerged in the last decade. In general, a possible categorization of methods includes two categories: camera pose regression [82, 85, 140, 148, 194] and place recognition [8, 62, 206] techniques. Using an image, pose regression techniques predict the pose of a camera, while the place recognition approaches find a correspondence with a previously visited location, depicted in another image. Multi-modal methods, which employ both images and LiDAR data, propose to jointly exploit visual information and the 3D geometry of a scene to achieve higher localization accuracy [27, 130, 188]. Recently, DNN-based methods emerged also for image-to-LiDAR-map registration. An example is CMRNet [28], which performs direct regression of the camera pose by implicitly matching RGB images with the corresponding synthetic LiDAR image generated using a LiDAR map and a rough camera pose estimate. Its ultimate goal is to refine an initial localization guess, *e.g.,* a GPS localization measure. CMRNet is map-agnostic, that is it can be deployed in any scenario where a pre-built 3D LiDAR map is given. Feng *et al.* [47] proposed another multi-modal approach, where a DNN is trained to extract descriptors from 2D and 3D patches by defining a shared feature space between heterogeneous data. Localization is then performed by exploiting points for which 2D-3D correspondences have been found. Similarly, Cattaneo *et al.* [30] proposed a DNN-based method for learning a common feature space between images and LiDAR maps to produce global descriptors, used for place recognition. Although the previous multi-modal pose regression techniques achieve outstanding results, none of them estimate the epistemic uncertainty of their predictions. This is a severe limitation, especially considering the final goal: to deploy them in critical scenarios, where it is important to detect when the model is likely to fail. Epistemic uncertainty estimation in Neural Networks (NNs) is a known problem. In the last years, different methods have been proposed to sample from the model posterior [90, 106] and, more recently, to provide a direct uncertainty estimate through evidential deep learning [3, 124, 154]. NNs uncertainty estimation gained popularity also in the computer vision field [83, 84], and different uncertainty-aware camera-based localization approaches have been proposed. For instance, Kendall *et al.* [80] introduced Bayesian PoseNet, a DNN that estimates the camera pose parameters and uncertainty by approximating the model posterior employing dropout sampling [54]. However, this uncertainty estimation method produces over-confident pose estimates. Another approach was recently proposed by Deng *et al.* [42], where an uncertainty-aware model leveraging on Bingham mixture models estimates a 6DoF pose from an image and the relative uncertainty. However, this method mostly deals with ambiguity within relatively small scenarios. Recently, Petek *et al.* [134] proposed an approach to camera localization that exploits an object detection module, which is used to enable localization within sparse HD maps. In particular, their method estimates the observer pose using the uncertainty of the objects in the HD map using a DER approach [3]. However, their technique does not employ a model that directly estimates the pose epistemic uncertainty. Another interesting approach is HydraNet [133], which is a neural network for estimating uncertainty on quaternions. The authors' model architecture incorporates an approach similar to ensemble. In particular, this model leverages on a set of distinct network heads, that are trained separately and provide distinct rotation estimates. However, this method does not address the issue of regressing a 6DoF pose. All the mentioned techniques deal with the problem of camera localization using only images, they learn to localize a camera in the environment represented in the training set. In contrast, CMRNet is map-agnostic, *i.e.,* by being able to take in input a LiDAR-map, it can perform localization also in previously unseen environments, where it is possible to better assess the effectiveness of uncertainty estimated. Furthermore, to the best of our knowledge, this is the first work to implement a DER-based approach for direct camera localization.

5.2 PROPOSED APPROACH

In our analysis of the literature, we could single out three more significant methods for estimating epistemic uncertainty in a DNN: Monte Carlo Dropout (MCD) [54], Deep Ensemble (DE) [106], and DER [3]. Although they all assume that epistemic uncertainty can be described by a normal distribution, they are different techniques and require different interventions on the network to which they are applied. Therefore, in this section, we first introduce it and then describe the modifications required in CMRNet to estimate uncertainty using each of the three different methods.

5.2.1 *Introduction to CMRNet*

CMRNet is a regression CNN used to estimate the 6DoF pose of a camera mounted on-board a vehicle navigating within a LiDAR map [28]. In particular, this model takes two different images as input: an RGB image and a LiDAR image obtained by synthesizing the map as viewed from an initial rough camera pose estimate H_{init}. CMRNet performs localization by implicitly matching features extracted from both images, and estimates the misalignment H_{out} between the initial and the camera pose.

In particular, H_{out} is computed as: $tr_{(1,3)} = (x, y, z)$ for translations, and unit quaternion $q_{(1,4)} = (q_x, q_y, q_z, q_w)$ for rotations. We propose to estimate its epistemic uncertainty by providing a reliability value for each pose component. The estimation of possible cross-correlations between the pose components has not been considered in this thesis work.

5.2.2 *Uncertainty-Aware CMRNet*

We define an input camera image with $\mathcal{I}_c$, an input LiDAR image as $\mathcal{I}_l$, a set of trained weights with $\mathcal{W}$ and an uncertainty-aware version of CMRNet as a function $f(\mathcal{I}_c, \mathcal{I}_l, \mathcal{W})$.
Monte Carlo Dropout: The idea behind MCD is to sample from a posterior distribution by providing different output estimates given a single input, which are later used for computing the mean and variance of a Gaussian distribution. This sampling is performed by randomly deactivating the weights of the fully-connected layers using a random dropout function $d(\mathcal{W}, p)$ multiple times during model inference, where p represents the dropout probability. Therefore, for MCD there is no modification of the network architecture. We applied the dropout to the regression part of the original CMRNet architecture. When many correlations between RGB and LiDAR features are found,

we expect to obtain similar samples, despite the dropout application, that is, we expect our model to be more confident with respect to its predictions. For each pose parameter μ_i, we compute the predicted value and the corresponding epistemic uncertainty as follows:

$$\begin{aligned} E\mu_i &= \frac{1}{n} \cdot \sum_n f(\mathcal{I}_c, \mathcal{I}_l, d_{regr}(\mathcal{W}, p)), \\ Var[\mu_i] &= \frac{1}{n} \cdot \sum_n (f(\mathcal{I}_c, \mathcal{I}_l, d_{regr}(\mathcal{W}, p)) - E\mu_i)^2 \end{aligned} \tag{24}$$

where n is the number of samples drawn for a given input. Please note that $E\mu_i$ and $Var[\mu_i]$, for the orientation, are computed after the conversion from unit quaternion to Euler angles.
Deep Ensemble: DE-based approaches perform posterior sampling by exploiting different models trained using different initialization of the weights, but sharing the same architecture.

Using different parameterizations of the same model leads to the recognition of a wider range of data-patterns, and to an increment of the overall accuracy [50]. On the other hand, when receiving in input patterns not well-represented in the training set, all the NNs in the ensemble would give out low-quality results, so leading to an increment of variance. In our case, we expect to obtain large epistemic uncertainty when each model identifies a different set of correspondences between RGB and LiDAR features, leading to significant different pose estimates. By training CMRNet n times with different random initializations, we obtain a set of weights $\mathcal{W}_{set} = \{\mathcal{W}_1, ..., \mathcal{W}_n\}$, which describe different local minima of the model function $f(\cdot)$. For each pose parameter μ_i we compute the predicted expected value and the corresponding epistemic uncertainty as follows:

$$\begin{aligned} E\mu_i &= \frac{1}{n} \cdot \sum_{j=1}^{n} f(\mathcal{I}_c, \mathcal{I}_l, \mathcal{W}_j), \\ Var[\mu_i] &= \frac{1}{n} \cdot \sum_{j=1}^{n} (f(\mathcal{I}_c, \mathcal{I}_l, \mathcal{W}_j) - E\mu_i)^2 \end{aligned} \tag{25}$$

where n represents the number of models of the ensemble. $E\mu_i$ and $Var[\mu_i]$ of rotations are computed after the conversion from unit quaternion to Euler angles.
Deep Evidential Regression: While adapting to MCD and DE methods does not require particular modifications of CMRNet, the technique proposed by Amini *et al.* [3] requires substantial changes both in the training procedure and in the final part of the architecture. In Deep Evidential Regression, the main goal is to estimate the parameters of a Normal Inverse Gamma distribution $NIG(\gamma, \nu, \alpha, \beta)$. A neural network is trained to estimate the NIG parameters, which are then

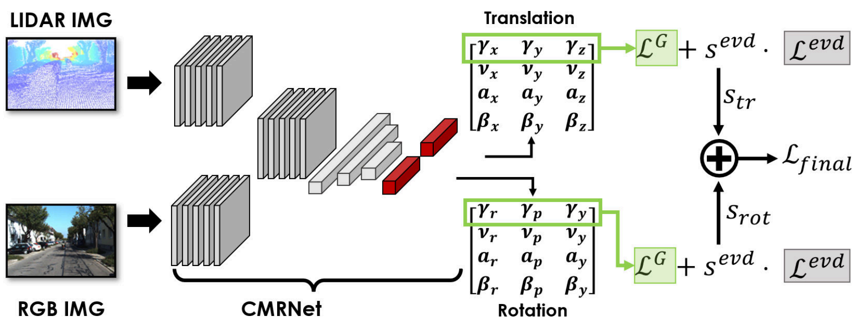

LIDAR IMG
RGB IMG
CMRNet
Translation
Rotation
$\begin{bmatrix}\gamma_x & \gamma_y & \gamma_z\\ v_x & v_y & v_z\\ \alpha_x & \alpha_y & \alpha_z\\ \beta_x & \beta_y & \beta_z\end{bmatrix}$
$\begin{bmatrix}\gamma_r & \gamma_p & \gamma_y\\ v_r & v_p & v_y\\ \alpha_r & \alpha_p & \alpha_y\\ \beta_r & \beta_p & \beta_y\end{bmatrix}$
$\mathcal{L}^G + s^{evd} \cdot \mathcal{L}^{evd}$
s_{tr}
s_{rot}
$\mathcal{L}_{final}$
$\mathcal{L}^G + s^{evd} \cdot \mathcal{L}^{evd}$

Figure 5.2: In this picture the CMRNet + DER approach is shown. The last FC-layers (red) are modified according to the method proposed by Amini *et al.* [3] for estimating the parameters $m_i = (\gamma_i, \nu_i, \alpha_i, \beta_i)$ of different Normal Inverse Gamma (NIG) distributions. During training, $\mathcal{L}^G$ (green) and $\mathcal{L}^{evd}$ (grey) loss functions are computed both for translation and rotation components.

used to compute the expected value and the corresponding epistemic uncertainty, for each pose parameter:

$$E\mu = \gamma, \qquad Var[\mu] = \frac{\beta}{\nu(\alpha - 1)} \tag{26}$$

To train the model, the authors propose to exploit the Negative Log Likelihood $\mathcal{L}^{NLL}$ and the Regularization $\mathcal{L}^R$ loss functions to maximize and regularize evidence:

$$\mathcal{L}(\mathcal{W}) = \mathcal{L}^{NLL}(\mathcal{W}) + \lambda \cdot \mathcal{L}^R(\mathcal{W}) \tag{27}$$

$$\mathcal{L}^{NLL} = -\log p(y|m) \qquad \mathcal{L}^R = \Phi \cdot |y - \gamma| \tag{28}$$

where $\Phi = 2\nu + \alpha$ is the amount of evidence, see [3] for details, and λ represents a manually-set parameter that affects the scale of uncertainty, $p(y|m)$ represents the likelihood of the NIG. Note that, $p(y|m)$ is a *pdf* that follows a t-Student distribution $St(\gamma, \frac{\beta(1+\nu)}{\nu\alpha}, 2\alpha)$ evaluated with respect to a target y. In evidential deep learning, evidence Φ represents a quantity in favor of a particular property, that is the confidence that a model associates to a certain output. Thus, with $\mathcal{L}^R$ we aim to scale the prediction error according to the model confidence and make to model capable of producing poor evidence in presence of out-of-distribution (OOD) data, and inflated evidence otherwise. For a complete description of loss functions and theoretical aspects of DER, please refer to the work of Amini *et al.* [3]. To integrate DER within CMRNet, we need to deal with the following issues: how to apply DER for regressing multiple parameters, how to manage rotations, and how to aggregate the results when computing the final loss. We changed the last FC-layers, which predict the rotation $q_{(1,4)} = (q_x, q_y, q_z, q_w)$

and translation $tr_{(1,3)} = (x, y, z)$ components, in order to estimate the NIG distributions associated to each pose parameter. As it can be seen in Fig. 5.2, we modified CMRNet to regress Euler angles instead of quaternions, then we changed the FC-layers to produce the matrices $eul_{(4,3)}$ and $tr_{(4,3)}$, where each column $|\gamma_i, \nu_i, \alpha_i, \beta_i|'$ represents a specific NIG [3]. Since the original CMRNet model represents rotations using unit quaternions $q_{(1,4)}$, we cannot compute the $\mathcal{L}^{NLL}$ and $\mathcal{L}^{R}$ loss functions directly, as addition and multiplication have different behavior on the S^3 manifold. As mentioned above, we modified the last FC-layer of CMRNet to directly estimate Euler angles $eul_{(1,3)} = (r, p, y)$. We also substitute the quaternion distance-based loss used in [28] with the smooth $\mathcal{L}_1$ loss [56], which will be later used also in $\mathcal{L}^{R}$ and $\mathcal{L}^{D}$, by also considering the discontinuities of Euler angles. Although the Euler angles representation is not optimal [151], it allows for easier management of the training procedure and enables a direct comprehension of uncertainty for rotational components. As we will demonstrate in Sec. 5.3, this change does not produce a significant decrease in accuracy. Since CMRNet performs multiple regressions, it is necessary to establish an aggregation rule for the $\mathcal{L}^{NLL}$ and $\mathcal{L}^{R}$ loss functions, which are computed for each predicted pose parameter. With the application of the original loss as in [3] we experienced unsatisfactory results. We are under the impression that, in our task, $\mathcal{L}^{NLL}$ presents an undesirable behavior: since the negative logarithm function is calculated over a probability density, it is not lower bound, as the density gets near to be a delta.

We propose to overcome the previous issues by avoiding the computation of the logarithm and considering a distance function that is directly based on the probability density $p(y|m)$, that is the pdf of the t-Student distribution. Therefore, we replaced $\mathcal{L}^{NLL}$ with the following loss $\mathcal{L}^{D}$ and we also reformulate $\mathcal{L}^{R}$:

$$\mathcal{L}^{D} = \frac{1}{n} \cdot \sum_{i=1}^{n} d(p(y_i|m_i)^{-1}, 0) \quad \mathcal{L}^{R} = \frac{1}{n} \cdot \sum_{i=1}^{n} d(y_i, \gamma_i) \cdot \Phi_i \quad (29)$$

Similarly to $\mathcal{L}^{NLL}$, the idea behind $\mathcal{L}^{\mathcal{D}}$ is to penalize predictions according to the confidence level output by our model with respect to the deviation between a target and an estimated values. However, since this loss function admits a lower bound and is defined in the positive interval, it allows direct computation of a distance metric $d(\cdot)$ on the vector of inverse densities. To ensure a better numerical stability, we clip $p(y_i|m_i)$ when it returns too low density values, *i.e.*, < 0.04. Regarding $\mathcal{L}^{R}$, we simply scale the distance error on each pose component with the respective evidence. We the compute the mean error by managing rotations and translations separately. The final evidence loss is computed as follows:

$$\mathcal{L}^{evd} = \mathcal{L}^{D} + \lambda \mathcal{L}^{R} \tag{30}$$

We noticed that the localization accuracy achieved was not satisfactory when employing only $\mathcal{L}^{evd}$ during training. Therefore, we opted to also employ the original geometric loss function $\mathcal{L}^{G}_{tr}$ used in [28], and to employ the smooth L1 loss on rotations as geometric loss $\mathcal{L}^{G}_{rot}$.

The overall loss is therefore computed as follows:

$$\mathcal{L}_{rot} = \mathcal{L}^{G}_{rot} + s^{evd}_{rot} \cdot \mathcal{L}^{evd}_{rot} \qquad \mathcal{L}_{tr} = \mathcal{L}^{G}_{tr} + s^{evd}_{tr} \cdot \mathcal{L}^{evd}_{tr} \tag{31}$$

$$\mathcal{L}_{final} = s_{rot} \cdot \mathcal{L}_{rot} + s_{tr} \cdot \mathcal{L}_{tr} \tag{32}$$

where the s hyper-parameters represent scaling factors.

5.2.3 *Integration within an EKF*

Although the previously presented uncertainty-aware versions of CMRNet enable the estimation of epistemic uncertainty, it remains unclear how to exactly take advantage of such estimates. A possible application is to exploit uncertainty for establishing whether the model output is reliable or not, *e.g.*, by fixing a threshold on the computed uncertainty and discarding those outputs whose uncertainty is larger than the threshold. However, we argue that, despite a threshold-based strategy can be handy to discover possible failures, its deployment in a more realistic scenario is limited. Therefore, in order to fully exploit the proposed uncertainty-aware CMRNet, we propose its integration within an EKF, where CMRNet represents the observation model.

Since our goal is to deploy our CNN on a terrain autonomous vehicle, we used a 3DoF velocity model [169] where the vehicle state at time t is $x_t = (x, y, \theta)$. In particular, the robot motion is controlled through translation and rotational velocities $u_t = (v, \omega)$ whose direct measurement is provided by an IMU. After Δt time, the next robot state is computed by using the following equation:

$$x_{t+\Delta t} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix} \tag{33}$$

Please note, although we employ a 3DoF velocity model, the Kalman state use a 6DoF representation, *i.e.*, z, $\mathtt{roll}$ and $\mathtt{pitch}$ are initialized with an initial guess, but then they are not altered by the motion model.

Initially, since we consider only three pose parameters, one can argue that such a velocity model is too simple for a real application. However, our aim is to validate the effectiveness of the proposed uncertainty estimation methods also when the filtering algorithm relies on an approximated motion model.

After estimating the next robot state, the new pose is used to synthesize a virtual point of view. This allows for obtaining a LIDAR image. Then, by providing to CMRNet the camera and LiDAR images at time $t + \Delta t$, our model estimates the error of the motion model state and jointly provides the uncertainties associated to the predicted pose parameters. The uncertainty estimates are used to define a diagonal covariance matrix $Q_{6,6}$, representing the measurement covariance.

In order to preserve the Normal assumption of both motion and observation models required by the EKF, we have to perform linearization both for the state transition matrix G [169] and the measurement matrix H by computing the respective jacobian matrices. Since, CMRNet provides direct observations of Kalman states, the computation of $\mathtt{jacobian}(H)$ is straightforward.

A general overview of the EKF algorithm is provided in Alg. 2. For the detailed implementation of the EKF prediction and update steps, please refer to [169].

Algorithm 2: EKF Algorithm

```
**Function** EKF_step($x_t$, $u_t$, $\Delta t$, $R$):
    new_state := EKF_prediction($x_t$, $u_t$, $\Delta_t$, R);
    RGB, LiDAR = synthezize_view(new_state) ;
    z, Q := CMRNet_estimate(RGB, LiDAR);
    new_state = EKF_measurement_update(z, Q);
    **return** *new_state*
**Function** main($x_0$):
    define initial state covariance matrix $P_{6x6}$;
    define state transition covariance matrix $R_{6x6}$;
    $x_t$ := EKF_init($x_0$, P);
    **while** *new control* $u_t$ *is received* **do**
        compute $\Delta t$ ;
        $x_t$ := EKF_step($x_t$, $u_t$, $\Delta t$, R);
    **end**
```

5.3 EXPERIMENTAL ACTIVITY

The experimental activity described in the following section has a dual purpose. On the one hand, it proves that the localization performances of the proposed models achieve comparable results concerning the original CMRNet implementation, providing at the same time reliable uncertainty estimates. On the other hand, we propose two possible applications of the estimated uncertainties: a rejection scheme for detecting localization failures and the integration of the proposed approach within an EKF.

5.3.1 *Dataset*

The proposed models were trained and tested on three different datasets: KITTI [55], KITTI360 [112] and Argoverse2 [187]. Each dataset was recorded within an outdoor environment and includes camera images, LiDAR point clouds and accurate groundtruth poses. Please note, this aspect implies that for each proposed method we have three distinct training procedures, *i.e.,* one for each dataset.

Regarding the KITTI dataset, we followed a similar experimental setting proposed in [28] by using images and LiDAR data from sequences 03 to 09 (~ 10k samples) for training, and sequence 00 (~ 4.5k samples) for the assessment of the estimated uncertainty quality. Run 00 presents a negligible overlap of approximately 4% compared to the other sequences, *i.e.,* resulting in a fair validation containing a different environment never seen by CMRNet at training time. Please note, the tuning of hyperparameters reported in Section 5.3.2 and the ablation study proposed in the Section 5.3.4 were performed only on this dataset. We exploited the ground truth poses provided by [15] to create accurate LiDAR maps. In addition, in this work we considered the KITTI360 and Argoverse2 datasets. With KITTI360 dataset, sequences from 03 to 10 (~ 40k samples)were used for training, run 02 (~ 10.5k samples) for testing and sequence 00 (~ 11.5k samples) for validation.

In the case of Argoverse2, the first 4 training runs (~ 64k samples) were used for training, validation run 0 (~ 15.6k samples) was used for testing and validation run number 1 (~ 20k samples) for finally validating the proposed methods. Since images included in this dataset present a very large field of view, images were resized and then cropped to have a same size of KITTI images by taking inspiration from [31].

To mimic real-life usage and differently from [28], for each dataset we removed all dynamic objects (*e.g.,* cars and pedestrians) from within the LiDAR maps, allowing some mismatches between the RGB image and the LiDAR image. Such a removal makes the task more difficult since now CMRNet has also to implicitly learn how to discard incorrect matches. This aspect also explains why we used validation run 1 of Argoverse2 dataset to test our models, since Argoverse2 testing runs does not provide 3D objects groundtruth labels.

Finally, to simulate the initial rough pose estimate, we added uniformly distributed noise both on translation $[-2m; +2m]$ and rotation components $[-10°; +10°]$. Please refer to [28] for more details about the pre-processing operations that were also used for KITTI360 and Argoverse2 datasets.

5.3.2 *Training Details*

For all three methods (*i.e.,* MCD, DE, DER), we followed a similar training procedure as in [28]. We trained all models from scratch for a total of 400 epochs, by fixing a learning rate of $1e^{-4}$, by using the ADAM optimizer and a batch size of 24 on a single NVidia GTX1080ti. The code was implemented with the PyTorch library [131]. CMRNet is a map agnostic model and can be potentially deployed in any scenario where a 3D map is given. However, the model is camera dependent, since the camera intrinsics affect both RGB images and LiDAR projections. Therefore, when a different dataset is considered, we re-trained the model from scratch.

Regarding the KITTI360 dataset, due to the considerable amount of samples, each epoch a subset of random samples were selected for a total amount of 20k. On the one hand, we still used all the training data, but we sped up the training process without decreasing the model performance. A similar approach did not lead to optimal results with Argoverse2, forcing us to exploit the entire training set. It is likely that this issue arose due to the pre-processing operations performed on the input images.

Concerning the DE models, random weights initialization was performed by defining a random seed before each training.

For DER we initially fixed the scaling parameters $(s_{rot}, s_{tr}, \lambda_{rot}\lambda_{tr}) = (1., 1., 0.01, 0.1)$ and $(s^{evd}_{rot}, s^{evd}_{tr}) = (0.1, 0.1)$. However, we experienced an increment of $\mathcal{L}^{evd}$ after approximately 150 epochs. Therefore, we decided to stop the training, change $(s^{evd}_{rot}, s^{evd}_{tr}) = (5e^{-3}, 5e^{-3})$, and then proceed with the training. This modification mitigated overfitting. Deactivating $\mathcal{L}^{evd}$ during the second training step led to uncalibrated uncertainties. Please note, that those hyperparameters were optimized by observing the results only on the KITTI dataset.

5.3.3 *Evaluation metrics*

We evaluated the proposed methods by comparing both localization estimates and uncertainty calibration accuracies. In particular, we assessed the localization by measuring the euclidean and quaternion distances between the ground truth and the estimated translation/rotation components. Note that, differently from [28], our primary goal is not to minimize the localization error. Instead, we aim to provide a reliability estimate by means of epistemic uncertainty estimation without undermining CMRNet performance.

We verified the accuracy of the estimated uncertainty using the calibration curves proposed by Kuleshov *et al.* [102]. From a technical perspective, a well-calibrated uncertainty-estimation curve should follow a $y = x$ trend. This procedure allows us to reveal whether the trained model produces inflated or underestimated uncertainties, by comparing the observed and the ideal confidence level. In this work, different curves are shown for each pose component, since the proposed models produce distinct uncertainty estimates for the position and rotation parameters.

5.3.4 *Localization assessment*

Our experimental activities encompass the evaluation of the localization performances using all the methods presented in Section 5.2, with respect to the original CMRNet proposal. Concerning CMRNet + MCD, we applied the dropout to the FC layers with a probability of 0.3 and obtained the approximated epistemic uncertainty by exploiting 30 samples. Our extensive experimental activity proves this setting provides the best trade-off between accuracy, uncertainty calibration, and computational time. We implemented a similar approach to identify the suitable number of networks as regards the CMRNet + DE approach. Here we identified the best performances in using 5 networks, not noticing any performance gain by adding more models to the ensemble.

Table 5.1 shows the obtained localization results, together with the statistics of the initial rough pose distribution for all the three considered datasets. MCD decreases the performances of the original CMRNet in all the three cases, resulting in the worst method among those evaluated. On the other hand, CMRNet + DE achieves the best results in terms of accuracy, at the expense of having to train and execute n different networks. This method reduces the errors' standard deviation, as expected from ensemble-based method.

Finally, it was found that CMRNet + DER achieves results comparable to the original CMRNet implementation, indicating that our modifications did not have any negative effect on accuracy. However, the model shows slightly worse performance in terms of translation error when tested on the Argoverse2 dataset. As the loss function hyperparameters were fixed based on an ablation study conducted on KITTI dataset, it is possible that this parameterization may not be optimal with respect to translation error. However, CMRNet + DER still achieves high accuracy in terms of rotation error. Moreover, as

Table 5.1: Localization Results

KITTI	**Transl. Error (m)**		**Rot. Error (deg)**	
	median	mean/std	median	mean/std
Rough Initial Pose	1.88	1.82 ± 0.56	9.8	9.6 ± 2.8
CMRNet (no iter)	0.52	0.65 ± 0.45	1.3	1.6 ± 1.2
CMRNet + MCD	0.58	0.69 ± 0.44	1.8	2.1 ± 1.3
CMRNet + DE	0.47	0.57 ± 0.39	1.2	1.5 ± 1.1
CMRNet + DER	0.54	0.65 ± 0.46	1.8	2.1 ± 1.4
KITTI360	**Transl. Error (m)**		**Rot. Error (deg)**	
	median	mean/std	median	mean/std
Rough Initial Pose	1.87	1.82 ± 0.56	9.8	9.6 ± 2.8
CMRNet (no iter)	0.40	0.48 ± 0.35	1.2	1.3 ± 0.8
CMRNet + MCD	0.44	0.52 ± 0.34	1.8	1.9 ± 1.0
CMRNet + DE	0.33	0.40 ± 0.29	1.0	1.2 ± 0.7
CMRNet + DER	0.39	0.48 ± 0.35	1.6	1.8 ± 1.0
Argoverse2	**Transl. Error (m)**		**Rot. Error (deg)**	
	median	mean/std	median	mean/std
Rough Initial Pose	1.89	1.84 ± 0.56	9.8	9.6 ± 2.8
CMRNet (no iter)	0.58	0.71 ± 0.49	1.1	1.3 ± 0.9
CMRNet + MCD	0.63	0.74 ± 0.47	1.6	1.8 ± 1.0
CMRNet + DE	0.52	0.61 ± 0.42	0.9	1.0 ± 0.7
CMRNet + DER	0.63	0.77 ± 0.53	1.5	1.8 ± 1.1

Localization results of different CMRNet versions. We present the results of the original model without any iterative refinement (no iter), but the same strategy proposed in [28] could be applied to all the other methods. Note that, we do not generally alter CMRNet accuracy with the proposed DER-based approach.

Table 5.2: Ablation study - CMRNet + DER

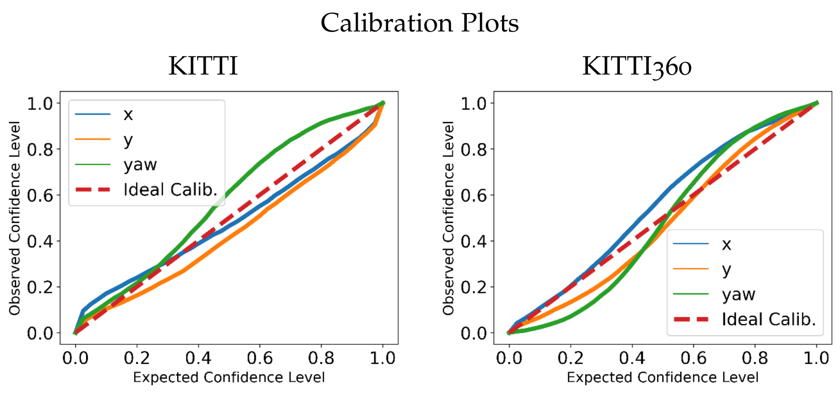
$\mathcal{L}^{evd}$	$\mathcal{L}^{G}$	s^{evd}	**Loc. Error (mean/std)**		**Calib. Error (mean/std)**	
			Tr. (m)	**Rot. (°)**	**Tr.**	**Rot.**
$\mathcal{L}^{NLL}$	-	1.	1.23 ± 0.57	2.0 ± 1.7	.080 ± .069	.135 ± .082
$\mathcal{L}^{D}$	-	1.	0.91 ± 0.53	2.6 ± 1.5	.041 ± .041	.080 ± .074
$\mathcal{L}^{NLL}$	✓	$1e^{-1}$	0.90 ± 0.56	1.8 ± 1.4	.090 ± .056	.172 ± .120
$\mathcal{L}^{D}$	✓	$1e^{-1}$	074 ± 0.49	2.5 ± 1.4	.035 ± .027	.093 ± .079
$\mathcal{L}^{NLL}$	✓	$5e^{-3}$†	0.68 ± 0.49	**1.7 ± 1.3**	.107 ± .073	.150 ± .010
$\mathcal{L}^{D}$	✓	$5e^{-3}$†	**0.65 ± 0.46**	2.1 ± 1.4	**.063 ± .040**	**.076 ± .060**

† is the two training steps procedure described in Section 5.2. The best results achieved among the two approaches that exhibit similar localization performance with respect to the original CMRNet (last two rows) are highlighted in bold. It is evident that the proposed implementation (last row) exhibits superior accuracy and calibration results. Please note, this ablation study was conducted on KITTI dataset.

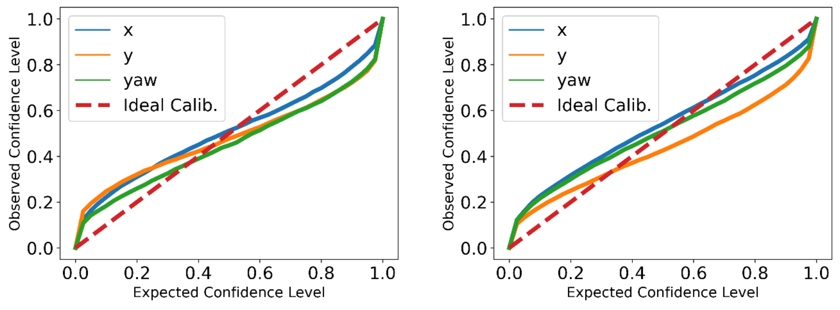
described in the following section, it also demonstrates to produce high-quality uncertainty. In any case, some applications may benefit from this approach as it provides a direct estimate of epistemic uncertainty., *i.e.,* a reduced computational time and space required for inference, because of the absence of sampling. Table 5.2 reports a brief ablation study performed to find the optimal training parameterization from which we obtained the best DER-based model (last row). As already mentioned, this study was performed on KITTI dataset.

5.3.5 *Uncertainty Calibration*

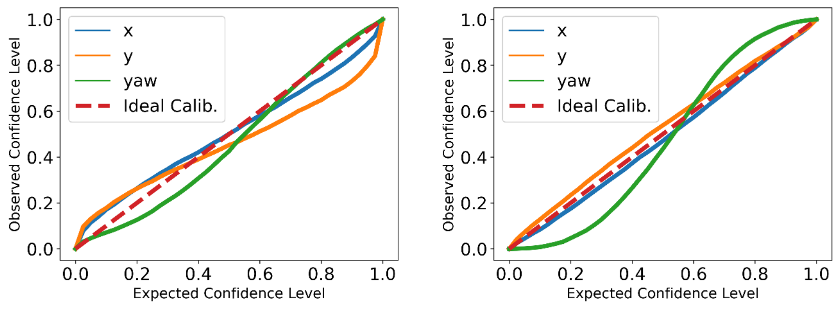
The quality of the uncertainty estimates, *i.e.,* the mean calibration errors for the translation and rotation components, are reported in Table 5.3. The errors represent the mean distances between the ideal (*i.e.,* $y = x$) and the observed calibration, for each confidence interval. Furthermore, in Figures 5.3 and 5.4 we show the calibration curves of the most relevant pose parameters. Only the x, y, and yaw components are reported since these are typically the most important parameters in the localization system of an urban terrain vehicle. All three methods obtain good uncertainty calibration, *i.e.,* they provide realistic quantities. However, CMRNet + DER shows a better performance in terms of mean calibration errors, considering the most important pose parameters for a ground vehicle (x, y, and yaw). Having a well-calibrated uncertainty-aware model with normal distributions has a major advantage, as its realistic uncertainty estimates can be employed within error filtering algorithms, such as Kalman filters.



(a) Monte Carlo Dropout



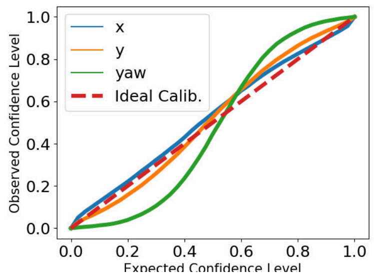
(b) Deep Ensemble



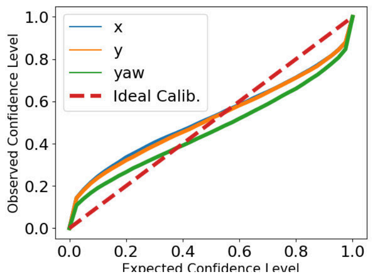
(c) Deep Evidential Regression

Figure 5.3: Calibration curves computed on KITTI and KITTI360 validation sets.

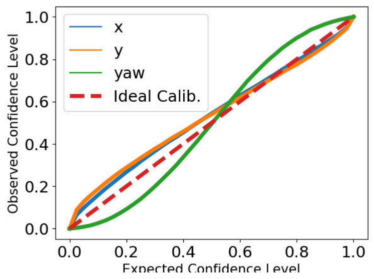
Calibration Plots - Argoverse2 dataset



(a) Monte Carlo Dropout



(b) Deep Ensemble



(c) Deep Evidential Regression

Figure 5.4: Calibration curves computed on Argoverse2 validation set.

Table 5.3: Calibration Errors - mean/std

KITTI	**CMRNet+MCD**	**CMRNet+DE**	**CMRNet+DER**
x	0.045 ± 0.025	0.077 ± 0.040	**0.042 ± 0.023**
y	**0.066 ± 0.032**	0.093 ± 0.056	0.081 ± 0.052
z	0.148 ± 0.082	0.062 ± 0.036	0.067 ± 0.027
roll	0.126 ± 0.069	0.068 ± 0.033	0.080 ± 0.043
pitch	0.162 ± 0.092	0.050 ± 0.041	0.106 ± 0.063
yaw	0.069 ± 0.049	0.089 ± 0.057	**0.042 ± 0.035**
KITTI360	**CMRNet+MCD**	**CMRNet+DE**	**CMRNet+DER**
x	0.054 ± 0.044	0.064 ± 0.040	**0.018 ± 0.010**
y	0.042 ± 0.028	0.092 ± 0.061	**0.026 ± 0.013**
z	0.171 ± 0.098	0.045 ± 0.022	0.080 ± 0.056
roll	0.157 ± 0.092	0.149 ± 0.088	0.098 ± 0.054
pitch	0.162 ± 0.091	0.123 ± 0.069	0.108 ± 0.069
yaw	0.076 ± 0.042	**0.067 ± 0.038**	0.092 ± 0.052
Argoverse2	**CMRNet+MCD**	**CMRNet+DE**	**CMRNet+DER**
x	**0.028 ± 0.014**	0.079 ± 0.046	0.037 ± 0.022
y	**0.036 ± 0.022**	0.076 ± 0.043	0.046 ± 0.030
z	0.146 ± 0.081	0.060 ± 0.038	0.053 ± 0.032
roll	0.146 ± 0.081	0.056 ± 0.036	0.099 ± 0.054
pitch	0.161 ± 0.091	0.066 ± 0.042	0.099 ± 0.055
yaw	0.106 ± 0.059	0.078 ± 0.047	**0.067 ± 0.035**

Table 5.4: Localization Results - Discarded Predictions

KITTI	**Transl. Error (m)**		**Rot. Error (deg)**		**Discarded**
	median	mean/std	median	mean/std	**Pred.**
MCD	0.58	0.68 ± 0.43	1.7	2.0 ± 1.2	27.2%
DE	0.42	0.50 ± 0.32	1.1	1.3 ± 0.8	24.7%
DER	0.49	0.58 ± 0.38	1.6	1.9 ± 1.1	**22.0**%
KITTI360	**Transl. Error (m)**		**Rot. Error (deg)**		**Discarded**
	median	mean/std	median	mean/std	**Pred.**
MCD	0.51	0.52 ± 0.34	1.7	1.8 ± 1.0	27.5 %
DE	0.29	0.34 ± 0.22	1.0	1.1 ± 0.6	24.9 %
DER	0.35	0.41 ± 0.26	1.5	1.6 ± 0.8	**23.8** %
Argoverse2	**Transl. Error (m)**		**Rot. Error (deg)**		**Discarded**
	median	mean/std	median	mean/std	**Pred.**
MCD	0.62	0.73 ± 0.46	1.5	1.7 ± 0.9	28.5 %
DE	0.46	0.54 ± 0.36	0.8	0.9 ± 0.5	25.3 %
DER	0.57	0.67 ± 0.45	1.4	1.6 ± 0.8	**22.6** %

5.3.6 *Inaccurate Predictions Detection*

By measuring the calibration error, we test the ability of an uncertainty estimator to produce realistic uncertainties. However, we still need to prove a direct proportion between the DNN prediction error and the corresponding uncertainty degree. Besides offering realistic uncertainty estimates, an uncertainty-aware model should assign a large uncertainty to an inaccurate prediction [3]. For instance, a higher level algorithm could exploit a CMRNet estimate according to its associated uncertainty, *e.g.,* by deciding whether to rely only on the measure provided by a Global Navigation Satellite System (GNSS) or even the subsequent correction performed by the CNN.

To assess that the proposed model provides large uncertainties in presence of very inaccurate predictions, we introduce the following threshold-based strategy. For both translation and rotation, we compute the trace of the covariance matrix and compare them to a threshold that allows us to discard predictions with large uncertainty. In particular, the covariance matrix is a diagonal matrix that reports the variances of the pose parameters predicted by the uncertainty-aware CMRNet. We define two distinct covariance matrices, that is C_{tr} for translations and C_{rot} for rotations, from which we compute traces $tr(C_{tr})$ and $tr(C_{rot})$.

Rather than deciding an arbitrary value for the thresholds, we use the value at the top 15% of the traces of the entire validation set, respectively for translation and rotation. A prediction is discarded if the following condition is not satisfied: $tr(C_{tr}) \notin A \land tr(C_{rot}) \notin B$, where A and B represent the sets of predictions whose uncertainty traces does not belong to the top 15% percentile.

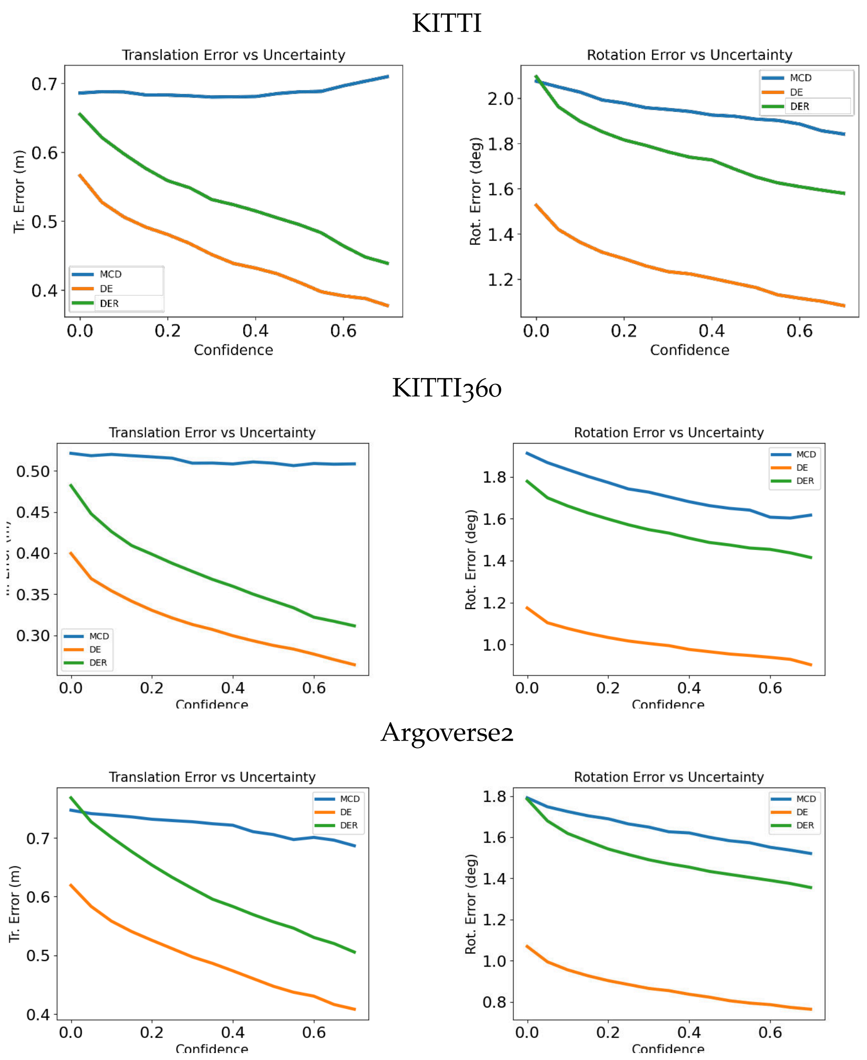


Figure 5.5: Prediction errors *vs* CMRNet confidence level. High confidence coincides with small uncertainty (except for MCD). Blue color corresponds to MCD, orange to DE, and green to DER. With DE and DER we can assign large uncertainty to inaccurate predictions.

The aim of this assessment is to observe the impact of the discarding strategy on model performance when a small number of samples are labelled as *unreliable*. We would like to observe an increment of localization accuracy by also minimizing the amount of discarded predictions, *i.e.,* we argue that unreliable poses should present large traces for both $\mathrm{tr}(C_{\mathrm{tr}})$ and $\mathrm{tr}(C_{\mathrm{rot}})$.

In Table 5.4 we report the translation and rotation errors, together with the percentage of discarded predictions from the validation set of each dataset. As can be seen, with CMRNet + DE we are able to detect inaccurate estimates and improve the overall accuracy. With CMRNet + DER we obtain a large localization improvement, outperforming the original model. Furthermore, CMRNet + DER discards

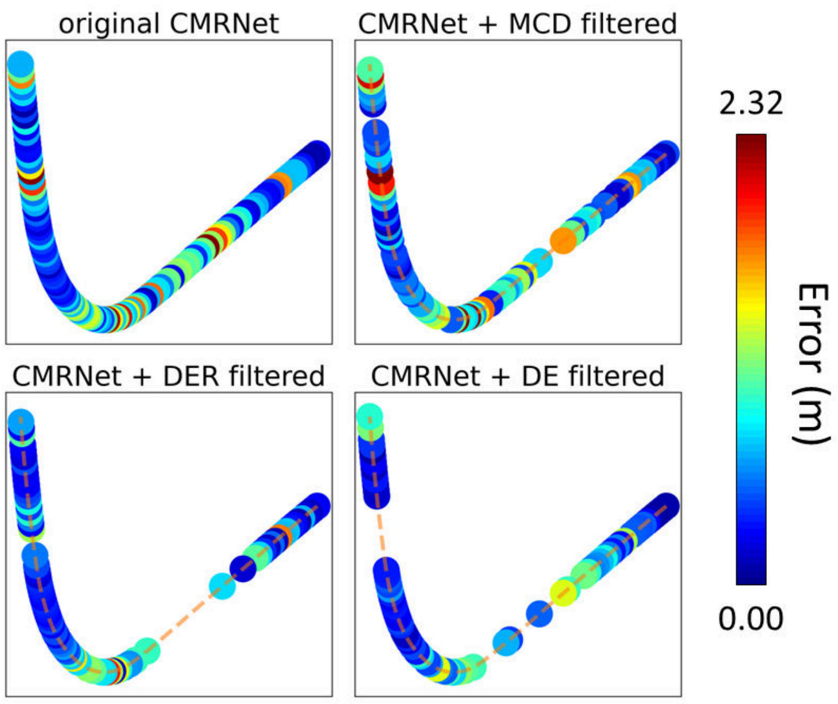


Figure 5.6: Qualitative comparison between original CMRNet and our uncertainty aware models on a slice of the KITTI 00 run. While the original CMRNet provides inaccurate estimates in the proximity of the depicted curve, CMRNet + DE and CMRNet + DER are able to identify localization failures and finally to discard them.

fewer predictions than the other methods, which means that it is able to produce more consistent uncertainties with respect to the different pose components. Although CMRNet + MCD provides good uncertainty calibration, this model is not able to produce inflated uncertainty estimates corresponding to poor prediction accuracy. In fact, we obtain the same localization results reported in Table 5.1 even though such a method discards the largest amount of samples.

In Fig. 5.5, we report the localization accuracy of each proposed method by varying the top% threshold used for discarding predictions. As can be seen, when the model confidence increases (low uncertainty), its accuracy increases as well. Another advantage of CMRNet + DE and CMRNet + DER is shown in Fig. 5.6. In this qualitative analysis, each plot represents the same piece of the path (125 frames) of the KITTI 00 run; in this curve, all methods show large localization errors. However, by exploiting DE and DER we are able to detect most localization failures. This is an interesting property since both DE and DER can also be exploited as a tool to discover in which scenes CMRNet is likely to fail, even for datasets without an accurate pose groundtruth.

5.3.7 *EKF results*

To further assess the effectiveness of the different proposed uncertainty estimation strategies, *i.e.,* MCD, DE, DER, together with the localization capability of CMRNet, we show the results obtained with the proposed implementation of the EKF described in Section 5.2.3. In particular, the proposed evaluation is performed on KITTI and KITTI360 datasets by considering sequence 00 in both cases, and it aims to simulate a realistic use-case scenario. Unfortunately, since Argoverse2 dataset mainly includes runs with a short path, it is not suitable for this type of evaluation. In particular, each uncertainty-aware model version is integrated within the EKF as the observation model that dynamically provides its own uncertainty estimates Q. Please note, we re-trained each model according to the dataset considered during evaluation. However, the data included in validation sequences was never used during the training procedure, *i.e.,* they represents a novel environment for uncertainty-aware CMRNet.

Regarding the EKF parameters, we set the elements of the diagonal covariance matrices P and R, representing the initial state uncertainty and the state transition uncertainty respectively, by aiming to find the optimal trade-off between the velocity model and observation model estimates. In particular, we fix the uncertainty translation and rotation parameters of P as $\sigma^2_{tr,P} = 0.5$ and $\sigma^2_{rot,P} = 0.125$. Similarly, we set $\sigma^2_{tr,R} = 0.5$ and $\sigma^2_{rot,R} = 0.0625$ for the covariance matrix R. Please note, these parameters are tuned by observing the filter performance only on KITTI dataset, and then we employ the same parametrization during the assessment performed on KITTI360. Finally, we use the IMU data provided by both datasets to update the localization estimate of our motion model.

The results of the proposed evaluation on KITTI dataset are reported in Table 5.5, while Table 5.6 show performance on KITTI360 dataset. Please note that the naming convention used, such as EKF + DER, refers to an implementation of the EKF that utilizes the CMRNet + DER observation model. Moreover, Figure 5.7 shows the heatmaps representing the trajectory error of the different EKF implementations, *i.e.,* different observation models. We report also the results obtained by the velocity model, that obviously achieves poor localization accuracy due to the cumulative drift. Please note that similar results are obtained by correcting the velocity model with CMRNet without employing any filtering algorithm. On KITTI dataset we observe that the EKF implementation that exploits EKF + DER achieves comparable results with respect to sampling-based methods, but with the advantage of utilizing a single model, avoiding any sampling and reducing the computational costs. However, on KTTI360 dataset CMRNet + DE

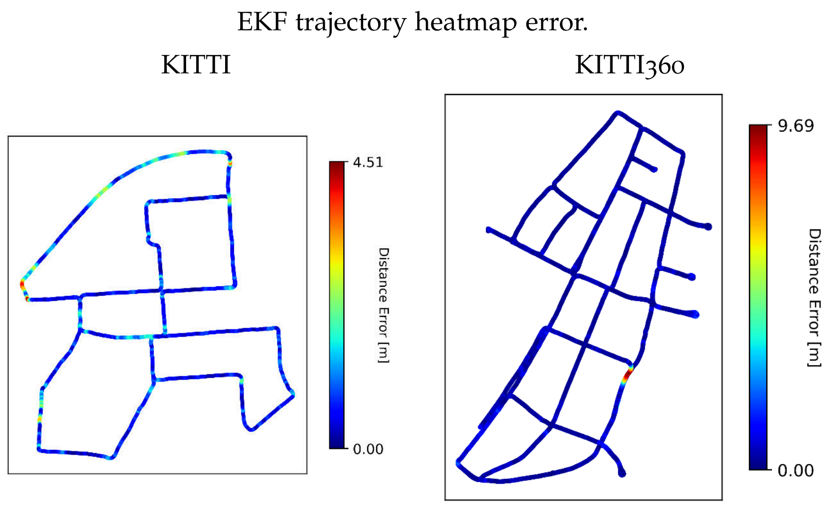
Table 5.5: EKF Results - KITTI

KITTI	**Translation Error (m)**			
	Motion model	**EKF + MCD**	**EKF + DE**	**EKF + DER**
max	297.22	4.50	**3.99**	4.88
mean/std	127.93 ± 84.86	**0.82 ± 0.61**	**0.80 ± 0.56**	**0.79 ± 0.61**
median	108.62	0.66	**0.65**	**0.63**
	Rotation Error (deg)			
	Motion model	**EKF + MCD**	**EKF + DE**	**EKF + DER**
max	21.56	24.09	**10.20**	**10.24**
mean/std	7.37 ± 5.60	1.49 ± 2.10	**1.31 ± 1.43**	**1.34 ± 1.28**
median	6.18	0.93	**0.81**	0.97

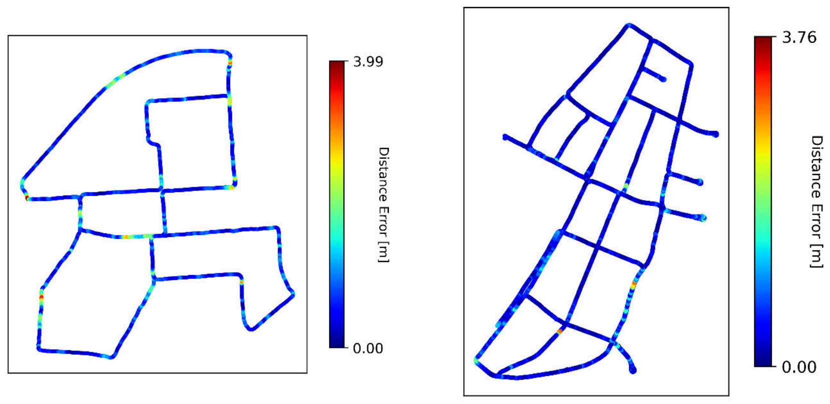
Table 5.6: EKF Results - KITTI360

KITTI 360	**Translation Error (m)**			
	Motion model	**EKF + MCD**	**EKF + DE**	**EKF + DER**
max	3967.20	9.69	**3.76**	11.95
mean/std	3472.73 ± 236.92	0.47 ± 0.69	**0.43 ± 0.37**	0.60 ± 0.80
median	3480.25	**0.34**	**0.32**	0.43
	Rotation Error (deg)			
	Motion model	**EKF + MCD**	**EKF + DE**	**EKF + DER**
max	174.88	11.46	10.93	**6.43**
mean/std	136.79 ± 35.57	0.86 ± 0.89	**0.79 ± 0.83**	0.99 ± 0.92
median	144.83	**0.60**	**0.54**	0.73

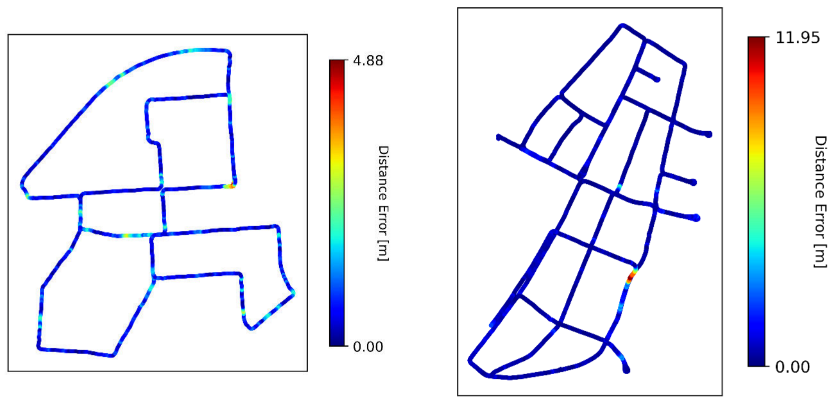
shows better translation accuracy than the other methods. However, EKF + DER obtain the best result in terms of maximum rotation error, that is relevant since we are considering a safety critical scenario. It can be seen in Figure 5.7 that in KITTI360 map there is a short path where each method struggles to obtain accurate results (warm colors). In that case, we observe the desired property of ensembles, *i.e.,* the mitigation of the overall prediction error, and the proposed EKF + DE is able to lower translation error. However, we argue that achieving these results with EKF + DER is quite impressive, since we utilize a single model without expensive sampling and both the model hyperameters and filter parameters were not tuned on KITTI360 dataset. Furthermore, the proposed EKF implementation represents a very simple localization system that, nevertheless, achieve good results. With the integration of additional GNSS measures and the employment of a more realistic motion model, it is likely that a filtering algorithm would further improve the localization accuracy, also with the direct uncertainty estimation method proposed with CMRNet + DER.



(a) Monte Carlo Dropout



(b) Deep Ensemble



(c) Deep Evidential Regression

Figure 5.7: Heatmaps representing the localization results of the proposed EKF implementations integrating the proposed uncertainty-aware CMRNet as the observation model. We evaluated each implementation on the validation runs of KITTI (left) and KITTI360 (right) datasets.

5.4 FINAL DISCUSSION

In this chapter, an application of state-of-the-art methods for uncertainty estimation in a multi-modal DNN for camera localization was proposed. In particular, we considered two sampling-based methods, *i.e.,* MCD and DE [54, 106], and a direct uncertainty estimation approach named DER [3]. To evaluate these methods, we proposed to integrate them within CMRNet [28], which performs map-agnostic camera localization by matching a camera observation with a LiDAR map. The experiments performed on KITTI, KITTI360 and Argoverse2 datasets evaluated the localization accuracy and uncertainty calibration, by also assessing the direct proportion between the increase in accuracy and the decrease in the estimated uncertainty. Moreover, we further evaluate the different uncertainty-aware models by integrating them within an EKF with the final aim of providing a realistic application scenario. Although CMRNet + MCD showed good localization accuracy and uncertainty calibration, it cannot guarantee that in presence of large uncertainty, we also obtain large errors. However, we could observe that the calibrated uncertainty provided by MCD leads to the desired effect when used within the proposed EKF. CMRNet + DE also achieves very accurate localization and well-calibrated uncertainty, resulting in an overall improvement in localization accuracy when a lower variance of the error distribution is experienced. Due to the ability of Deep Ensembles in increasing the overall accuracy of a DNN model, CMRNet + DE obtains very low localization error when employed in our EKF also when in presence of challenging scenes when the filter estimates start to drift. Finally, without undermining its original localization accuracy, we applied a DER-based approach to CMRNet showing the ability to provide well-calibrated uncertainties that can be also employed to detect localization failures using a one-shot estimation scheme.Moreover, we demonstrated the competitive results obtained by EKF + DER. This represents a huge improvement considering that we employed only one model that directly provides uncertainty pose estimates. To the best of our knowledge, this is the first work that integrates a DER-based approach in a DNN for camera pose regression.

Please note that we plan to submit the approach proposed in this chapter to a journal in the following weeks.

6 CONCLUSIONS

This thesis presented different methods that exploit Deep Neural Networks (DNNs) to perform vehicle localization in urban areas. In particular, the primary focus was to enhance localization deep learning techniques with the aim of improving their reliability, as they are deployed in safety-critical scenarios. The first part of this thesis addressed the entire localization pipeline by tackling the problems of both global and local localization. In particular, the proposed Light Detection And Ranging (LiDAR)-based approach utilizes a 3D DNN to perform the tasks of loop closure detection and point cloud registration, and it is finally deployed within a SLAM system. The decision to use a LIDAR-based approach is due to the general robustness provided by LIDAR sensors, which makes them a solid choice considering the demanding standards of autonomous driving. Although the previous model achieved impressive results, particularly in the presence of reverse loops, it does not explicitly provide a confidence measure for its predictions, making it more challenging to detect possible failures. To address this typical issue affecting deep learning methods and also DNN for localization, the second part of this thesis was centered on the topic of uncertainty quantification within neural networks. Specifically, the main focus was epistemic uncertainty, which refers to model uncertainty. To address this issue in the entire localization pipeline, two different uncertainty-aware approaches were proposed: a global localization method and a local refinement technique.

The three main contributions of this thesis work can be summarized as follows:

- a 3D DNN named LCDNet was introduced to face the problems of LiDAR-based loop closure detection and point cloud registration. The proposed technique achieved outstanding results especially in the presence of reverse loops, by outperforming other state-of-the-art methods. Furthermore, LCDNet demonstrated its effectiveness in a real-case application, by integrating it within an existing Simultaneouos Localization And Mapping (SLAM) system. Finally, the proposed method shows great generalization capability by performing accurate localization on a self-collected dataset;
- an exploratory study was proposed regarding the usage of a sampling-based technique named Deep Ensembles within the problem of LiDAR-based place recognition. In particular, the

proposed method faced the task of sampling feature-wise uncertainty by exploiting a knowledge distillation training strategy aiming to define comparable feature spaces between ensemble members. From the evaluation performed, we observed the ability of the proposed method to detect global localization failures, especially in challenging environments;

- finally, starting from an existing multi-modal localization refinement Convolutional Neural Network (CNN), three different uncertainty estimation methods were integrated within such a model, *i.e.,* Monte Carlo Dropout (MCD), Deep Ensemble (DE), and Deep Evidential Regression (DER). In particular, a notable contribution was the adaptation of DER, which required different changes to the original CNN architecture and to the training procedure. Moreover, DER demonstrated to obtain well-calibrated uncertainty estimates and generally achieved comparable results to the other methods when integrated within an Extended Kalman Filter (EKF). This result is quite impressive considering that we only employ a single model that does not perform any uncertainty sampling.

BIBLIOGRAPHY

[1] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti. "Point Clouds Registration with Probabilistic Data Association." In: *Int. Conf. on Intelligent Robots and Systems*. 2016, pp. 4092–4098.

[2] Yasin Almalioglu, Muhamad Risqi U. Saputra, Pedro P. B. de Gusmão, Andrew Markham, and Niki Trigoni. "GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks." In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5474–5480. DOI: 10.1109/ICRA.2019.8793512.

[3] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. "Deep Evidential Regression." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14927–14937.

[4] Mikaela Angelina Uy and Gim Hee Lee. "PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2018.

[5] Anastasios N. Angelopoulos and Stephen Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. 2022. arXiv: 2107.07511 [cs.LG].

[6] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. "Uncertainty sets for image classifiers using conformal prediction." In: *arXiv preprint arXiv:2009.14193* (2020).

[7] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. "PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2019.

[8] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2016.

[9] Relja Arandjelovic and Andrew Zisserman. "All about VLAD." In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2013, pp. 1578–1585.

[10] *Argo AI, driverless startup backed by Ford and VW, is shutting down.* https://www.theverge.com/2022/10/26/23423998/argo-ai-shut-down-ford-vw-av-self-driving. Accessed: January 2024.

[11] Romuald Aufrère, Jay Gowdy, Christoph Mertz, Chuck Thorpe, Chieh-Chih Wang, and Teruko Yata. "Perception for collision avoidance and autonomous driving." In: *Mechatronics* 13.10 (2003), pp. 1149–1161.

[12] T. Bailey, E. M. Nebot, J. K. Rosenblatt, and H. F. Durrant-Whyte. "Data association for mobile robot navigation: a graph theoretic approach." In: *Int. Conf. on Robotics & Automation*. 2000, pp. 2512–2517.

[13] Xicheng Ban, Hongjian Wang, Tao Chen, Ying Wang, and Yao Xiao. "Monocular Visual Odometry Based on Depth and Optical Flow Using Deep Learning." In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–19. DOI: 10.1109/TIM.2020.3024011.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In: *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer. 2006, pp. 404–417.

[15] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences." In: *Int. Conf. on Computer Vision*. 2019.

[16] Federico Bianchi, Amanda Cercas Curry, and Dirk Hovy. "Artificial Intelligence Accidents Waiting to Happen?" In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 193–199.

[17] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf.

[18] M. Bosse and R. Zlot. "Place recognition using keypoint voting in large 3D lidar datasets." In: *Int. Conf. on Robotics & Automation*. 2013.

[19] Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, C Guarino Lo Bianco, and Aurelio Piazzi. "The ARGO autonomous vehicle's vision and control systems." In: *International Journal of Intelligent Control and Systems* 3.4 (1999), pp. 409–441.

[20] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[21] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer, 2009.

[22] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K. Varshney, and Dawn Song. "Anomalous Example Detection in Deep Learning: A Survey." In: *IEEE Access* 8 (2020), pp. 132330–132347. DOI: 10.1109/ACCESS.2020.3010274.

[23] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuScenes: A Multimodal Dataset for Autonomous Driving." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[24] Kaiwen Cai, Chris Xiaoxuan Lu, and Xiaowei Huang. "STUN: Self-Teaching Uncertainty Estimation for Place Recognition." In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 6614–6621. DOI: 10.1109/IROS47612.2022.9981546.

[25] Anh-Quan Cao, Gilles Puy, Alexandre Boulch, and Renaud Marlet. "PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds." In: *Int. Conf. on Computer Vision*. 2021.

[26] Qiong Cao, Yiming Ying, and Peng Li. "Similarity Metric Learning for Face Recognition." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2013.

[27] Tim Caselitz, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. "Monocular camera localization in 3D LiDAR maps." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1926–1931. DOI: 10.1109/IROS.2016.7759304.

[28] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard. "CMRNet: Camera to LiDAR-Map Registration." In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 1283–1289. DOI: 10.1109/ITSC.2019.8917470.

[29] D. Cattaneo, M. Vaghi, S. Fontana, A. L. Ballardini, and D. G. Sorrenti. "Global visual localization in LiDAR-maps through shared 2D-3D embedding space." In: *Int. Conf. on Robotics & Automation*. 2020, pp. 4365–4371.

[30] D. Cattaneo, M. Vaghi, S. Fontana, A. L. Ballardini, and D. G. Sorrenti. "Global visual localization in LiDAR-maps through shared 2D-3D embedding space." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 4365–4371. DOI: 10.1109/ICRA40945.2020.9196859.

[31] Daniele Cattaneo, Domenico Giorgio Sorrenti, and Abhinav Valada. "CMRNet++: Map and Camera Agnostic Monocular Visual Localization in LiDAR Maps." In: *Int. Conf. on Robotics & Automation Workshop on Emerging Learning and Algorithmic Methods for Data Association in Robotics* (2020).

[32] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. "LCD-Net: Deep Loop Closure Detection and Point Cloud Registration for LiDAR SLAM." In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2074–2093. DOI: 10.1109/TRO.2022.3150683.

[33] Min Young Chang, Suyong Yeon, Soohyun Ryu, and Donghwan Lee. "SpoxelNet: Spherical Voxel-based Deep Place Recognition for 3D Point Clouds of Crowded Indoor Spaces." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 8564–8570. DOI: 10.1109/IROS45743.2020.9341549.

[34] Kuangyi Chen, Huai Yu, Wen Yang, Lei Yu, Sebastian Scherer, and Gui-Song Xia. "I2D-Loc: Camera localization via image to LiDAR depth flow." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 194 (2022), pp. 209–221.

[35] S. Y. Chen. "Kalman Filter for Robot Vision: A Survey." In: *IEEE Transactions on Industrial Electronics* 59.11 (2012), pp. 4409–4420. DOI: 10.1109/TIE.2011.2162714.

[36] X. Chen, T. Läbe, A. Milioto, T. Röhling, J. Behley, and C. Stachniss. "OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization." In: *Autonomous Robots* (2021). ISSN: 1573-7527. DOI: 10.1007/s10514-021-09999-0.

[37] Ran Cheng, Christopher Agia, Yuan Ren, Xinhai Li, and Liu Bingbing. "S3CNet: A Sparse Semantic Scene Completion Network for LiDAR Point Clouds." In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Re-

search. PMLR, 2021, pp. 2148–2161. URL: https://proceedings.mlr.press/v155/cheng21a.html.

[38] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. “Scaling algorithms for unbalanced optimal transport problems.” In: *Mathematics of Computation* 87.314 (2018), pp. 2563–2609.

[39] Christopher Choy, Wei Dong, and Vladlen Koltun. “Deep Global Registration.” In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2020.

[40] Mark Cummins and Paul Newman. “FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance.” In: *Int. Journal of Robotics Research* 27.6 (2008), pp. 647–665.

[41] Arthur P Dempster. “A generalization of Bayesian inference.” In: *Journal of the Royal Statistical Society: Series B (Methodological)* 30.2 (1968), pp. 205–232.

[42] Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. “Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation.” In: *International Journal of Computer Vision* (2022), pp. 1–28.

[43] John Denker and Yann LeCun. “Transforming Neural-Net Output Levels to Probability Distributions.” In: *Advances in Neural Information Processing Systems*. Ed. by R.P. Lippmann, J. Moody, and D. Touretzky. Vol. 3. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper_files/paper/1990/file/7eacb532570ff6858afd2723755ff790-Paper.pdf.

[44] Marvin Eisenberger, Aysim Toker, Laura Leal-Taixé, and Daniel Cremers. “Deep Shells: Unsupervised Shape Correspondence with Optimal Transport.” In: *Advances in Neural Information Processing Systems* 33 (2020).

[45] Kilian Fatras, Thibault Séjourné, Rémi Flamary, and Nicolas Courty. “Unbalanced minibatch optimal transport; applications to domain adaptation.” In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3186–3197.

[46] M. Feng, S. Hu, M. H. Ang, and G. H. Lee. “2D3D-Matchnet: Learning To Match Keypoints Across 2D Image And 3D Point Cloud.” In: *Int. Conf. on Robotics & Automation*. 2019, pp. 4790–4796.

[47] Mengdan Feng, Sixing Hu, Marcelo H Ang, and Gim Hee Lee. “2D3D-Matchnet: Learning To Match Keypoints Across 2D Image And 3D Point Cloud.” In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4790–4796. DOI: 10.1109/ICRA.2019.8794415.

[48] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[49] Simone Fontana, Daniele Cattaneo, Augusto L. Ballardini, Matteo Vaghi, and Domenico G. Sorrenti. "A benchmark for point clouds registration algorithms." In: *Robotics and Autonomous Systems* 140 (2021), p. 103734. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2021.103734. URL: https://www.sciencedirect.com/science/article/pii/S0921889021000191.

[50] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. "Deep ensembles: A loss landscape perspective." In: *arXiv preprint arXiv:1912.02757* (2019).

[51] *GM-owned Cruise admits failures in driverless car accident*. https://www.bbc.com/news/business-68101907. Accessed: January 2024.

[52] Yarin Gal. "Uncertainty in Deep Learning." PhD thesis. University of Cambridge, 2016.

[53] Yarin Gal and Zoubin Ghahramani. "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference." In: *ICLR workshop track*. 2016.

[54] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 1050–1059. URL: https://proceedings.mlr.press/v48/gal16.html.

[55] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2012.

[56] Ross Girshick. "Fast R-CNN." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[57] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation With Left-Right Consistency." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[58] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. "Digging Into Self-Supervised Monocular Depth Estimation." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[59] Teofilo F Gonzalez. “Clustering to minimize the maximum intercluster distance.” In: *Theoretical computer science* 38 (1985), pp. 293–306.

[60] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. “Learning rich features from RGB-D images for object detection and segmentation.” In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer. 2014, pp. 345–360.

[61] Mordechai Haklay and Patrick Weber. “OpenStreetMap: User-Generated Street Maps.” In: *IEEE Pervasive Computing* 7.4 (2008), pp. 12–18. DOI: 10.1109/MPRV.2008.80.

[62] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. “Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14141–14152.

[63] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M. Dai, and Dustin Tran. *Training independent subnetworks for robust prediction*. 2021. arXiv: 2010.06610 [cs.LG].

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

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[66] L. He, X. Wang, and H. Zhang. “M2DP: A novel 3D point cloud descriptor and its application in loop closure detection.” In: *Int. Conf. on Intelligent Robots and Systems*. 2016, pp. 231–237.

[67] Ming He, Chaozheng Zhu, Qian Huang, Baosen Ren, and Jintao Liu. “A review of monocular visual odometry.” In: *The Visual Computer* 36.5 (2020), pp. 1053–1065.

[68] Alexander Hermans, Lucas Beyer, and Bastian Leibe. “In defense of the triplet loss for person re-identification.” In: *arXiv preprint arXiv:1703.07737* (2017).

[69] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. “Real-time loop closure in 2D LIDAR SLAM.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278. DOI: 10.1109/ICRA.2016.7487258.

[70] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." In: *Autonomous robots* 34 (2013), pp. 189–206.

[71] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. "RandLA-Net: Efficient semantic segmentation of large-scale point clouds." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117.

[72] Leyao Huang. "Review on LiDAR-based SLAM techniques." In: *2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)*. IEEE. 2021, pp. 163–168.

[73] Eyke Hüllermeier and Willem Waegeman. "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods." In: *Machine Learning* 110 (2021), pp. 457–506.

[74] Rubén Izquierdo, Álvaro Quintanar, David Fernández Llorca, Iván García Daza, Noelia Hernández, Ignacio Parra, and Miguel Ángel Sotelo. "Vehicle trajectory prediction on highways using bird eye view representations and deep learning." In: *Applied Intelligence* 53.7 (2023), pp. 8370–8388.

[75] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. "Aggregating Local Descriptors into a Compact Image Representation." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2010, pp. 3304–3311.

[76] J. Johnson, M. Douze, and H. Jégou. "Billion-scale similarity search with GPUs." In: *IEEE Transactions on Big Data* (2019).

[77] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. "Aggregating local descriptors into a compact image representation." In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3304–3311. DOI: 10.1109/CVPR.2010.5540039.

[78] Kimmo Karkkainen and Jungseock Joo. "FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age for Bias Measurement and Mitigation." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1548–1558.

[79] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, and Daniel Toal. "A survey of state-of-the-art on visual SLAM." In: *Expert Systems with Applications* 205 (2022), p. 117734.

[80] Alex Kendall and Roberto Cipolla. “Modelling uncertainty in deep learning for camera relocalization.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4762–4769. DOI: 10.1109/ICRA.2016.7487679.

[81] Alex Kendall and Roberto Cipolla. “Modelling uncertainty in deep learning for camera relocalization.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4762–4769. DOI: 10.1109/ICRA.2016.7487679.

[82] Alex Kendall and Roberto Cipolla. “Geometric Loss Functions for Camera Pose Regression With Deep Learning.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[83] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[84] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[85] Alex Kendall, Matthew Grimes, and Roberto Cipolla. “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[86] Giseop Kim, Sunwook Choi, and Ayoung Kim. “Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments.” In: *IEEE Transactions on Robotics* (2021), pp. 1–19. DOI: 10.1109/TRO.2021.3116424.

[87] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. “MulRan: Multimodal Range Dataset for Urban Place Recognition.” In: *Int. Conf. on Robotics & Automation*. 2020.

[88] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. “MulRan: Multimodal Range Dataset for Urban Place Recognition.” In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 6246–6253. DOI: 10.1109/ICRA40945.2020.9197298.

[89] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes.” In: (2022). arXiv: 1312.6114 [stat.ML].

[90] Durk P Kingma, Tim Salimans, and Max Welling. "Variational Dropout and the Local Reparameterization Trick." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/bc7316929fe1545bf0b98d114ee3ecb8-Paper.pdf.

[91] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural Safety* 31.2 (2009). Risk Acceptance and Risk Communication, pp. 105–112. ISSN: 0167-4730. DOI: https://doi.org/10.1016/j.strusafe.2008.06.020. URL: https://www.sciencedirect.com/science/article/pii/S0167473008000556.

[92] Joshua Knights, Peyman Moghadam, Milad Ramezani, Sridha Sridharan, and Clinton Fookes. "Incloud: Incremental learning for point cloud place recognition." In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 8559–8566.

[93] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. "Hough Transform and 3D SURF for Robust Three Dimensional Classification." In: *Europ. Conf. on Computer Vision*. 2010, pp. 589–602.

[94] Roger Koenker and Gilbert Bassett Jr. "Regression quantiles." In: *Econometrica: journal of the Econometric Society* (1978), pp. 33–50.

[95] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. "Style Transfer by Relaxed Optimal Transport and Self-Similarity." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2019, pp. 10043–10052.

[96] Jacek Komorowski. "MinkLoc3D: Point Cloud Based Large-Scale Place Recognition." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1790–1799.

[97] Jacek Komorowski. "Improving Point Cloud Based Place Recognition with Ranking-based Loss and Large Batch Training." In: *2022 26th International Conference on Pattern Recognition (ICPR)*. 2022, pp. 3699–3705. DOI: 10.1109/ICPR56361.2022.9956458.

[98] Jacek Komorowski, Monika Wysoczanska, and Tomasz Trzcinski. "EgoNN: Egocentric Neural Network for Point Cloud Based 6DoF Relocalization at the City Scale." In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 722–729. DOI: 10.1109/LRA.2021.3133593.

[99] Xin Kong, Xuemeng Yang, Guangyao Zhai, Xiangrui Zhao, Xianfang Zeng, Mengmeng Wang, Yong Liu, Wanlong Li, and Feng Wen. "Semantic Graph Based Place Recognition for 3D Point Clouds." In: *Int. Conf. on Intelligent Robots and Systems* (2020).

[100] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[101] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. "The DARPA Robotics Challenge Finals: Results and Perspectives." In: *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Ed. by Matthew Spenko, Stephen Buerger, and Karl Iagnemma. Cham: Springer International Publishing, 2018, pp. 1–26. ISBN: 978-3-319-74666-1. DOI: `10.1007/978-3-319-74666-1_1`. URL: `https://doi.org/10.1007/978-3-319-74666-1_1`.

[102] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. "Accurate Uncertainties for Deep Learning Using Calibrated Regression." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2796–2804. URL: `https://proceedings.mlr.press/v80/kuleshov18a.html`.

[103] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. "A Survey on Deep Learning Techniques for Stereo-Based Depth Estimation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 1738–1764. DOI: `10.1109/TPAMI.2020.3032602`.

[104] Pierre-Yves Lajoie and Giovanni Beltrame. "Self-Supervised Domain Calibration and Uncertainty Estimation for Place Recognition." In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 792–799. DOI: `10.1109/LRA.2022.3232033`.

[105] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[106] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[107] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. "PointPillars: Fast Encoders for Object Detection From Point Clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[108] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[109] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "EP n P: An accurate O (n) solution to the P n P problem." In: *International journal of computer vision* 81 (2009), pp. 155–166.

[110] Jiaxin Li and Gim Hee Lee. "DeepI2P: Image-to-Point Cloud Registration via Deep Classification." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15960–15969.

[111] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. "UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7286–7291. DOI: 10.1109/ICRA.2018.8461251.

[112] Yiyi Liao, Jun Xie, and Andreas Geiger. "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2023), pp. 3292–3310. DOI: 10.1109/TPAMI.2022.3179507.

[113] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[114] Qiang Liu and Fuhai Duan. "Loop closure detection using CNN words." In: *Intelligent Service Robotics* 12.4 (2019), pp. 303–318.

[115] Rong Liu, Jinling Wang, and Bingqi Zhang. “High Definition Map for Automated Driving: Overview and Analysis.” In: *Journal of Navigation* 73.2 (2020), 324–341. DOI: 10.1017/S0373463319000638.

[116] David G Lowe. “Distinctive image features from scale-invariant keypoints.” In: *International journal of computer vision* 60 (2004), pp. 91–110.

[117] Fan Lu, Guang Chen, Yinlong Liu, Lijun Zhang, Sanqing Qu, Shu Liu, and Rongqi Gu. “HRegNet: A Hierarchical Network for Large-scale Outdoor LiDAR Point Cloud Registration.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16014–16023.

[118] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. “DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[119] Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision.” In: *International Joint Conference on Artificial Intelligence*. 1981, 674–679.

[120] Junyi Ma, Jun Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. “OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition.” In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 6958–6965. DOI: 10.1109/LRA.2022.3178797.

[121] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. “1 Year, 1000km: The Oxford RobotCar Dataset.” In: *The International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498.

[122] Keita Mason, Joshua Knights, Milad Ramezani, Peyman Moghadam, and Dimity Miller. “Uncertainty-Aware Lidar Place Recognition in Novel Environments.” In: *arXiv preprint arXiv:2210.01361v1* (2022).

[123] Donald Meagher. “Geometric modeling using octree encoding.” In: *Computer graphics and image processing* 19.2 (1982), pp. 129–147.

[124] Nis Meinert and Alexander Lavin. “Multivariate Deep Evidential Regression.” In: *CoRR* abs/2104.06135 (2021). arXiv: 2104.06135. URL: https://arxiv.org/abs/2104.06135.

[125] Antoine Miech, Ivan Laptev, and Josef Sivic. “Learnable pooling with Context Gating for video classification.” In: *arXiv preprint arXiv:1706.06905* (2017).

[126] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. "Rangenet++: Fast and accurate lidar semantic segmentation." In: *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2019, pp. 4213–4220.

[127] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. "CoordiNet: Uncertainty-Aware Pose Regressor for Reliable Vehicle Localization." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2229–2238.

[128] 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* 31.5 (2015), pp. 1147–1163.

[129] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[130] Peer Neubert, Stefan Schubert, and Peter Protzel. "Sampling-based methods for visual navigation in 3D maps by synthesizing depth images." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2492–2498. DOI: 10.1109/IROS.2017.8206067.

[131] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[132] Valentin Peretroukhin, Matthew Giamou, David M. Rosen, W. Nicholas Greene, Nicholas Roy, and Jonathan Kelly. "A Smooth Representation of SO(3) for Deep Rotation Learning with Uncertainty." In: *Robotics: Science and Systems*. 2020.

[133] Valentin Peretroukhin, Brandon Wagstaff, Matthew Giamou, and Jonathan Kelly. "Probabilistic Regression of Rotations using Quaternion Averaging and a Deep Multi-Headed Network." In: *CoRR* abs/1904.03182 (2019). arXiv: 1904.03182. URL: http://arxiv.org/abs/1904.03182.

[134] Kürsat Petek, Kshitij Sirohi, Daniel Büscher, and Wolfram Burgard. "Robust Monocular Localization in Sparse HD Maps Leveraging Multi-Task Uncertainty Estimation." In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4163–4169. DOI: 10.1109/ICRA46639.2022.9812266.

[135] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. "Comparing ICP variants on real-world data sets." In: *Autonomous Robots* 34.3 (2013), pp. 133–148.

[136] Gowdham Prabhakar, Binsu Kailath, Sudha Natarajan, and Rajesh Kumar. "Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving." In: *2017 IEEE Region 10 Symposium (TENSYMP)*. 2017, pp. 1–6. DOI: 10.1109/TENCONSpring.2017.8069972.

[137] Gilles Puy, Alexandre Boulch, and Renaud Marlet. "FLOT: Scene Flow on Point Clouds Guided by Optimal Transport." In: *Europ. Conf. on Computer Vision*. 2020, pp. 527–544.

[138] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *IEEE Conf. on Computer Vision and Pattern Recognition* (2017).

[139] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[140] Noha Radwan, Abhinav Valada, and Wolfram Burgard. "VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry." In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4407–4414. DOI: 10.1109/LRA.2018.2869640.

[141] Gabrielle Ras, Ning Xie, Marcel Van Gerven, and Derek Doran. "Explainable deep learning: A field guide for the uninitiated." In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 329–396.

[142] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[143] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "ORB: An efficient alternative to SIFT or SURF." In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[144] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge." In: *International journal of computer vision* 115 (2015), pp. 211–252.

[145] R. B. Rusu, N. Blodow, and M. Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration." In: *Int. Conf. on Robotics & Automation*. 2009, pp. 3212–3217.

[146] Güray SONUGÜR. "A Review of quadrotor UAV: Control and SLAM methodologies ranging from conventional to innovative approaches." In: *Robotics and Autonomous Systems* 161 (2023), p. 104342. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2022.104342. URL: https://www.sciencedirect.com/science/article/pii/S0921889022002317.

[147] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperGlue: Learning Feature Matching With Graph Neural Networks." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2020, pp. 4937–4946.

[148] Paul-Edouard Sarlin et al. "Back to the Feature: Learning Robust Camera Localization From Pixels To Pose." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3247–3257.

[149] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. "Improving image-based localization by active correspondence search." In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*. Springer. 2012, pp. 752–765.

[150] L. Schaupp, M. Bürki, R. Dubé, R. Siegwart, and C. Cadena. "OREOS: Oriented Recognition of 3D Point Clouds in Outdoor Scenarios." In: *Int. Conf. on Intelligent Robots and Systems*. 2019, pp. 3255–3261.

[151] Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. "RegNet: Multimodal sensor registration using deep neural networks." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1803–1810. DOI: 10.1109/IVS.2017.7995968.

[152] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2015.

[153] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. "Generalized-icp." In: *Robotics: Science and Systems*. 2009.

[154] Murat Sensoy, Lance Kaplan, and Melih Kandemir. "Evidential Deep Learning to Quantify Classification Uncertainty." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/a981f2b708044d6fb4a71a1463242520-Paper.pdf.

[155] Jacopo Serafin, Edwin Olson, and Giorgio Grisetti. “Fast and robust 3D feature extraction from sparse point clouds.” In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4105–4112. DOI: 10.1109/IROS.2016.7759604.

[156] Tixiao Shan and Brendan Englot. “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain.” In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4758–4765. DOI: 10.1109/IROS.2018.8594299.

[157] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Rus Daniela. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping.” In: *Int. Conf. on Intelligent Robots and Systems*. 2020, pp. 5135–5142.

[158] Liushuai Shi, Le Wang, Chengjiang Long, Sanping Zhou, Mo Zhou, Zhenxing Niu, and Gang Hua. “SGCN: Sparse Graph Convolution Network for Pedestrian Trajectory Prediction.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8994–9003.

[159] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. “PV-RCNN: Point-voxel feature set abstraction for 3d object detection.” In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2020, pp. 10529–10538.

[160] Ashutosh Singandhupe and Hung Manh La. “A review of slam techniques and security in autonomous driving.” In: *2019 third IEEE international conference on robotic computing (IRC)*. IEEE. 2019, pp. 602–607.

[161] Richard Sinkhorn. “A relationship between arbitrary positive matrices and doubly stochastic matrices.” In: *The annals of mathematical statistics* 35.2 (1964), pp. 876–879.

[162] Sriram Siva, Zachary Nahman, and Hao Zhang. “Voxel-Based Representation Learning for Place Recognition Based on 3D Point Clouds.” In: *Int. Conf. on Intelligent Robots and Systems*. 2020.

[163] Xiaolin Song, Kaili Zhao, Wen-Sheng Chu, Honggang Zhang, and Jun Guo. “Progressive refinement network for occluded pedestrian detection.” In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer. 2020, pp. 32–48.

[164] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[165] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. “Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation.” In: *Int. Conf. on Intelligent Robots and Systems*. 2011, pp. 1249–1255.

[166] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. “RSL-Net: Localising in Satellite Images From a Radar on the Ground.” In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1087–1094. DOI: 10.1109/LRA.2020.2965907.

[167] *The false promises of Tesla's Full Self-Driving*. https://www.theverge.com/2023/8/23/23837598/tesla-elon-musk-self-driving-false-promises-land-of-the-giants. Accessed: January 2024.

[168] Frank Thomanek and Ernst D Dickmanns. “Autonomous road vehicle guidance in normal traffic.” In: *Recent Developments in Computer Vision: Second Asian Conference on Computer Vision, ACCV'95 Singapore, December 5–8, 1995 Invited Session Papers 2*. Springer. 1996, pp. 499–507.

[169] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN: 0262201623 9780262201629. URL: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance.

[170] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. “Stanley: The robot that won the DARPA Grand Challenge.” In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[171] Akihiko Torii, Hajime Taira, Josef Sivic, Marc Pollefeys, Masatoshi Okutomi, Tomas Pajdla, and Torsten Sattler. “Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.3 (2021), pp. 814–829. DOI: 10.1109/TPAMI.2019.2941876.

[172] Inam Ullah, Yu Shen, Xin Su, Christian Esposito, and Chang Choi. “A Localization Based on Unscented Kalman Filter and Particle Filter Localization Algorithms.” In: *IEEE Access* 8 (2020), pp. 2233–2246. DOI: 10.1109/ACCESS.2019.2961740.

[173] Matteo Vaghi, Augusto Luis Ballardini, Simone Fontana, and Domenico Giorgio Sorrenti. “Uncertainty-Aware DNN for Multi-Modal Camera Localization.” In: *arXiv preprint arXiv:2211.01234v2* (2023).

[174] Matteo Vaghi, Fabio D’Elia, Augusto Luis Ballardini, and Domenico Giorgio Sorrenti. “Understanding the Effect of Deep Ensembles in LiDAR-Based Place Recognition.” In: *AIxIA 2023 – Advances in Artificial Intelligence*. Ed. by Roberto Basili, Domenico Lembo, Carla Limongelli, and Andrea Orlandini. Cham: Springer Nature Switzerland, 2023, pp. 295–309. ISBN: 978-3-031-47546-7.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. “Attention is All you Need.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[176] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. “Continual SLAM: Beyond Lifelong Simultaneous Localization and Mapping Through Continual Learning.” In: *Robotics Research*. Ed. by Aude Billard, Tamim Asfour, and Oussama Khatib. Cham: Springer Nature Switzerland, 2023, pp. 19–35.

[177] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Vol. 29. Springer, 2005.

[178] H. Wang, C. Wang, and L. Xie. “Intensity Scan Context: Coding Intensity and Geometry Relations for Loop Closure Detection.” In: *Int. Conf. on Robotics & Automation*. 2020, pp. 2095–2101.

[179] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. “DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks.” In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2043–2050. DOI: 10.1109/ICRA.2017.7989236.

[180] Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Ronald Clark, Bo Yang, Andrew Markham, and Niki Trigoni. “PointLoc: Deep Pose Regressor for LiDAR Point Cloud Localization.” In: *IEEE Sensors Journal* 22.1 (2022), pp. 959–968. DOI: 10.1109/JSEN.2021.3128683.

[181] Ying Wang, Zezhou Sun, Jian Yang, and Hui Kong. “LiDAR Iris for Loop-Closure Detection.” In: *Int. Conf. on Intelligent Robots and Systems*. 2020.

[182] Yue Wang and Justin M Solomon. "Deep closest point: Learning representations for point cloud registration." In: *Int. Conf. on Computer Vision*. 2019, pp. 3523–3532.

[183] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds." In: *ACM Trans. Graph.* 38.5 (2019).

[184] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. "Dynamic graph cnn for learning on point clouds." In: *ACM Transactions on Graphics (tog)* 38.5 (2019), pp. 1–12.

[185] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/488e4104520c6aab692863cc1dba45af-Paper.pdf.

[186] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. "A comparison of loop closing techniques in monocular SLAM." In: *Robotics and Autonomous Systems* 57.12 (2009). Inside Data Association, pp. 1188–1197. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2009.06.010. URL: https://www.sciencedirect.com/science/article/pii/S0921889009000876.

[187] Benjamin Wilson et al. "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting." In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. Curran, 2021.

[188] Ryan W. Wolcott and Ryan M. Eustice. "Visual localization within LIDAR maps for automated urban driving." In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 176–183. DOI: 10.1109/IROS.2014.6942558.

[189] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. "Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2016.

[190] H. Yang, J. Shi, and L. Carlone. "TEASER: Fast and Certifiable Point Cloud Registration." In: *IEEE Transactions on Robotics* (2020).

[191] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. “Go-ICP: A globally optimal solution to 3D ICP point-set registration.” In: *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2015), pp. 2241–2254.

[192] Zi Jian Yew and Gim Hee Lee. “RPM-Net: Robust Point Matching Using Learned Features.” In: *IEEE Conf. on Computer Vision and Pattern Recognition*. 2020, pp. 11824–11833.

[193] Huan Yin, Xuecheng Xu, Yue Wang, and Rong Xiong. “Radar-to-lidar: Heterogeneous place recognition via joint learning.” In: *Frontiers in Robotics and AI* 8 (2021), p. 661199.

[194] Zhichao Yin and Jianping Shi. “GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[195] Amir Roshan Zamir and Mubarak Shah. “Accurate image localization based on google maps street view.” In: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer. 2010, pp. 255–268.

[196] Enrica Zereik, Marco Bibuli, Nikola Mišković, Pere Ridao, and António Pascoal. “Challenges and future trends in marine robotics.” In: *Annual Reviews in Control* 46 (2018), pp. 350–368.

[197] Wenxiao Zhang and Chunxia Xiao. “PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[198] X. Zhang, Y. Su, and X. Zhu. “Loop closure detection for visual SLAM systems using convolutional neural network.” In: *International Conference on Automation and Computing*. 2017, pp. 1–6.

[199] Zhengyou Zhang. “Iterative point matching for registration of free-form curves and surfaces.” In: *Int. Journal of Computer Vision* 13.2 (1994), pp. 119–152.

[200] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. “Object Detection With Deep Learning: A Review.” In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[201] Y. Zhong. “Intrinsic shape signatures: A shape descriptor for 3D object recognition.” In: *Int. Conf. on Computer Vision Workshops*. 2009, pp. 689–696.

[202] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Fast Global Registration." In: *Europ. Conf. on Computer Vision*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. 2016, pp. 766–782.

[203] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing." In: *arXiv:1801.09847* (2018).

[204] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.

[205] Yachen Zhu, Yanyang Ma, Long Chen, Cong Liu, Maosheng Ye, and Lingxi Li. "GOSMatch: Graph-of-Semantics Matching for Detecting Loop Closures in 3D LiDAR data." In: *Int. Conf. on Intelligent Robots and Systems*. 2020.

[206] Yingying Zhu, Jiong Wang, Lingxi Xie, and Liang Zheng. "Attention-Based Pyramid Aggregation Network for Visual Place Recognition." In: *Proceedings of the 26th ACM International Conference on Multimedia*. MM '18. Association for Computing Machinery, 2018, 99–107. ISBN: 9781450356657. DOI: 10.1145/3240508.3240525. URL: https://doi.org/10.1145/3240508.3240525.