Precise localization in 3D prior map for autonomous driving

Mohamed Lamine Tazir

UNIVERSITÉ CLERMONT AUVERGNE

ÉCOLE DOCTORALE SPI

Sciences Pour l'Ingénieur

T H È S E

pour l'obtention du grade de

Docteur d'Université

Spécialité : Electronique et Systèmes

présentée et soutenue par

Mohamed Lamine TAZIR

Precise localization in 3D prior map for autonomous driving

préparée à l'INSTITUT PASCAL, Équipe PERSYST

soutenue le 17 Décembre 2018

Jury :

Rapporteurs :	Frédéric LERASLE	–	PR – Université Paul Sabatier
	Sylvie TREUILLET	–	MCF-HDR – Université d'Orléans
Directeur :	Laurent TRASSOUDAINE	–	PR – Université Clermont Auvergne
Co-directeur :	Paul CHECCHIN	–	PR – Université Clermont Auvergne
Examinateurs :	Beatriz MARCOTEGUI	–	PR – MINES ParisTech
	Roland CHAPUIS	–	PR – Université Clermont Auvergne

Abstract:

The concept of self-driving vehicles is becoming a happening reality and will soon share our roads with other vehicles – autonomous or not-. For a self-driving car to move around in its environment in a securely, it needs to sense to its immediate environment and most importantly localize itself to be able to plan a safe trajectory to follow. Therefore, to perform tasks such as trajectory planning and navigation, a precise localization is of upmost importance. This would further allow the vehicle to constantly plan and predict an optimal path in order to weave through cluttered spaces by avoiding collisions with other agents sharing the same space as the latter. For years, the Global Positioning System (GPS) has been a widespread complementary solution for navigation. The latter allows only a limited precision (range of several meters). Although the Differential GPS and the Real Time Kinematic (RTK) systems have reached considerable accuracy, these systems remain sensitive to signal masking and multiple reflections, offering poor reliability in dense urban areas. All these deficiencies make these systems simply unsuitable to handle hard real time constraints such as collision avoidance. A prevailing alternative that has attracted interest recently, is to use upload a prior map in the system so that the agent can have a reliable support to lean on. Indeed, maps facilitate the navigation process and add an extra layer of security and other dimensions of semantic understanding. The vehicle uses its onboard sensors to compare what it perceives at a given instant to what is stored in the backend memory of the system. In this way, the autonomous vehicle can actually anticipate and predict its actions accordingly.

The purpose of this thesis is to develop tools allowing an accurate localization task in order to deal with some complex navigation tasks outlined above. Localization is mainly performed by matching a 3D prior map with incoming point cloud structures as the vehicle moves. Three main objectives are set out leading with two distinct phases deployed (the map building and the localization). The first allows the construction of the map, with centimeter accuracy using static or dynamic laser surveying technique. Explicit details about the experimental setup and data acquisition campaigns thoroughly carried out during the course of this work are given. The idea is to construct efficient maps liable to be updated in the long run so that the environment representation contained in the 3D models are compact and robust. Moreover, map-building invariant on any dedicated infrastructure is of the paramount importance of this work in order to rhyme with the concept of flexible mapping and localization. In order to build maps incrementally, we rely on a self-implementation of state of the art iterative closest point (ICP) algorithm, which is then upgraded with new variants and compared to other implemented versions available in the literature.

However, obtaining accurate maps requires very dense point clouds, which make them inefficient for real-time use. In this context, the second objective deals with points cloud reduction. The proposed approach is based on the use of both color information and the geometry of the scene. It aims to find sets of 3D points with the same color in a very small region and replacing each set with one point. As a result, the volume of the map will be significantly reduced, while the proprieties of this map such as the shape and color of scanned objects remain preserved.

The third objective resort to efficient, precise and reliable localization once the maps are built and treated. For this purpose, the online data should be accurate, fast with low computational effort whilst maintaining a coherent model of the explored space. To this end, the Velodyne HDL-32 comes into play. Ultimately, in order to localize sparse data acquired with the Velodyne, to the dense point cloud map of the high-resolution Leica P20 platform, a map-matching process is launched for each acquired frame of the Velodyne. This is where things get very handy since there are no directly correspondences between the two data and on top of two very different resolutions of each point cloud, the sparse to dense data configuration comes in to play. Over here, classical state of the art techniques fail vehemently leading to no positional output of the sensors. In order to address this issue, we propose a new strategy based on an intelligent point selection. We proceed with a voxelization and clustering of points based on the orientation of normals on both the sparse and dense points cloud. This process, which sits on classical ICP optimization, consists in matching points representing each local surface of the online frame with the corresponding local surfaces of the map cloud. This approach baptized as Cluster Iterative Point Cloud (CICP) proves to be an efficient viable solution to the problem of sparse to dense point cloud registration and map matching.

Keywords: ICP, registration, mapping, localization, prior map, sampling, selection, map-matching.

Résumé:

Les véhicules autonomes, qualifiés aussi de véhicules sans conducteur, deviennent dans certains contextes une réalité tangible et partageront très bientôt nos routes avec d'autres véhicules classiques. Pour qu'un véhicule autonome se déplace de manière sécurisée, il doit savoir où il se trouve et ce qui l'entoure dans l'environnement. Pour la première tâche, pour déterminer sa position dans l'environnement, il doit se localiser selon six degrés de liberté (position et angles de rotation). Alors que pour la deuxième tâche, une bonne connaissance de cet environnement « proche » est nécessaire, ce qui donne lieu à une solution sous forme de cartographie. Par conséquent, pour atteindre le niveau de sécurité souhaité des véhicules autonomes, une localisation précise est primordiale. Cette localisation précise permet au véhicule non seulement de se positionner avec précision, mais également de trouver sa trajectoire optimale et d'éviter efficacement les collisions avec des objets statiques et dynamiques sur son trajet. Actuellement, la solution la plus répandue est le système de positionnement (GPS). Ce système ne permet qu'une précision limitée (de l'ordre de plusieurs mètres) et bien que les systèmes RTK (*Real Time Kinematic*) et DGPS (*Differential GPS*) aient atteint une précision bien plus satisfaisante, ces systèmes restent sensibles au masquage des signaux, et aux réflexions multiples, en particulier dans les zones urbaines denses. Toutes ces déficiences rendent ces systèmes inadaptés pour traiter des tâches critiques telles que l'évitement des collisions.

Une alternative qui a récemment attiré l'attention des experts (chercheurs et industriels), consiste à utiliser une carte à priori pour localiser la voiture de l'intérieur de celui-ci. En effet, les cartes facilitent le processus de navigation et ajoutent une couche supplémentaire de sécurité et de compréhension. Le véhicule utilise ses capteurs embarqués pour comparer ce qu'il perçoit à un moment donné avec ce qui est stocké dans sa mémoire. Les cartes à priori permettent donc au véhicule de mieux se localiser dans son environnement en lui permettant de focaliser ses capteurs et la puissance de calcul uniquement sur les objets en mouvement. De cette façon, le véhicule peut prédire ce qui devrait arriver et voir ensuite ce qui se passe réellement en temps réel, et donc peut prendre une décision sur ce qu'il faut faire.

Cette thèse vise donc à développer des outils permettant une localisation précise d'un véhicule autonome dans un environnement connu à priori. Cette localisation est déterminée par appariement (*Map-matching*) entre une carte de l'environnement disponible a priori et les données collectées au fur et à mesure que le véhicule se déplace. Pour ce faire, deux phases distinctes sont déployées. La première permet la construction de la carte, avec une précision centimétrique en utilisant des techniques de construction de cartes statiques ou dynamiques. La seconde correspond à la capacité de localiser le véhicule dans cette carte 3D en l'absence d'infrastructures dédiées comprenant le système GPS, les mesures inertielles (IMU) ou des balises.

Au cours de ce travail, différentes techniques sont développées pour permettre la réalisation des deux phases mentionnées ci-dessus. Ainsi, la phase de construction de cartes, qui consiste à recaler des nuages de points capturés pour construire une représentation unique et unifiée de l'environnement, correspond au problème de la localisation et de la cartographie simultanée (SLAM). Afin de faire face à ce problème, nous avons testé et comparé différentes méthodes de recalage. Cependant, l'obtention de cartes précises nécessite des nuages de points très denses, ce qui les rend inefficaces pour une utilisation en temps réel. Dans ce contexte, une nouvelle méthode de réduction des points est proposée.

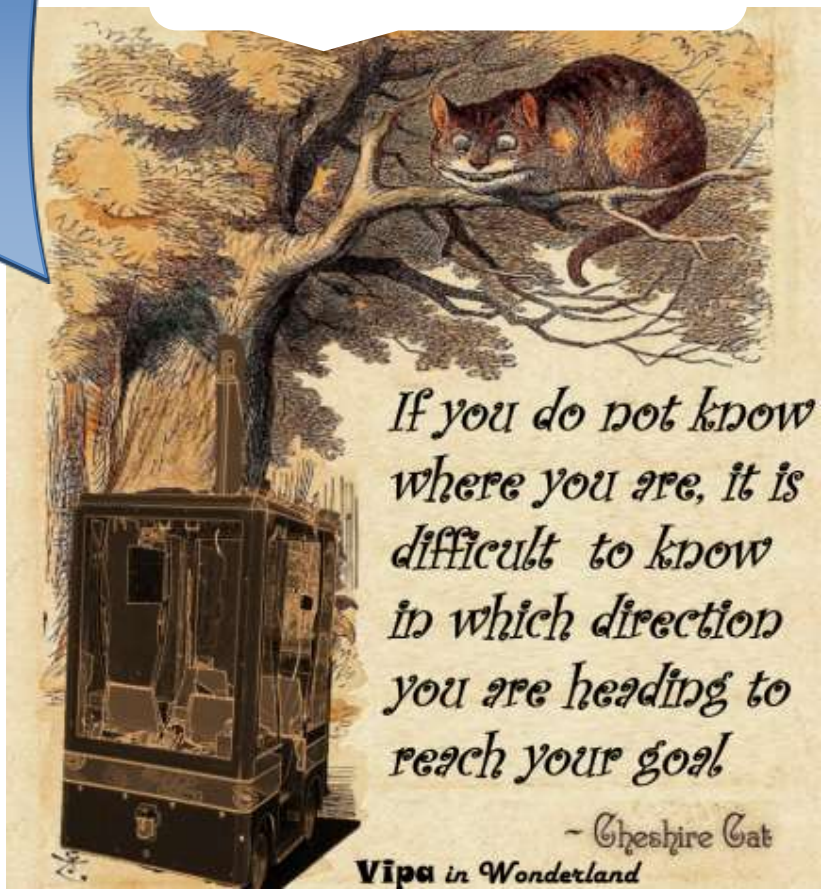
L'intérêt principal dans l'utilisation d'une carte à priori est d'utiliser seulement un algorithme de localisation pendant la phase de fonctionnement du véhicule, afin d'obtenir le plus haut degré de précision. Néanmoins, dans l'exemption des capteurs embarqués compatibles avec ceux utilisés dans l'élaboration de la carte à priori, il n'est pas possible de bénéficier de toutes les cartes produites avec le maximum de précision désirée. Afin de résoudre de manière satisfaisante ce problème de localisation, une stratégie de *« sparse to dense scan-matching »*, qui prend comme entrée des nuages de points de résolution différente, recueillies avec différents capteurs est proposé. Finalement, l'extension de ce principe est appliquée pour aboutir à une localisation robuste et précise pour les véhicules autonomes utilisant des données éparses produites à une fréquence de rafraîchissement élevée, et des cartes denses plus précises de l'environnement.

Mots-clés: ICP, recalage, cartographie, localisation, carte à priori, échantillonnage, sélection, appariement.

Contents

Chapter I

Introduction



CHAPTER 1

Introduction

Contents

The thesis subject that will be presented in this manuscript is part of a very current context and very followed by the scientific community, which is the localization of an autonomous vehicle. We will start by presenting generalities on autonomous driving, including information about the history and technology of autonomous vehicles. This will be followed by the main challenges faced the implementation of this technology and particularly the precise localization challenge.

1.1 Introduction

Transport is one of the main pillars of many human civilizations because of its importance to connect the different places of the world. That is why the human has sought innovation and discovery for thousands of years to develop and improve different means of transport. After the invention of the steam engine and then its development to work on oil and its derivatives, modern means of transport, such as cars, trains, ships, and planes, have emerged. However, if there is one mean that embodies this industrial revolution, it is the car, which has gradually become the main mode of transport of individuals and goods. Despite the presence of different shapes, types, and models, the vehicle in its overall design (steering wheel, pedal, speed) has changed very little for almost two centuries of its existence. Although it has become over time, faster, safer and more comfortable. In the early twentieth century, digital technology, essentially intangible, seems to replace the very concrete car as a new symbol of the development of civilizations. More interesting, the digital progressively seizing the cars introducing a new fundamental revolution, a revolution of the same order as that of the carriage to the automobile. This revolution is to make the car autonomous-that can move on its own without the help of a driver. This type of car in which one would move without even taking a glance at the road, where the brake, the gas pedal, the steering wheel and the gear-shift lever would otherwise have disappeared, to which it would suffice to indicate the destination to let itself then drive.

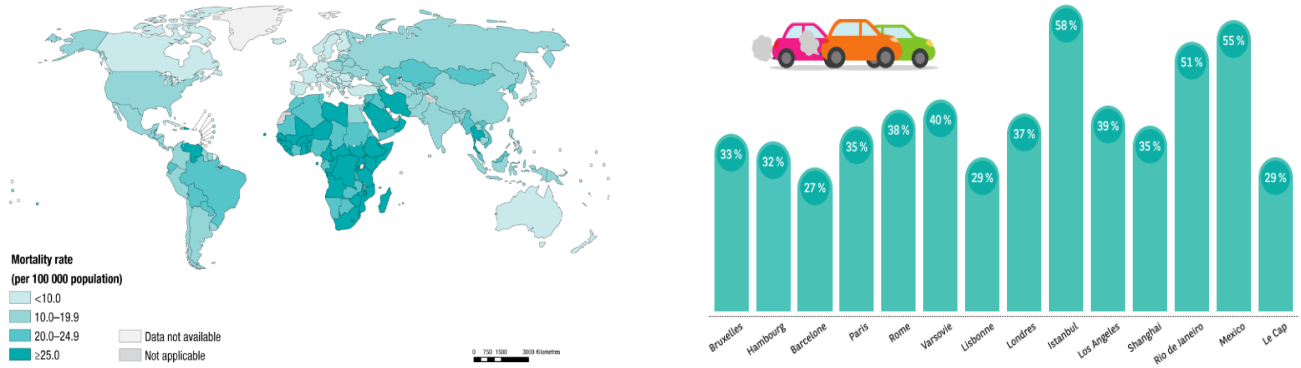
1.2 The autonomous car: this dream of modern man

Driving a car is very easy. Let us turn on the engine, open our eyes, foot on the accelerator, and we are totally seated. So why bother developing autonomous cars? The reason we want an autonomous car is that we hope that its commissioning will improve the quality of life within society. It does not get tired on the road, nor is it distracted by a mobile phone. It has no emotions, and does not drink alcohol. According to the NHTSA [1] definition, a car is totally autonomous if it can drive in real traffic and on a non-specific infrastructure performing all necessary control actions, in all environmental conditions, without any human intervention. It must be able to be as equal as those of a human driver must. Therefore, an autonomous vehicle is primarily a vehicle. Its characteristics do not have to necessarily differ from a non-autonomous vehicle. The only requirement is to be able to control the various elements of the car (mainly direction and engine power) via electrical controls. It requires an interface between the entity that makes the driving decisions, and the systems that influence the physical behavior of the vehicle.

In recent years, autonomous driving has been the center of significant interest in both academia and industry research. This has been motivated by the various potential benefits that the autonomous vehicle could bring in our daily lives. Indeed, the generalization of such a technology can save millions of lives, because of a better reaction time, ensured by a greater reliability of the computer systems. Noting that more than 1.3 million people die annually [Curiel-Ramirez 2018] from traffic accidents, and $94\,\%$ of those incidents result from human errors [Van Brummelen 2018] (Figure 1.1(a)). It helps to minimize the risk of major dispersion elements such as smartphone use and reading messages while driving. The use of self-driving cars improves the quality of life for people around the world; anyone can be alone in cars without wanting to drive, or without being able to drive because of fatigue or illness. It allows transportation for public traditionally excluded from mobility such as persons

[1] **NHTSA**: The **N**ational **H**ighway **T**raffic **S**afety **A**dministration

with reduced mobility, young and old age people unqualified to drive, inhabitants of peripheral or under-served rural areas.



(a) (b)

Figure 1.1: *(a) Estimated road traffic fatalities per* 100 000 *population in 2013 (world Health Organization) (b) Level of urban congestion: additional travel time compared to fluid traffic; Daily average (TomTom traffic).*

Getting rid of congestion is very important too (Figure 1.1(b)). These new systems may help to solve it, through better circulation, and instant homogenization of traffic, thanks to the communication system between vehicles. The safety distance currently left between cars ranges from 40 to 50 meters on the road, which will be only 6 meters in the case of independent vehicles, which will also comply with the allowable speeds, so the number of cars would increase by more than 273 % [Lutin 2018].

It is expected that the automation of driving will have repercussions far beyond the technological domain. Because it is the entire automotive and transportation industry that will be impacted. Today, the car is a commodity bought and personal. In the future, autonomous vehicles could be ordered and pick up their passengers. The automotive industry would no longer sell goods but transport services. For instance, the free travel concept, which is an idea recently patented by Google inspired by its business model. The idea, in short, proposes free autonomous taxis that will not pay for the passengers but on the advertising diffused within the vehicle or by merchants wishing to attract customers by paying their way. Autonomous cars also can improve the delivery sector, through the automatic delivery of products from grocery stores or supermarkets. We can also cite the decrease in the number of car parks, especially in the city center, since the car can deposit its occupants and park alone much further (outside the city). Finally, these self-driving cars adhere by the rules more than humans, reducing fines and traffic violations, as well as releasing the police to other tasks.

1.3 Levels of driving automation: autonomous driving is here

From driver assist through to fully automated driverless vehicles _ is autonomous driving is already there or should we wait to see those cars filling roads and streets? To answer this question, at first, it can be useful to define

(a) (b) (c)

Figure 1.2: *Example of future autonomous services. Left: autonomous bus. Middle: autonomous pizza delivery. Right: automatic delivery of products. The autonomous driving technology will revolutionize many sectors such as transport or product delivery. Customers will no longer need to go to the shops to buy their needs, but the UGVs (unmanned ground vehicle) and UAVs (Unmanned Aerial Vehicle) will carry food, clothing, and any kind of goods. In many cases, the goods will arrive at home at the same time as the arrival of pizza in 2018.*

the different levels of automation. For this purpose, SAE [2] and NHTSA define six levels of automation for road vehicles, as displayed in Table 1.1

Level 0 is titled "manual driving". In this level, the driver is in total charge of the vehicle. He performs all driving functions at all time. Level 1 "Driver assistance", at this level, an automated system can performs some driving tasks, such as lane-departure warning (LDW), anti-skid braking (ABS), electronic stability program (EPS), or adaptive cruise control (ACC). Most cars currently on the road integrated this level of automation. Regarding level 2 "partial automation", the system performs at least two main control functions. The driver has a rather supervisory role during the intervention of the system. The vehicle at this level is equipped with more advanced driver assistance functions, which can intervene in more complex situations, such as lane keeping assist (LKA), hands-free parking (HFP) or traffic jam pilot (TJP). Tesla's Model S and Model X [3] are already equipped with these abilities. These first three levels have already been achieved because they include only driving aids (ADAS) which allow a partial delegation of the driving. It is important to note that until then, the driver remains responsible for his car, and must remain vigilant and ready to intervene quickly. The vehicle of Level 3 monitors its environment while controlling its own movement, but always asks human drivers to remain vigilant and ready to take control of the car in case if necessary. The type of application is the same as Level 2, but the responsibility is transferred to the machine. Google Car possessing a human driver in cases of emergency is considered at this level. The last two levels involve technology that might be mature enough to be released soon but will be available in the coming years. Level 4 specifically is defined as "intensive automation", which means that the vehicle has an almost complete capacity for autonomy and can counter the driver's decision. The presence of the driver in the vehicle is needed only in some driving modes. Level 5, known as "Full self-driving automation", means that the vehicle is intelligently designed to perform all driving tasks, under all conditions, without having a driver at any time. An example of this level is given by the VipaLab and EZ10 vehicles [4], shown in Figure 1.2, which has no steering wheel, gas pedal, or brake pedal.

[2] **SAE**: Society of Automotive is an international organization that exchanges information and ideas for everything related to vehicle engineering.

[3] Source: https://www.tesla.com/

[4] Source: http://www.institutpascal.uca.fr/index.php/en/the-institut-pascal/equipments

Table 1.1: *Comparison with the state-of-the-art methods.*

	LEVEL SAE	Name	narrative definition	direction, acceleration and deceleration	conduct environmental monitoring	use of dynamic driving	capacity of the system (driving modes)
The human driver controls the driving environment							
unsupervised	0	No Automation	The human driver carries out all driving tasks	human driver	human driver	human driver	n/a
	1	driver assistance	The specific execution of a driving mode by a driver assistant system for a steering maneuver or for acceleration/deceleration using the information of the driving environment and waiting that the human driver performs all other remaining tasks	human driver and system	human driver	human driver	certain driving modes
	2	partial automation	The specific execution of a driving mode by one or more driver assistant systems of a steering maneuver or of acceleration/deceleration using the information of the driving environment and waiting that the human driver performs all other remaining tasks.	system	human driver	human driver	certain driving modes
The system of autonomous driving (the system) controls the driving environment							
supervised driving	3	conditional automation	The specific execution of driving mode by an autonomous driving system of all aspects of a dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene.	system	system	human driver	certain driving modes
	4	intensive automation	The specific execution of driving mode by an autonomous driving system of all aspects of a dynamic driving task, even if the human driver does not properly respond to an intervention request.	system	system	system	certain driving modes
	5	Full automation	The permanent execution by an autonomous driving system of all dynamic driving tasks that	system	system	system	All driving modes

1.4 A brief history of autonomous driving: back to the past

Some might think that the history of self-driving cars goes back for a few years, but the experiments on this kind of car started since the 1950s, while the first real autonomous car appeared at the Tsukuba Robotics Laboratory in Japan in 1977. It was with the launch of the autonomous car that follows a special marking on the ground. This car was able to reach a speed of 30km/h. In the 1980s and precisely in 1984, Mercedes-Benz in collaboration with University of the Bundeswehr in Munich, tested an autonomous van equipped with cameras. It reached a speed of 100km/h and was capable of driving on a highway without traffic [Dickmanns 1992]. Two years later, the "NAVAB" laboratories of the Carnegie Mellon University launched the development of NAVLAB [Thorpe 1988] autonomous vehicles, which uses RALPH (Rapidly Adapting Lateral Position Handler) software to locate the road and steer the car autonomously on it. Tests were conducted over 4000 km on a variety of road types. The car was able to reach 90km/h on some parts of the tests, but the average speed was 50km/h [Dean Pomerleau 1995]. In 1987, the European PROMETHEUS [5] program totaling 800 million euros helped to develop the technology needed for autonomous cars. Launched at the request of the automotive industry, in particular German industry, PROMETHEUS was the first large-scale European research initiative to improve long-term road traffic both in

[5] **PROMETHEUS**: **PRO**gra**M**me for a **E**uropean **T**raffic of **H**ighest **E**fficiency and **U**nprecedented **S**afety

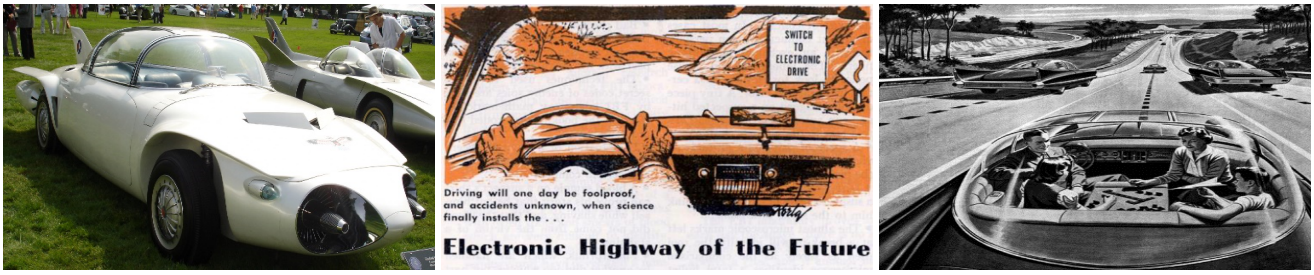
terms of the vehicle and the infrastructure [Benenson 2008]. In 1994, the first results of this project appeared and a demonstration was conducted in a real situation from Paris of two autonomous cars (VaMP and VITA-2) capable of changing lanes and overtaking at a speed of 130 km/h [Dickmanns 2003]. Further results are achieved in 1995, when the autonomous S-Class Mercedes-Benz driving from Munich to Bavaria to Copenhagen in Denmark and back. This car reached 175 km/h on the Autobahnhighway in Germany [Vitor 2014]. At about the same time, an automated steering vehicle drove from Washington DC to San Diego, in "No hands across America" tour. The driver controlled only the acceleration and the braking [Pomerleau 1996]. In parallel, other features are starting to be developed, as the case of automated parking [Paromtchik 1996] or following other vehicles. Three years later, Schiphol airport of Amsterdam tested a fleet of vehicles without a driver, driving on a dedicated road. In 1999, Siemens put an autonomous bus in operation. This bus was based in its direction on specific markings on the road, while a human driver controls the speed. After two years, Toyota introduced its autonomous bus capable of driving without human involvement on dedicated roads.

(a) (b)

Figure 1.3: *VipaLab and EZ10 autonomous vehicle (Pascal Institute).*

In the early 2000s, considerable advances were made on the other side of the Atlantic on autonomous ground vehicles, with the launching of DARPA [6] research program [Seetharaman 2006]. The latter has boosted research on the various areas related to the autonomous car, ranging from perception to control of movement. DARPA launched a series of challenges: the "Grand Challenges" in 2004 and in 2005 and the "Urban Challenge" in 2007 (Figure 1.5) [Siciliano 2009]. These challenges have demonstrated that it is possible to design vehicles capable of crossing all types of roads, from dirt roads to rugged mountainous passages, in fully autonomous mode. They showed that new important capabilities were possible, and that cars, might one day not need drivers. However, the most important thing is that these projects had opened the doors of this area of autonomous driving, and had influenced many people and had encouraged them to get involved in research in this field. From there, and in 2009, Google recruited the winner of the urban challenge, S. Thrun, and began manufacturing and developing technology to build a self-driving vehicle. Google tested its technology with Toyota cars on highways in California in the same year. In 2010, it starts to design its own autonomous 2-seater vehicle, which introduced it four years later. It has come as an electric vehicle with a maximum autonomy of 130 kilometers and can reach speeds of 40 km/h. In

[6] **DARPA**: **D**efense **A**dvanced **R**esearch **P**rojects **A**gency



(a) (b) (c)

Figure 1.4: *Left: The General Motors Firebird II was described as having an "electronic brain" intended for use with "the highway of the future" (1956). Middle: Automatic Highway: An experiment conducted by RCA Labs and the State of Nebraska on an automatic highway in 1958. This highway contained detection circuits installed in the roadway, which could detect the speed of the car and send it guidance signals [Schwarz 2013]. Right: Advertising of autonomous vehicle in the United States in 1965 (Computer History Museum).*

May 2012, the state of Nevada in the USA issued its first self-driving license. That year, Google announced that its cars had completed half a million kilometers of self-driving on highways without incident. In July 2013, the VisLab Company of the Italian University of Parma set out its self-driving vehicle "BRAiVE" on multiple streets open to public traffic. In 2014, Mercedes-Benz launched S-Class vehicles with intelligent drive featuressuch as autonomous braking, lane keeping assist, speed control [Ziegler 2014].

(a) (b)

Figure 1.5: *Left: Stanley Won the 2005 DARPA Grand Challenge. Right: Boss Won the 2007 DARPA Urban Challenge.*

In 2015, five USA states (Nevada, California, Virginia,Florida and Michigan), along with Washington, D.C., allowed the full-fledged auto-driving test on public roads. During the same year, experiments on this type of vehicle began in European countries such as the United Kingdom and France. Germany, the Netherlands and Spain have also allowed the testing of autonomous cars on public streets. Since then, many major companies and research organizations have developed prototypes of self-driving vehicles. The development of these vehicles has grown rapidly with the participation of more than 35 automotive and technology companies, including General Motors, Toyota, Apple, Google, Intel, Audi, BMW, Tesla and Uber, Ford, and many others. The race to the $100\,\%$ autonomous vehicle is now well and truly committed. The following table summarizes the most autonomous

driving projects and their faced problems since 2016.

Table 1.2: *Summary of different autonomous driving projects and its faced problems [Van Brummelen 2018].*

PROJECT(S)/COMPETITION(S)	PROBLEMS EXPOSED/ ADDRESSED BY PROJECT	CURRENT STATE OF PROBLEM: LARGELY ADDRESSED (LA), RELATIVELY ADDRESSED (RA) OR LARGELY UNADDRESSED (UA)
PROMETHEUS (1987–1995)	. Autonomous lane Keeping	. LA
	. Adaptive cruise control	. LA
	. Automatic emergency calling systems	. LA
NO HANDS ACROSS AMERICA (1995) MUNICH TO ODENSE UBM TEST (1995), ARGO (1998)	.Vision-based object detection/tracking	. LA
	. Perception in unfavorable lighting conditions	. RA
	. Improvement of obstacle and road marking detection	. RA
	. Complexities of urban driving	. UA
	. Perception in difficult weather conditions	. UA
DAPPA GRAND CHALLENGE, (2004) SECOND DAPPA GRAND CHALLENGE (2006)	. Off-road navigation	. LA
	. Obstacle avoidance	. RA
DAPPA URBAN CHALLENGE (2007)	. Traffic light and sign detection	. LA
	. Ability to test in real traffic situations	. RA
	. Obstacle detection – especially pedestrian and cyclist detection	. RA
	. High-speed autonomous driving; efficiency of detection algorithms	. RA
	. Complex urban driving (dense traffic, intersections, etc.)	. UA
HIGHLY AUTOMATED VEHICLES FOR INTELLIGENT TRANSPORATION (HAVEIT) (2008–2011)	. Temporary autonomous driving	. RA
	. V2V to increase redundancy in data	. RA
	. Safety software architecture of AVs; detection of hardware/software/sensor failure	. UA
SAFE ROAD TRAINS FOR THE ENVIRONMENT (SARTRE) (2009–2012)	. Vehicle platooning and relevant environmental and safety benefits	. RA
VISLAB INTERCONTINENTAL AUTONOMOUS CHALLENGE (VIAC) (2010)	. Vehicle platooning in real traffic situations	. LA
	. Vehicle platooning without a priori information	. LA
	. Autonomous driving without a priori information	. UA
GRAND COOPERATIVE DRIVING CHALLENGE (2011)	. Efficient cooperative driving intersections	. UA
EFUTURE (2013)	.Energy-efficient AV technology	. RA
	. Standardized Advanced Driver Assistance Systems (ADAS)	. RA
	. Data fusion for increased perception accuracy	. RA
	. Human acceptance of AVs	. UA
EUROPEAN TRUCK CHALLENGE (2016)	. Real-world platooning using V2V communication	. RA

1.5 Towards Autonomous Driving: the revolution has begun

Autonomous driving technologies have attracted much attention in the last decade because of their rapid development and their potential benefits in terms of safer and convenient mobility. In this section, we will discuss

the significant development of both research and industry for the autonomous driving field.

1.5.1 Research

To get a better idea of the activity of a given field, the databases of published articles and patents are always a good indicator. They provide a good insight of the development of this field. Regarding autonomous vehicles, Thomson Reuters recently published a global information study on published research papers and patents. Figure 1.6 displays the number of inventions related to autonomous technologies by year between 2010 and 2016. More specifically, in four years, between 2012 and 2016, the patent rate of connected and autonomous vehicles has exceeded the 1200 patents (According to the Oliver Wyman cabinet). Technology companies, baited by earnings prospects,are the pioneers with almost a third of the patents. However,the first place of the ranking is for the German manufacturer Audi, who deposited 223 patents, followed by Google with 221 patents. The third on the podium is the BMW manufacturer with 198 patents. Among the top 10 companies, there are Facebook, Uber, Microsoft, Amazon, and Apple. The advantage of these companies is to have vast financial possibilities and to be accustomed to recruiting the talents necessary for the development of new software and hardware needed for this technology.

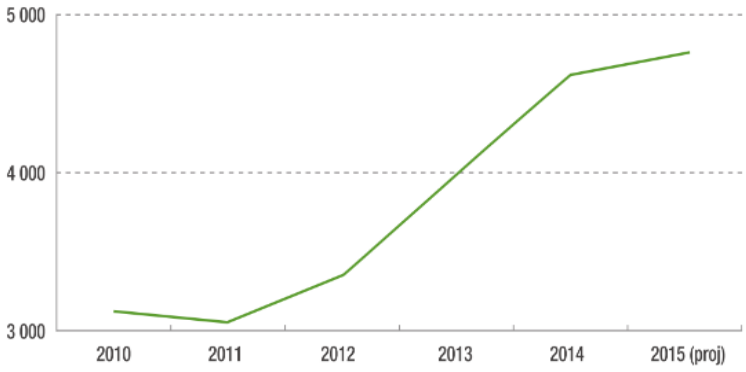


Figure 1.6: *Inventions of autonomous technologies by year of publication in the world [Thomson Reuters 2016].*

Regarding academic research, the total number of published papers between 1990 and 2018 exceeds 23 500. This result is obtained from IEEE Xplore with the keywords “autonomous vehicle”. The same observation is noted from Web of Science database. Figure 1.7 shows the evolution of the number of publications on the Web of Science database over the years. From this figure, a publication jump is observed between 2005 and 2007. This coincides with the DARPA challenges, which opened the doors of this area of autonomous driving, and had influenced many researchers and had encouraged them to get involved in this field [Van Brummelen 2018]. This was also motivated by the appearance of the 3D LiDAR marketed by Velodyne, who owes his appearance to these competitions. Since it was used for the grand challenge (2005). Even if the vehicle of the Velodyne Company had to abandon the race because of the damage to the LiDAR by the vibrations after several miles, the company created a product that has since become a staple in the race to the vehicle automation. Another factor contributed to the development of research in autonomous driving field is allowing self-driving cars to conduct experiments on public roads, with the legalization of several laws regulating these experiments. This is started at the very beginning in the state of

Nevada in the USA in 2012, which issued its first self-driving license that year. This appears in Figure 1.7 by the overcoming the barrier of 1000 published paper per year. All these reveal the vitality of this field and highlight the huge trend towards this technology.

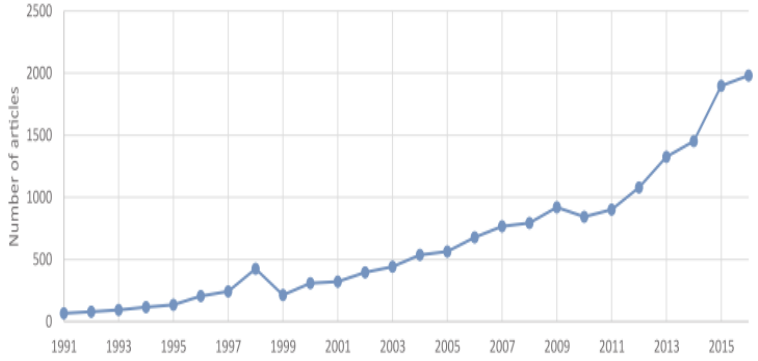


Figure 1.7: *Number of articles on the Web of Science database related to autonomous vehicles [Van Brummelen 2018].*

1.5.2 Progress made by industry: everyone is racing ... ferocious pursuit

The impetus towards autonomous driving had given rise to fierce competition between various companies. Indeed, a lot of them are working on projects concerning the autonomous driving whether it be automakers like, Audi, Tesla, Mercedes-Benz, and Renault. As well as other companies specialized in a completely different field, such as Google, Bosch, or even Uber. All announce wanting to market their own self-driving car for 2020.

This automatisation of vehicles can take two ways: the way of partial automation or the way of full automation.

1.5.2.1 Partial automation

Over the past three decades, the car industry has developed more and more driving assistive technologies. The goal was to facilitate the driving task and improve road safety. Technologies such as adaptive cruise control, emergency braking, forward collision warning, lane-keeping assistance etc.,monitor the driver to avoid distraction and drowsiness [Rezaei 2014, Vicente 2015], or alerting him/her about potential dangers [Carvalho 2015]. In a similar vein, the readers are referred to the survey [Ohn-Bar 2016] made by Ohn-Bar and his colleagues, which highlights three main domains where humans interact with vehicles: 1) inside the vehicle cabin, 2) around the vehicle, and 3) inside surrounding vehicles. This is done to anticipate potential dangers for a safe and comfortable ride. All these functionalities are covered by the general term of Advanced Driver Assistance Systems (ADAS). The combination of all these technologies gradually transforms the vehicles to look more like autonomous cars. Table 1.3 summarizes some driver assistance features.

Some forms of partial vehicle autonomy are already available. Nowadays, famous automakers provide models that can perform certain tasks autonomously. This can be viewed as a prelude to the development of fully self-driving cars in the future. For instance, Mercedes Benz produces a car from the “S-Class” series (Figure 1.8).

Table 1.3: *Comparison of five driver assistance functions, ACC: adaptive cruise control; SAB: Semi-Autonomous Braking; AFD: Autonomous Freeway Driving; ALC: Autonomous Lane Change and SAP: Semi-Autonomous Parking. [Schwarz 2013, Van Brummelen 2018]*

VEHICLES	ACC	SAB	AFD	ALC	SAP
AUDI A6	✓	✓	✓		✓
BMW750I XDRIVE	✓	✓	✓		✓
FORD	✓	✓	✓		✓
INFINITI Q50S	✓	✓	✓		✓
LEXUS RX	✓	✓	✓		✓
MAZDA3 [Ulrich 2014]	✓	✓	✓		✓
MERCEDES-BENZ E AND S-CLASS	✓	✓	✓		✓
OTTO SEMI-TRUCKS	✓	✓	✓		✓
RENAULT GT NAV					✓
TESLA MODEL S	✓	✓	✓	✓	✓
VOLVO XC90	✓	✓	✓		✓

(a) Mercedes-Benz autonomous S500 Intelligent Drive.

(b) S-Class active brake assist

Figure 1.8: *Mercedes Benz develop many assistance systems, extended functions and innovative protection systems under the name "Intelligent Drive".*

It alerts the driver when it deviates from its trajectory or when it approaches the car in front of it. If the driver does not respond to these alerts, the car will drive the wheels and return to the stable position, or apply the brakes automatically.

Audi has incorporated the natural adaptation features with the surrounding cars to the latest version of its future A7, called "Jack" (Figure 1.9). The company claims that "Jack" performs all road maneuvers like any human or better, such as keeping an appropriate distance between itself and other cars or trucks, and sending alerts while performing autonomous line change.

The company has developed an ADAS system called "zFAS" equipped with highly sophisticated information processors. All environment sensors (radar sensors, front camera, and ultrasonic sensors) are directly connected to the ZFAS and the environmental representation is calculated. As soon as the surrounding environment is well-known and understood, zFAS can make a decision based on this, i.e. choosing the next maneuver and planning a

valid trajectory that is transmitted to the vehicle's actuators [Will 2017].

Audi is currently testing self-driving cars on the autobahn-A9 test highway, the first test highway in Germany with an infrastructure fully dedicated to autonomous vehicles. This highway is equipped with clearer and more adaptive markings that can be captured from farther distances, which facilitates testing processes.

(a) (b) (c)

Figure 1.9: *Audi A7 piloted driving concept.*

In France, in February 2014, Renault unveiled its Next Two project (Figure 1.10), which consists of a "Renault Zoe" as concept vehicle equipped by a driving delegation function. The driving delegation concept, on this model, is only available on expressways and highways where there are no pedestrians or cyclists, or on any road during traffic jams if the speed is less than 30 km/h. This vehicle is equipped with a radar placed on the front shield, a camera located at the central mirror, a rear camera and a belt of ultrasonic sensors around the vehicle. "Zoe" also is able to park or pick up its driver thanks to an autonomous valet.

(a) (b) (c)

Figure 1.10: *"Zoe" the concept vehicle of the Next Two Renault project.*

At present, because of the limitations and high costs of available sensors, most commercial vehicles only consist of Level 1 or Level 2 of autonomy, which require constant driver attention and control. The ADAS features in these vehicles generally consist of adaptive cruise control, lane keeping, and hands-free parking. Nonetheless, Level 3 autonomous features are available in the Tesla Model X and Model S (Figure 1.11). The latter is a well-known example of a semi-autonomous commercial vehicle. It can autonomously drive along highways

by performing lane changes and adjusting the speed. However, in complex situations, such as intersections or unknown or unpredictable situations, the human driver must take control of the vehicle [Van Brummelen 2018]. Although Tesla has made great strides in the field of advanced driver assistance, recent accidents have raised concerns about the ability of the driver to use technology safely. This was demonstrated by an unfortunate event of May 2016, when a Model S Tesla vehicle crashed on a truck causing the death of its driver [Birek 2018].

Figure 1.11: *Semi-autonomous driving system on the Tesla Model S.*

Other vehicle prototypes, equipped with more advanced ADAS systems, are the Volvo XC90, the Infiniti Q50, and the BMW i3. These prototypes have an autonomy level very close to the Level 3 (where the vehicle monitors its environment while controlling its own movement), with reinforced and permanent driver's attention. Nevertheless, the use cases and the operating situations are very limited and the robustness of the system with respect to sensor defects and degradation of measurements is not yet sufficient.

Besides that, other companies refuse to add semi-autonomous features to their cars and aim to produce completely autonomous cars. These companies aim to procure complete automation, which will be addressed as follows.

1.5.2.2 Towards full automation

Starting from the beginning, this was initiated after the DARPA challenges, when Google has hired S. Thrun and launched the Google Car project (known today as Waymo [7]) in 2009. Several different types of cars have been converted into autonomous cars to conduct tests, including the Toyota Prius, Audi TT, Lexus RX450h (Figure 1.12(a)) and Fiat Chrysler Pacifica. Google has also developed its own custom vehicle. Introduced in 2014, it is an electric vehicle designed entirely by Google without steering wheel, gas pedal, or brake pedal (Figure 1.12(b)). It has also entered the automotive field through the "Android auto" technology that brings the (Android) operating system to cars. In late 2016, Google Corporation (Alphabet Inc.) announced that its technology would be launched to a new subsidiary called Waymo [Teoh 2017]. The name Waymo is derived from its mission, "a new way forward in mobility". Waymo passes to the higher speed. After acquiring a hundred

[7] Source: https://www.waymo.com/

Chrysler Pacifica vehicles in 2016 and 2017, it is an order of 62,000 vehicles that Waymo has passed to Fiat Chrysler, announced the company on May 2018. These vehicles will be equipped with sensors by Waymo in order to achieve a level 4 autonomy, i.e. the vehicle can move without a driver on board, in accordance with the company's successful experiments in November 2017. Waymo, which is considered as the pioneer of driverless vehicle technology, has in its possession up to now more than 10 million miles (sixteen million kilometers) traveled on public roads across 25 US cities [8](Figure 1.12(c)). Of course, miles do not prove anything about vehicle safety, But Waymo's accelerated testing program is just one of several signs that the Alphabet Company is investing heavily to maintain its early lead in autonomous driving technology.

(a) (b) (c)

Figure 1.12: *Google's self-driving car. (a): Lexus RX450h. (b): Google designed car. (c): Graph showing the evolution of the tests made by Waymo. It shows that it took more than 5 years to Waymo (ex: Google car) to reach a million miles. Then it took over a year to get from one million to two million miles. After, about six months each to get to 3 million and 4 million miles.*

From San Francisco to New York, Delphi led a 5633-kilometer trip in their Audi SQ5 vehicle (Figure 1.13(a)) called the "Roadrunner" in Mars 2015. This vehicle was equipped with an autonomous driving system with 6 long-range radars, 4 short-range radars, 3 vision-based cameras, 6 LiDARs and a high-resolution GPS system. This project called "US Coast-to-coast automated Drive" aims to cross the United States from the west coast to the east coast. The journey was covered with 99 percent of the drive in fully automated mode, according to Delphi.

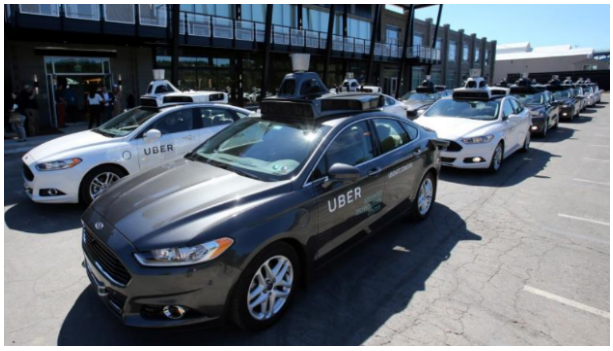
In August 2016, Delphi was tapped by Singapore to begin a pilot program for the world's first automated taxi program using Audi SQ5s. Tests are scheduled to run for three years.

We cannot talk about autonomous cars without talking about Uber. This Silicon Valley startup is the most important company in the field of luxury passenger cars and has recently announced its work on a self-driving automobile project. In late 2016, it launched a demonstration of autonomous Volvo and Ford cars in Pittsburgh. A few months later, it did the same in San Francisco. The cars will be collecting mapping data as well as testing its self-driving capabilities. In March 2017, an Uber test vehicle was involved in an accident that caused the pedestrian's death. The causes of this accident include low visibility, blind crossing of the street at night, crossing a high-speed road. Some experts say that a human driver could have avoided the fatal accident, which shows that there is still a long way to transfer to the autonomous vehicle all the human competence.

On the French side, we find the Lyon-based start-up Navya created in 2014 and specialized in the design

[8]Source: https://waymo.com/ontheroad/

(a)



(b)

Figure 1.13: *(a): the "Roadrunner" vehicle of Delphi. (b):Uber owns a fleet of self-driving cars.*

of autonomous vehicles. In 2015, it launched its autonomous Arma shuttle shown in Figure 1.14(a), the first autonomous serial driverless vehicle to be marketed. Since September 2016, the company has started tests in the Confluence district of Lyon and since July 2017 in La Défense (Hauts-de-Seine). Still in France, in La Rochelle, autonomous minibuses transported 15 000 passengers in 2014 and 2015. In Sophia Antipolis, three vehicles were tested in 2016. In July 2017, the metropolis of Toulouse experimented in Pibrac an electric shuttle without a driver. In the Department of Puy-de-Dôme, experimentation is multiplying, with the example of the Michelin site of Ladoux that collaborates with Transdev and the start-up EasyMile (Figure 1.14(b)).

(a) The Navya autonomous Shuttle

(b) EZ10 shuttle, from the EasyMile company

Figure 1.14: *French autonomous cars.*

1.5.3 Giant alliances to draw the future of self-driving cars

A number of the most important and influential companies in the world have invested billions of dollars in autonomous car technology and these are not speculative bets. In addition, new configurations of partnerships are being created, giving rise to giant alliances to draw the future of autonomous cars. Alliances of a new type like Microsoft partnership with Toyota, PSA and IBM, Purchase of Otto by Uber, Apple with the VTC platform

Didi Chuxing, and Google car becomes Waymo, a subsidiary in partnership with Honda and Fiat. This devastating entry of world-class digital actors with colossal financial capabilities has made a significant contribution to the rapid progress achieved over the last years and continues with this new pattern of partnerships to develop and update this field for one purpose, produce the autonomous car as soon as possible. The main risk is to lead to totally unbalanced partnerships, in which the role of the manufacturer would be limited to the deployment of the operating system. The manufacturer only provides the vehicles and leaves all the responsibility of the artificial intelligence and data management to the digital enterprise.

1.5.4 Millions of kilometers traveled

For the manufacturers, autonomous vehicle tests can accumulate experience over millions of kilometers traveled. For instance, Google's autonomous vehicle fleet has on its own up to now more than eleven million kilometers; Uber's fleet had logged 3 million kilometers. Tests also offer the opportunity to validate the technologies in real conditions and to improve their efficiency by learning from errors and encountered situations. The goal is to treat as many situations as possible, including the most critical, at night or in poor weather conditions (rain, snow, ice). In addition, and more importantly, it is a way to convince the public with concrete examples.

1.5.5 Only one conclusion ... Everyone wants it

The work for self-driving car development is accelerating dramatically, involving more than 35 car and technology companies, including Apple, Google, Intel, Audi, BMW, Tesla, Ford, Vodafone, Nissan, etc., all are racing to be the first to introduce self-driving cars on the market, and each seems to want to bet on this promising market.

1.5.6 Institut Pascal: where is it in this fierce race

The involvement of the Institut Pascal in the field of the autonomous car began in the late eighties, with the partnerships on the development of driving assistance with PSA and the participation in the PROMETHEUS project. However, the first autonomous test vehicle was acquired in 2000, which is the CyCab (Figure 1.15). The latter is an electric two-seater serving as an experimental platform for some laboratories in France, including the Institut Pascal [Nizard 2017]. Since 2006, in collaboration with Ligier Group, have been developing self-driving shuttles named RobuCab for transporting up to 10 people. 4 years later, the VipaLab is introduced. This is the laboratory vehicle currently used by the experimenters to validate their work. This vehicle, which is intended to be deployed on dedicated sites such as PAVIN, is able to handle obstacles with laser rangefinders installed at the four corners of the vehicle. The following prototype is called Vipa, brings comfort, aesthetics, safety and increased maneuverability. In 2014, the EZ10 improves the whole concept. This vehicle is designed to be used in urban areas, such as car parks, serving a hospital or airport terminal from a distant car park [Marmoiton 2016]. Figure 1.15 draws the history of local developments related to the autonomous vehicle at the Institut Pascal.

With this progress, the demands for experimentation under realistic conditions became more and more urgent. The laboratory acquired in 2008 a space called PAVIN [9] (Plateforme d'Auvergne pour les Véhicules Intelligents)

[9] PAVIN: http://www.institutpascal.uca.fr/index.php/en/the-institut-pascal/equipments

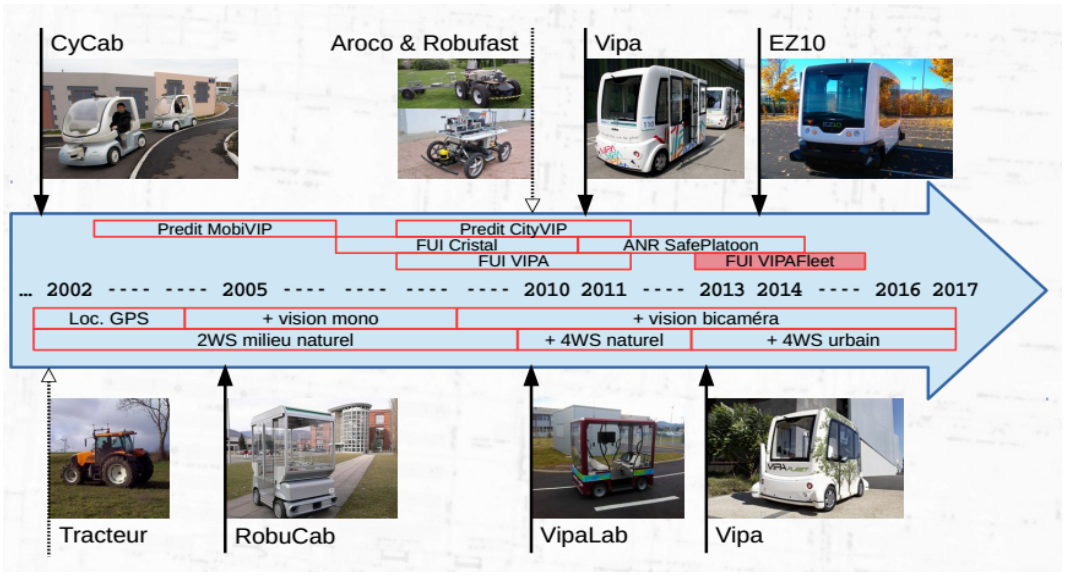


Figure 1.15: *History of local developments related to the autonomous vehicle at the Institut Pascal.*

to test urban vehicles without drivers.

1.5.6.1 PAVIN

The PAVIN platform is an experimentation site that allows testing on a reduced scale, and in a safe way, autonomous vehicles moving in an urban environment as shown in Figure 1.16. The site thus consists of a set of streets and crossroads of different natures (single and double lanes, roundabouts, stops, traffic lights, etc.).

(a) (b)

Figure 1.16: *The PAVIN platform.*

1.6 Towards Autonomous Driving: the road is still long

A long and complex road ahead of the autonomous car to become the standard of the future driving. A fully autonomous car requires planning its itinerary based on its location and the surrounding environment. The process that allows the vehicle to memorize its environment, inform it of its current position, and then define the strategy to reach its goal can be divided into four phases: perception, localization, path-planning and vehicle control, as illustrated in Figure 1.17. Perception uses sensors to scan and monitor the environment continuously. This includes the ability to build a map representing the spatial structure of the environment, to update a pre-built map or to correct an incomplete or erroneous map. Localization allows determining the position of the vehicle in this map that corresponds to its exact position in the real world. Once the location of the vehicle in the map is established, path planning generates a safer path to its destination. Finally, vehicle control phase allows the vehicle to anticipate the movements to be carried out to achieve its goal. These four phases constitute a recursive process that must be executed at high frequency. This is in order to manage dynamic objects of high speed (cars, motorcycles) and of low speed (cyclists, pedestrians) [Van Brummelen 2018]. On the way towards autonomous vehicles, there are still important challenges to face. Obviously, there is still technical work to do to make the navigation process more efficient and reliable. In essence, there are several fundamental questions that a self-driving car must answer, including:

- Which sensors should be used to insight its environment? (Perception)
- What does its environment look like? (Perception)
- Which localization and mapping techniques should it use with respect to sensing the ego-vehicle's environment? (Perception)
- Where is it in the world? (Localization)
- Where is the road? (Path planning)
- How does it reach its goal with respect to the find path? (Control)

These questions, which are usually performed in an easy and automatic way by the human driver, are incredibly challenging to the modern driving system. From all these challenges and in spite of the progress made every day to achieve the improvement of the latter, the precise localization within the range of few centimeters remains the major challenge for it. Indeed, locating and understanding the position of an autonomous vehicle is essential for it to make the right decisions. In addition, it requires a good perception system. Therefore, its importance because it involves almost all autonomous driving phases, and if it fails, the entire autonomous platform would no longer be able to operate.

During this work, we were interested in algorithmic concepts and tools allowing the precise localization of an autonomous vehicle in its environment, with a small reach to the mapping techniques.

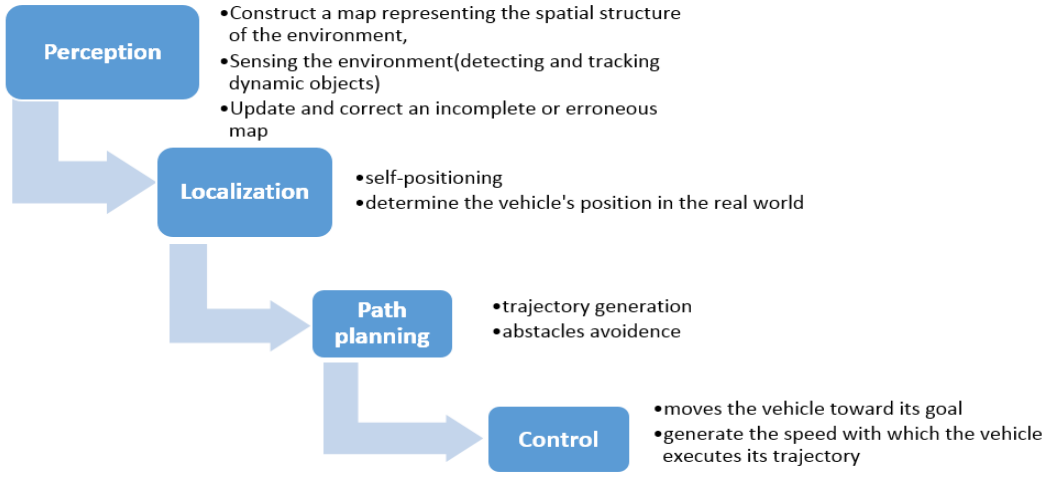


Figure 1.17: *Main capacities of an autonomous car on the road and the means to achieve these capacities.*

1.7 Positioning the thesis in the provided framework

Localization is a fundamental step for any autonomous vehicles. It deals with the problem of estimating the vehicle's position relative to a map. As the Cheshire cat says in Alice in Wonderland, **"If you don't know where you're going, any road can lead you there"**. We say in this thesis **"If you do not know where you are, it is difficult to know in which direction you are heading to reach your goal"**. In the roboticists jargon, without localization, no path planning, no obstacle avoidance and therefore, no displacement or navigation.

In recent years, the ultimate sensor for localizing a vehicle is the GPS [10]. It provides a geo-referenced position from the order of a few centimeters to few meters proportionately to its price. This sensor, currently highly speared, has also become an essential application of smartphones. However, it is impossible to envisage the localization of an autonomous vehicle on a traffic lane that measures an average of 2 meters wide with such precision [Levinson 2007, Vivacqua 2017]. Moreover, in dense urban environments, the multiple reflections of GPS signals from buildings facades disturb the signal quality and therefore the location accuracy. Furthermore, this system does not work in indoor environments such as car parks, tunnels, etc.

To overcome GPS problems, some approaches use proprioceptive sensors such as encoders and IMU [11]. These sensors calculate the current position by integrating velocity and acceleration measurements from an initial position. Since each speed and acceleration measure is tainted by error, this leads to significant cumulated errors on the integration process. In addition, these sensors deliver false measurements if the wheels of the vehicle slide on the road. All these reasons, make the localization of autonomous vehicles only by odometer-type sensors, a solution difficult to envisage. Even with a GPS and IMU fusion, still cannot completely provide a precise localization in a dense urban environment [Vivacqua 2017].

[10] **GPS**: **G**lobal **P**ositioning **S**ystem
[11] **IMU**: **I**nertial **M**easurement **U**nit

On the other hand, exteroceptive sensors give measurements on the environment. These direct measurements, rich in a quantity of usable information, allow calculating the position of the vehicle in a precise way. Similar to the early navigators using stars to localize themselves and find their way, the vehicle uses the information provided by exteroceptive sensors (features, landmark, etc.) to localize itself. Among all the exteroceptive sensors that an autonomous vehicle can dispose of, we find the vision sensors, like cameras. These sensors measure the light intensity reflected by the objects of the environment. The extraction of information about the relative position between the camera and the objects of the environment allows the estimation of the current position of the vehicle with respect to an initial reference. Despite the fact that these sensors are inexpensive and energy-efficient, these sensors are still suffering from environmental variations (their sensitivities to lighting conditions, noise, geometric illusions, etc.) or common known factors (lack of overlap between images, texture-less surfaces, motion blur, etc.) [Sun 2018]. In addition, the image flow management remains a delicate and time-consuming operation. Another major drawback is that this type of sensor can only provide 2D information. In other words, any depth information is lost, unless using several sensors (stereovision) and the implication of very costly triangulation techniques to extract this type of measurement.

Countless works have been carried out for several decades, aiming to improve the measurements of proprioceptive sensors or to get better information from exteroceptive sensors. However, there are still many challenges, around both associated methods and algorithms as well as the sensors used.

Nevertheless, an alternative that has begun to gain attention recently, is to use a prior map to localize the car within it [Caselitz 2016]. Just as a human being, who is able to use a map, explores it and combines it with his visual inputs to locate effectively, the vehicle uses prior made maps that combine it with local data of its onboard sensors to find its position in the global map.

This alternative has generated a real question in the field of autonomous vehicles. Not all researchers and designers of these vehicles are convinced by the importance of using an a priori map of the environment, or even its need for autonomous navigation. Some of them believe that a vehicle does not need to know exactly where it is to react to the surrounding environment. They consider that by following the lanes, obeying signs and traffic lights and responding appropriately to other vehicles and road users, this vehicle can operate at least as well as a human driving on an unknown roadway.

On the other hand, other researchers do not agree with this proposal. They consider that maps facilitate the navigation process, and add an extra layer of security and understanding, as shown in Figure 1.18. The vehicle uses its onboard sensors to compare what it "sees" at a given time with what is stored in its memory. A priori maps allow the vehicle to better understand where it is, in the world, before starting to receive the real-time data. In this way, the vehicle has an idea about what should happen, it can see what is actually happening in real time, and therefore, can make a judgment on what to do. This has prompted many companies to take an interest in mapping services. Ranging from technology giants, such as Google, Uber and Intel to major car manufacturers such as Volkswagen, Ford, BMW and Mercedes. All of this has given rise to precise maps (Google Maps, OpenStreetMap) that allow the vehicle to navigate safely.

Both sides have pros and cons. In the present time, it seems that those who use a priori maps make the greatest progress, as it is shown in the Table 1.4 [Van Brummelen 2018]. In this thesis, we have chosen to invest in this class of methods (localization in known dynamic environments (refer to Chapter 2 for more detail about this cataloging)). We build our own maps with a dense and precise scanning in a static way using the LiDAR Leica

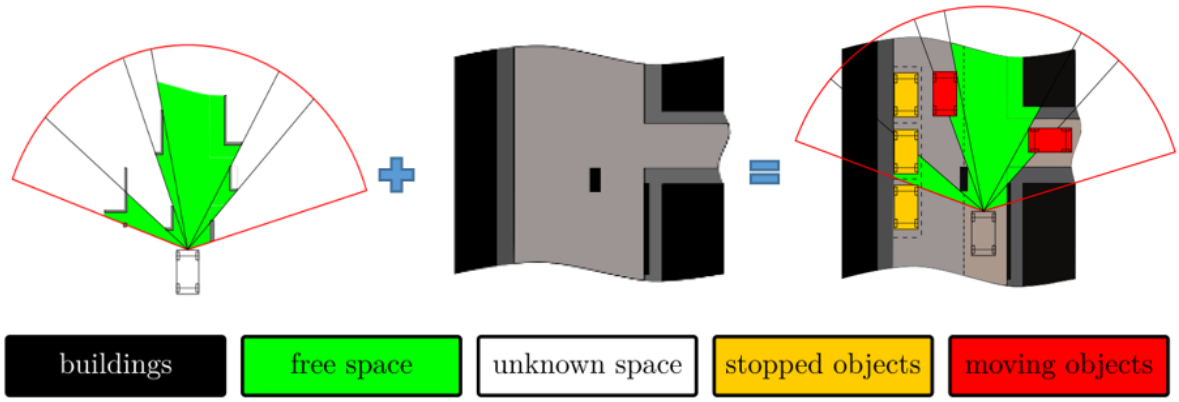


Figure 1.18: *Left: Data interpretation of a LiDAR point cloud. Middle: Prior map information. Right:The superposition of the map and the sensors outputs. An a priori map facilitates the navigation process, and adds an extra layer of security and understanding. It allows the vehicle to better localize itself in the world by allowing it to focus its sensors and computing power only on moving objects. Figure parts are taken from [Kurdej 2015] with change in context and comments.*

P20, or dynamically by moving in the VIPA vehicle equipped with the Velodyne HDL 32E LiDAR.

This does not mean that the driverless vehicles localization is limited to the aforementioned techniques or sensors, other methods and resources may well exist. This is the case of a new method published quite recently [Xiang 2018], which uses the V2V [12] and V2R [13] connection to locate the vehicle in a wireless network environment (Figure 1.19). This relative positioning method is based on the synchronization between the vehicle, other vehicles, and the positioning beacons installed at the roadside. Whatever such methods always depends on the quality and reliability of the wireless network.

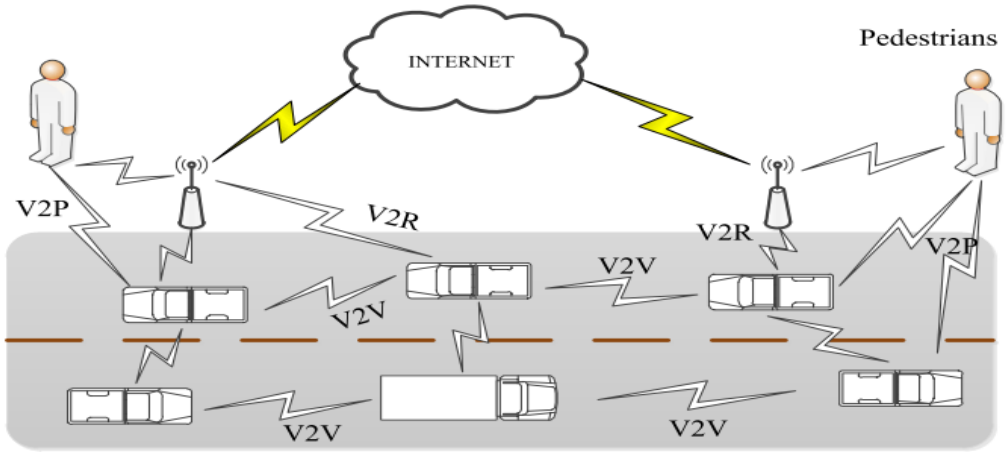


Figure 1.19: *Relative positioning method in wireless network environment [Xiang 2018].*

[12] **V2V**: **V**ehicle to **V**ehicle
[13] **V2R**: **V**ehicle to **R**oad

Table 1.4: *Methods of localization of different research vehicles, dependent on some a priori information, but not prior maps.*

Research vehicles	A priori method	SLAM-based method	year
Bundeswehr University of Munich's "VaMP"		✓	1994
Carnegies Mellon's "NavLab" in "no hands Across America"		✓	1995
University of Parama's "ARGO"		✓	1998
Carnegies Mellon's Urban Challenge entry, "Boss" (1st place)	a	✓	2007
Stanford's Urban Challenge entry, "Junior" (2nd place)	a	✓	2007
Virginia Tech's Urban Challenge entry, "Odin" (3rd place)	a	✓	2007
MIT's Urban Challenge entry, "Talos" (4th place)		✓	2007
Google's research vehicles	✓		2009
VisLab's "BRAiVE"		✓	2010
AutoNOMOS labs' "made in Germany"	✓		2010
Braunschweig University of Technology's "Leonie"	a	✓	2010
Oxford University's "Wildcat"	a	✓	2010
Nagoya and Nagasaki University's Open ZMP Robocar HV	✓		2012
LIVIC's "CARLLA"		✓	2011
Stanford's "Shelley"	✓		2012
Karlsruhe Institute of Technology's "Bertha Benz"	✓		2014
Audi's research vehicle		✓	2014
Toyota's research vehicle	✓		2015
Honda's research vehicle	✓		2015
Volvo's research vehicle	✓		2015
Ford's Hybrid Fusion research vehicle	✓		2015
General Motors' research vehicles	✓		2015
nuTonomy' s vehicles	✓		2016
Uber's vehicles	✓		2016

1.8 Digital maps: an Immersive Virtual World

Maps human use may return to pre-writing, where there could be simple painted-maps that were drawn before a man can write. Maps were, from the outset, trivial for man, who had always used them during his travels and trips. With this consideration, the map is as old as human civilization, but the oldest map that reached us is a Babylonian, painted on a mud plate of about 2300 BC.

Mapping has accompanied the evolution of humans, moving from drawing on pieces of stones, bones or mud to using paper, to reach today, and for more than a decade ago, digital-based maps. The methods of mapping have also changed to keep up with the level of human learning of writing, of symbols, of mathematics and of scales. To this end, man used with letters and symbols, shapes and fine images. Moreover, the forms in which maps are drawn has also passed from flat and circular forms to three-dimensional shapes offering the most accurate details.

The emergence of digital maps in recent years has been a breakthrough in man's use of maps in general. This digitalization has always helped the modern man reach the right directions while driving or walking. For self-

driving cars, the need for more accurate, complete and clear maps go far beyond basic turn-by-turn direction maps that we know today. Precision is very important; a few centimeters may lead to killing a person. Completion should be reflected even on the small details when creating these maps. Therefore, autonomous cars need HD [14] maps. HD maps refer to detailed static records of the environment (very high precision, usually of few centimeters (5–10 cm)). These maps may be enhanced with other information as street signs, lane markings, traffic signals, etc.

At present, the most successful autonomous driving projects use the prior maps for self-localization in the mapped environment [Vivacqua 2017]. Many companies, such as GoogleX of Alphabet Inc., Tesla, BMW, Uber, Honda, Toyota, Navya, and NuTonomy, use high definition street map for the localization of their own highly automated vehicles. These companies use specific vehicles to collect detailed data from the different sensors that equip these cars such as precise 3D point clouds, images and GPS information. The fusion of all this data creates detailed maps that will be stored on large databases, so that their vehicles can move autonomously on these mapped locations. Others can simply refer to specialized maps creation companies like HERE [15], TomTom [16] or GoogleX (Google Maps) and use their maps. Localization is performed by detecting the similarities between the map and the current sensor data, while moving objects detection is achieved by observing the discrepancies between the map and the current sensor data.

In this context, actors have emerged, from start-ups to internet giants, through research labs. For instance, Google's vehicle fleet has traveled more than 2.4 million kilometers autonomously using map-based localization [Van Brummelen 2018]. GoogleX has made a huge advance because it relies on Google Street View project maps which are today at their disposal and used to enrich the environmental knowledge of their vehicles (even though this was not designed for the autonomous vehicle project). Stanford has improved their maps with a priori list of traffic light locations so that its vehicle, Junior, can detect traffic lights in different lighting conditions [Levinson 2011]. In [Schreiber 2013] a highly accurate map coupled with curbs and road markings for precise and robust localization. This localization system uses an IMU and stereo camera only. The authors of [Tao 2013] propose a localization method based on the use of lane marking. The road lines are stored as polylines in an a priori map. Mercedes in 2014, in the Bertha's Benz Memorial Road commemoration. The 100 km autonomous driving route, between Mannheim and Pforzheim, was carried out using very precise contextual road-maps as well as a geo-referenced landmark database for the vehicle localization [Ziegler 2014]. Ford autonomous vehicle research group has invested in Civil Maps [17], a technology start-up with its own cameras, which works on the generation of HD Maps for autonomous vehicle purpose. The work described in [Häne 2017] uses a multi-camera system to generate accurate dense maps, and then visually localize the car with respect to those maps. This system has been used successfully on the autonomous cars of the V-Charge project, demonstrating the practical feasibility of this system. Moreover, the map can be used to anticipate road conditions. This is the case of [Anderson 2018], where the road information is extracted from the map in advance, and then the vehicle which is equipped with an active suspension, changes its driving state according to the state of the road.

Clearly, prior maps can only help to improve the localization system of self-driving cars. Beyond that, other forms and information can be added to these maps to further improve the driving system. These include semantic

[14] **HD**: **H**igh **D**efinition

[15] Here is an American company that was bought by a German consortium consisting of Daimler, BMW, and Audi.

[16] Source: https://www.tomtom.com/fr_fr/

[17] Source: https://civilmaps.com/

information, such as traffic signs, road lights, ground marking, etc. Information such as geometric infrastructure models and road surfaces can be used as well. In this thesis, we will study how localization can be improved by using such a priori maps.

1.9 Motivation and goal of this thesis

Whatever the subject, behind every research work, there is always a social context that motivates it. Given the current rates of road mortality, what has prompted us to participate in this research is our hope to participate, even with little, in saving human lives. Autonomous vehicle does not get tired on the road, nor is it distracted by a mobile phone. It has no emotions and does not drink alcohol. Other benefits that contribute in this direction are in the foreground the reduction of pollution. Many researchers [Guo 2017] predict that autonomous vehicles can decrease fuel usage and pollution. In a second plan: to have more free time, more space, and many more benefits [Guo 2017]. We hope that the commissioning of this technology will improve the quality of life of the whole society.

The main objective of this thesis is to develop tools allowing an accurate localization of an autonomous vehicle in a prior known environment. The known environment means that the prior map of the environment is available to the vehicle before it starts to move. Localization will be performed by map-matching between this 3D map and point clouds received as the vehicle moves.

From a technical point of view, the motivation for using map for localization research is three-fold. The first motivation is to provide an accurate representation of the environment for the vehicle to operate successfully in a fully autonomous manner. The map itself serves as an additional "virtual" sensor in the vehicle sensor system. It facilitates the navigation process, and adds an extra layer of security and understanding, as **Mark Jenkins** in **The HardWay** says "*Maps are essential. Planning a journey without a map is like building a house without drawings.*". The second motivation is using the Velodyne as the only sensor, yielding a system with minimal sensor requirements, because for an autonomous driving context, the lower price is highly relevant while keeping the accuracy. The final motivation is to provide 3D laser maps as the vehicle evolves in a three-dimensional environment. These maps can be useful for other agents, e.g. for human users.

1.10 Challenges Addressed and Overall Approach

With the development of highly automated driving vehicles, a need for new type of high-precision map,called HD map, has also appeared. These maps have got much higher requirements in terms of details and updates compared to the currently available navigation systems. Because the latter, which are, used for vehicle navigation systems or geographic information systems are not enough to meet the new requirements of intelligent vehicle systems such as autonomous driving. There are four main roadmap requirements for intelligent vehicle systems: centimeter accuracy, storage efficiency, sparse-to-dense matching, and map updates. However, as far as we know, no existing research has treated these four requirements at the same time.

This thesis focuses on the following core challenges to reach these requirements:

a. High accuracy In order to satisfy the first constraint, we have chosen a very powerful sensor, which is the Leica P20. This tool provides highly detailed point clouds with a precision that can reach the millimeter level. Figure 1.20 gives an idea of point clouds that can be obtained from this sensor. The Leica P20 has a field of view of 360° horizontal and 270° vertical. It is equipped with a digital camera for colorizing the points cloud by calibrated photo overlays. This serves to view the site for additional detail and provide photo rendering of the point cloud data. The Cyclone software allows assembling and manipulating several point clouds taken from different point of views.

Figure 1.20: *Example of scanning of the PAVIN experimentation platform obtained with the Leica P20.*

b. Storage efficiency A map for autonomous driving requires very high precision, but such a precise map requies a large amount of storage space. In our localization approach, a map obtained with a dense and precise scanning is used. However, this map quickly becomes difficult to handle because of its huge amount of data, which can reach billions of points. A data reduction stage is inevitable in order to exploit the richness of all the information carried by this map.

c. Sparse-to-dense matching In order to achieve a system that works in real time, we opted to equip the vehicle with a high-frequency sensor, which is the Velodyne HD 32E. The localization is accomplished by pairing (scan matching) between the available map and the online data received as the vehicle move. The downside is that sensor generates sparse data. Up to now, we find in the literature only the alignment algorithms, which deal with point clouds coming from the same sensor. Nevertheless, in the exemption of the embedded sensors compatible with those used in the elaboration of the prior map, it is not possible to benefit from all the produced maps with the maximum of the desired precision. This prompted us to explore the scanning matching methods, and more specifically the “dense to sparse or sparse to dense matching”, as our localization strategy is done by pairing a sparse data with dense map data.

d. Map update Having a compact, efficient and re-usable mapping system for autonomous navigation is of the paramount importance. We cannot only rely on a single turn built environment representation as the latter changes,

with time, over seasons and years due to ongoing natural and man-made occurrences. Therefore, updating the map so that it serves its true purpose over the longevity is of utmost importance. For this reason, updating the reference map was our last challenge.

1.11 Thesis outline

This thesis is organized in three parts: BACKGROUND, POINT CLOUD REGISTRATION, MAPPING AND LOCALIZATION. Each part is structured and organized as follows:

BACKGROUND: contains two chapters (Related works: Chapter 2) and (Fundamentals: Chapter 3).

Chapter 2: critical analysis and review of the literature related to mapping techniques and localization techniques. Analysis of relations among the approaches in literature and discussion of their limitations.

Chapter 3: recall of some basic notions necessary to understand the topics discussed in this thesis.

POINT CLOUD REGISTRATION: contains two chapters (Registration, Chapter 4) and (CICP: Cluster Iterative Closest Point, Chapter 5).

Chapter 4: both bibliographical and experimental study of the Iterative closest Point (ICP) method. The goal is to understand this method, which is represented as one of the key methods in robotics mapping and localization field.

Chapter 5: Novel approach for sparse to dense point cloud registration exploiting normals differently. The traditional ICP pipeline is modified to accommodate a smarter way of surface patch correspondence.

MAPPING AND LOCALIZATION: contains three chapters (Creating of the Reference Map: Chapter 6), (Points Cloud Reduction: Chapter 7), and (Localization within a Prior Map: Chapter 8).

Chapter 6: reference map creation either statically or dynamically, in an incremental manner, is discussed. This step consists in estimating the 3D pose connecting the scans.

Chapter 7: original approach to sample number of points based on both the use of color information and the geometry of the scene.

Chapter 8: 3D based localization using prior map for an autonomous vehicle equipped with a sparse sensor.

Part I

BACKGROUND

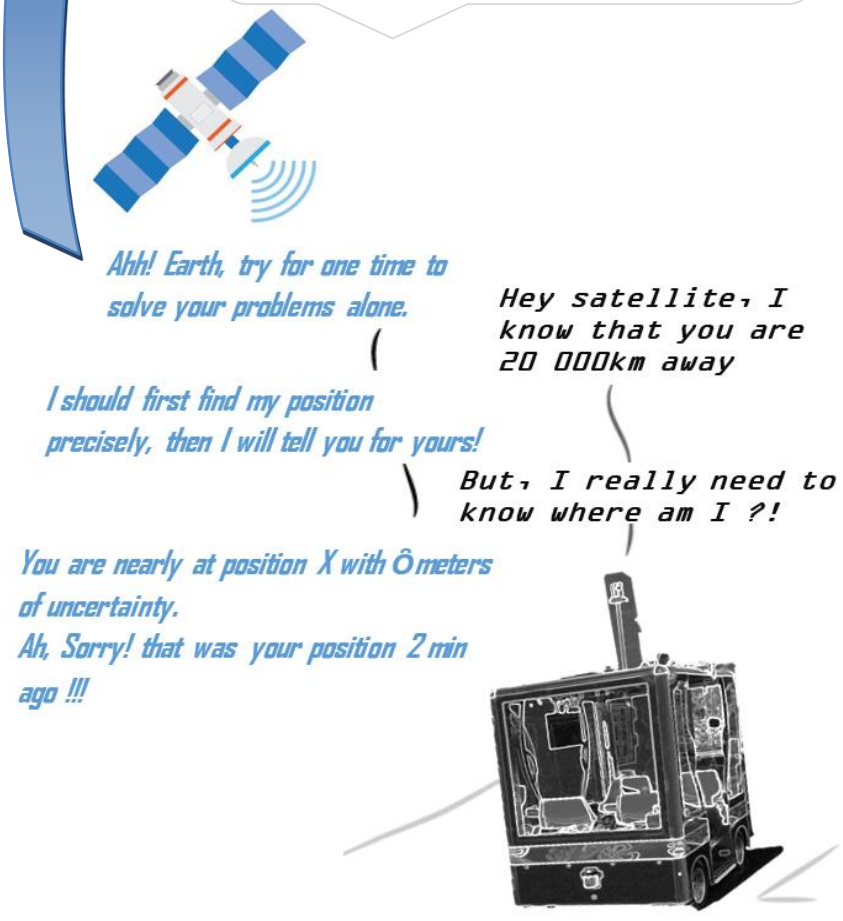
The objective of this part is to describe the current state of research in the areas of localization and mapping. In addition to the theoretical context, we briefly present the mathematical tools used in this thesis.

CHAPTER 2 *introduces the reader to localization and mapping domains, and offers a critical analysis and review of the literature related to these two areas. An analysis of relations among the approaches in literature and discussion of their limitations is also provided.*

CHAPTER 3 *recalls some basic notions necessary to understand the topics discussed in this thesis. In particular, it deals with registration techniques and clustering methods. It gives an overview of mathematical theories used for that and for the purpose of autonomous driving localization.*

Chapter II

Related works



CHAPTER 2

Related works

Contents

When a vehicle is driving autonomously, it is imperative that all safety conditions are met. Among all these conditions, we evoke the accurate and precise localization. In fact, localizing and understanding the position of an autonomous vehicle is essential for it to make the right decisions in terms of path planning and obstacle avoidance. This chapter provides the necessary literature review in these lines of work. A brief review of principal sensors used in autonomous driving localization will be provided followed by a detailed review of mapping strategies. Subsequently, localization approaches are reviewed. We propose an original classification of localization methods, by grouping them into four categories, depending on the surrounding environment and the robot's knowledge of this environment. The idea behind this classification is to better understand the localization problem as a whole.

2.1 Introduction

One of the most important capabilities of an autonomous vehicle is to be able to accurately determine its positions at any time. Indeed, a precise localization is essential for any autonomous system to operate successfully in a fully autonomous manner. This is necessary not only for fundamental tasks like navigation and path planning, but also to complete other tasks such as detection and obstacle avoiding. A common approach to the autonomous vehicle is to use a detailed prior map for localization purpose [Wolcott 2015, Caselitz 2016]. Just as a human being, who is able to use a map, explores it and combines it with his visual inputs to locate effectively, the vehicle uses prior made maps that combine it with local data of its onboard sensors to find its position in the global map. A priori map facilitates the navigation process and adds an extra layer of security and understanding. It helps the autonomous car to localize itself in the world with a greater degree of accuracy, by providing a crucial context, allowing it to focus its sensors and computing power on moving objects such as pedestrian, cyclists, and cars. This chapter provides a broad coverage of the approaches and means involved in equipping mobile robots with the localization and mapping capabilities. The aim here is not to give a very detailed state of the art in the field; this may take a few months to identify the whole field; this may also be available in many reviews and theses cited in the bibliography. We are contented here to cite only the main existing approaches. This is in order to give a theoretical basis for our research and choices made during this thesis.

2.2 Perception

The perception is closely related to localization and mapping. For example, an autonomous vehicle needs to know its position in the world via localization, to better estimate, which objects are in its surrounding environment [Van Brummelen 2018]. This relationship works both ways. In other words, if the localization and mapping module identifies a measurement, it will request the perception module to correct this measure. As well as, if the perception module senses its surrounding, it will request the localization and mapping module in order to update it accordingly [Van Brummelen 2018]. A practical example could be that if a self-driving car detects by localization and mapping that it is approaching a traffic light, it should give priority to the perception module, in this case, the vision module in order to detect and identify the green light to continue its path (which can only be done if the vehicle has localized itself). Alternatively, it could go in the opposite direction if the car passes over the middle line, it is likely that it will get out of its way, and that the localization algorithms must be updated accordingly. Recent advances in perception sensors, especially 3D, contribute decisively in improving the localization task for the autonomous car. In the following, we will address the different means used for the perception of the environment and for the localization of an autonomous system.

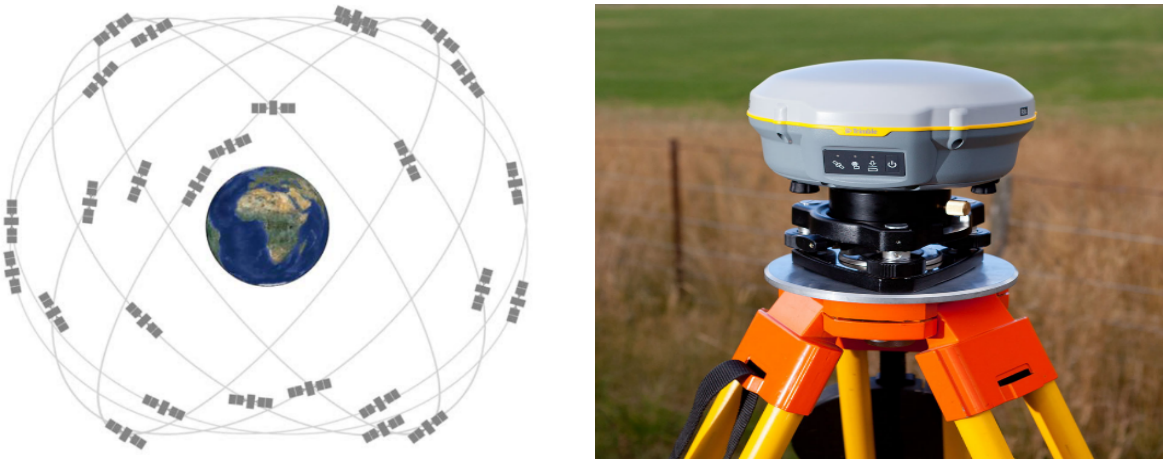
2.2.1 Sensors for autonomous driving

Up until now, each vehicle has an average of 60 to 100 onboard sensors and the high-end versions have about 150 sensors. Because cars are becoming more and more intelligent and because there is a need for robust and redundant information, the number of sensors should exceed the bar of 200 sensors per car in 2020 [Chirca 2016]. Similarly, the field of autonomous vehicles has experienced significant growth during this last decade due to advances in

sensing technologies and computing power. These sensors, which are used to monitor the internal state of the vehicle and perceive the surrounding world, are grouped under exteroceptive and proprioceptive sensors, which we will detail in what follows. However, this subsection is not intended to be an exhaustive list of all existing sensors used for self-driving vehicle localization. Depending on the case of application, there are multitudes of possible solutions. We will, therefore concentrate here on the most common systems in the context of the localization of autonomous systems, i.e. allowing sufficient precision for its control.

2.2.1.1 GPS

In recent years, the ultimate sensor for localizing a vehicle is the GPS. It provides a geo-referenced position from the order of a few centimeters to few meters proportionately to its price. This position is obtained by measuring the travel time of electromagnetic waves between a constellation of 24 satellites, evolving in 6 orbits around the earth at an average altitude of 20 200 km (Figure 2.1(a)), and a receiver (Figure 2.1(a)). Mainly, the receiver receives signals from at least four satellites. Because it must determine four data which are the three positions of the receiver in a reference frame, and the shift of its clock relative to the GPS time of the satellites. GPS-based localization system can only provide limited accuracy, in the range of several meters (5 to 20 meters). Although it is highly spread, it suffers from strong limitations for localization of autonomous systems. Indeed, it is impossible to envisage the localization of an autonomous vehicle on a traffic lane that measures an average of 2 meters wide with such precision [Levinson 2007, Vivacqua 2017]. Moreover, only the position is available and the orientation remains unknown. Additionally, in dense urban environments, the multiple reflections of GPS signals from buildings facades disturb the signal quality and therefore the location accuracy. Furthermore, this system does not work in indoor environments such as car parks, tunnels, etc. [Brubaker 2016].



(a) Satellites constellation

(b) GNSS Trimble: An example of a GPS terminal that allows to receive the signals emitted by the satellites, and which converts this data into geographical positions for the GPS terminal

Figure 2.1: *GPS geolocation.*

Different solutions have emerged to improve the satellite localization, such as the Differential GPS (DGPS)

and Real-Time Kinematic (RTK) systems, which use two receivers to partially overcome the disturbances due to the propagation in the atmospheric layers. These systems provide centimeter-level details, but for a much larger cost. In addition, they are still sensitive to signal masking, and multiple reflections. This means reduced accuracy anywhere the sky is blocked by obstructions like skyscrapers, dense tree regions, tunnels or in underground places, etc. All these deficiencies make these systems unsuitable to handle critical tasks such as autonomous systems localization.

2.2.1.2 Proprioceptive sensors

To overcome GPS problems, some approaches use proprioceptive [1] sensors such as wheel encoders, gyro sensors, and IMU. These sensors determine the current position of the vehicle by integrating velocity and acceleration measurements from an initial position.

a. Wheel odometry

Odometry consists of measuring the distance traveled by the mobile system based on the rotations of one of the non-driving wheels in general. This distance is to be determined in linear and angular displacement:

- due to the incremental characteristic of this localization, measurement errors accumulate over time and cause drift of the robot's estimated position,
- this technique is limited to wheeled ground vehicles,
- errors due to sliding.

In order to improve the odometry performance, it is possible to equip the robot with non-propulsive wheels that will be used for measuring the displacement in order to limit the sliding errors.

b. Gyro sensors

Also known as angular velocity sensors or gyroscope, are useful sensors in mobile localization. These sensors are motion sensors that provide angular velocity information relative to an inertial reference frame. Gyro sensors come in a variety of types; we can find mechanical, optical, and vibration gyro sensors.

c. Accelerometers

Just like gyro sensors, the accelerometers are also very useful sensors in mobile mapping and localization. Always with the aim of tracking the movements of a vehicle, the accelerometer allows to measure the acceleration in a given direction, and by integrating this measure, it is easy to find out the speed of the vehicle in this direction. Several errors affect the measurements of these sensors and thus generate drifts of the speed and position estimates. The errors are mainly due to the multiple integrations needed to calculate the trajectory.

[1] Proprioceptive: because it allows informing about the state of an intrinsic element of the robot

d. Inertial unit

An inertial unit, which generates heading, attitude and position information, uses three gyroscopes and three accelerometers. The device is responsible for integrating the data of its various sensors to have the speed and the angles when the vehicle is moving. It is a very popular proprioceptive sensor; in general, mobile vehicles are equipped with only an inertial unit and an odometer, since the accelerometers and gyroscopes are already integrated into the inertial unit. Inertial sensor measurements are potentially very reliable because they do not depend on the environment. Unfortunately, these sensors are noisy, which leads to a drift of the position over time. Figure 2.2 shows 2 examples of commercial inertial units.

(a) (b) (c)

Figure 2.2: *Examples of proprioceptive sensors, from left to right: LandINS Inertial Unit from iXSea, an odometer, an optical gyroscope.*

Since each speed and acceleration measure is tainted by error, this leads to significant cumulated errors on the integration process. In addition, these sensors deliver false measurements if the wheels of the vehicle slide on the road [Yousif 2016]. All these reasons and others, make the localization of autonomous vehicles only by odometer-type sensors, a solution difficult to envisage. Even with a GPS and IMU fusion, still cannot completely provide a precise localization in a dense urban environment [Vivacqua 2017]. Table 2.1 presents a comparison of the characteristics and performance of a GNSS positioning system and an inertial navigation system.

Table 2.1: *Advantages and drawbacks of both GNSS and INS positioning.*

	GNSS positioning system	**inertial navigation system**
Advantages	- Does not diverge in the long-term - World wide Coverage - Low cost	- Autonomy (insensitive to interference) - Information in translation and rotation - Accurate in the short term - Independent to external conditions
drawbacks	- Sensitivity to the environment - Not very accurate in the short term - No information on the attitude of the mobile	- Drift over time - accuracy depends on the price

2.2.1.3 Exteroceptive sensors

An exteroceptive sensor is a sensor that provides information about the environment, the external state of the mobile vehicle, or more generally the robot. This class of sensors gives measurements on what is external to the vehicle. These direct measurements, rich in a quantity of usable information, allow computing the position of the vehicle in a precise way. Similar to the early navigators using stars to localize themselves and find their way, the vehicle uses the information provided by exteroceptive sensors (features, landmark, etc.) to localize itself. Among all the exteroceptive sensors that an autonomous vehicle can dispose of, we find the vision sensors, radar, and laser-based sensors.

a. Radar

The Radar [2] is a measuring instrument that indicates the presence of a distant object, its size, its speed, and its direction. It is the most used automotive sensor for object detection and tracking, due to its economical price and its effectiveness under extreme weather conditions (rain, snow). The downside is its low resolution. There are different categories depending on their opening angle and their range. Radars with a large opening angle have a short range and vice-versa. The information that gathers is like points cloud. It sends waves that bounce off the obstacles encountered and return to it in the form of echoes. An associated software allows determining the distance that separates this object from the vehicle and sometimes identifying this object. Among its standard applications, in the automotive industry, is the Automatic Cruise Control (ACC) or the Emergency Brake Assist (EBA) systems. Figure 2.3 shows the Smartmicro automotive radar as an example of these sensors.

(a) (b) (c)

Figure 2.3: *Some exteroceptive sensors, from left to right: Smartmicro Automotive Radar; monocular camera; BISEEMOS: the Institut Pascal stereoscopic cameras.*

b. Ultrasonic

It is an object detection sensor, which emits ultrasonic sound waves and detects their return in order to determine the distance. It is mainly used for very short ranges of a few tens of centimeters. It alerts the driver before the collision during the parking maneuvers.

[2] **Radar**: **R**adio **D**etection **A**nd **R**anging

c. Vision-based sensors

These sensors measure the light intensity reflected by the objects of the environment. They are very rich sensors, they bring in fact both photometric and geometric information of the observed scene. The different types of cameras (color or black and white, outlook or panoramic, sensitive in the visible or infrared spectrum, etc.) can provide localization measurements by detecting identifiable elements in the environment, by estimating a basic displacement, or by building a map of the environment. There are different types of cameras:

Monocular cameras The reconstruction of the scene from a single camera is performed by SfM [3] techniques that require a camera movement in addition to tracking the various keypoints along the movement. Moreover, with a single camera, the result of the SfM algorithm is a cloud of 3D points, but the distances between points are calculated to a scale factor close. This drawback is easily overcome by fusion with the odometer information that allows calculating the scaling factor.

Stereoscopic cameras This sensor consists of a pair of classical cameras, located at a fixed and known distance, whose field of vision overlaps. This stereoscopic pair allows perceiving depth through two images taken from two staggered viewpoints, even when the camera does not move. This enables the 3D reconstruction of the scene from these pairs of images, and thus obtaining clouds of 3D points. Stereo cameras are widely studied from the computer vision community [Fang 2017]. Figure 2.3 on the right show the BISEEMOS stereoscopic cameras, which designed by the Dream team of Institut Pascal for stereo vision purposes. Its architecture has been designed for dedicated parallel algorithms by using a high-performance FPGA.

d. Laser-based sensors

In recent decades, LiDAR [4] has emerged as a powerful technological solution in a variety of areas. This boom has been motivated by the need for efficient, safer, more accurate and faster modeling that allows an exhaustive analysis of the scene [Puttonen 2013]. There are several types of LiDARs; the one that interests us in this thesis is Laser Range Finder, which allows measuring the distance with objects. That is mean, which allows obtaining the location of these objects in the environment.

3D laser scanning technologies 3D LiDARs are commercially available in many varieties, and all of them work on the same basic principle [Puttonen 2013]. They emit a pulse and detect its reflection in order to probe the object or the environment. The time of flight (TOF) class offers a greater range and good accuracy and is therefore implemented in large part of scanners [Puttonen 2013]. This technology is extremely advanced but it is designed to be easy to use. The LiDAR emits a fast pulse or continuous laser beam, and detects only one point at a time in the direction in which it is pointed. For this, the device scans its entire field of view point by point and must change its direction of view for each measurement. This view direction can be changed either by rotating the scanner itself or

[3] **SfM**: **S**tructure **f**rom **M**otion
[4] **LiDAR**: **Li**ght **D**etection **A**nd **R**anging

by using a system of rotating mirrors allowing the laser beam to scan a plane. The result is a systematic sweeping of the beam over the area to be scanned.

When the laser light scattered by the surface of the object and reflected back in different directions, some energy back to the LiDAR. Based on the time needed to the reflected laser beam to return, distance from the scanner to the object will be calculated. However, there is more in 3D scanning than just measuring distances. For each distance measurement, additional critical data are recorded, including the horizontal angle corresponding to the rotation of the laser and the vertical angle corresponding to the rotation of the moving mirror. The scanner automatically combines these data, to compute a 3D coordinate (x, y, z) for each point in the LiDAR's frame, using a transformation from the spherical coordinate system to a Cartesian coordinate system. The resulting scan is a set of 3D coordinate measurements; it is the detailed 3D representation of the scene, often called points cloud [Heritage 2009].

To add realistic textures or colors to the scans, matching photos can be taken. Either using a camera that is built into the scanner or using an external camera. Once the camera is calibrated, the color information is automatically readjusted on scanned data, giving rise to colorized point clouds.

For mobile localization and mapping, several types of LiDAR sensors exist. Figure 2.4 presents some examples of used sensors:

- The Hokuyo and Sick sensors are 2D LiDARs, which scan the environment according to a plane, at a high frequency of several hundred of Hertz. This gives data in 2 dimensions. In order to be able to perceive the environment in 3D, some researchers have developed systems to oscillate single-layer LiDARs. In the case of relatively slow movements, these solutions can be very effective as illustrated by the work of Zhang and Singh [Zhang 2014], which use a Hokuyo laser scanner driven by a motor for rotational motion, and an encoder that measures the rotation angle. The laser scanner has a field of view of $180°$ with a resolution of $0.25°$. [Lin 2013] present a method of calibrating a specific system to obtain multi-layer information from a 2D LiDAR mounted on pan-tilt unit. These sensors are very popular because they are generally low-priced and lightweight.

(a) 2D Hokuyo sensor (UTM30-LX) (b) 2D SICK sensor (LMS511), (c) 3D Velodyne LiDAR (HDL-64-E), (HDL-32-E), (d) VPL-16-E, (e) 3D Leica P20 LiDAR

Figure 2.4: *Some examples of LiDAR sensors used in mobile robotics.*

- Multi-layer solutions also exist. Unlike 2D sensors that are single-layer, this sensor is a 3D LiDAR, which allows direct scanning of the environment with a 360° horizontal field of view, and a vertical opening of several degrees different depending on the model and the number of layers. Technically, they consist of several transmitters/receivers assembled on the same axis of rotation of azimuth, but shifted in elevation. The well-known in the research field are those of the Velodyne [5] company. In this thesis, we use one of these sensors, which is the Velodyne HDL 32-E, because it represents the most accurate sensors for real-time mapping [Choi 2014, Zhang 2015]. This sensor produces 3D scans by rotating a set of 32 beams around its vertical axis at 10 Hz. It generates up to 700000 points per second or 2200 points per laser beam, in a range of 1 to 70 meters. This sensor provides horizontally an angular resolution of approximately 0.16 degrees with a field of view (FOV) of 360 degrees. Its vertical field of view ranges from -30.67 to $+10.67$ degrees with an angular resolution of 1.33 degree. Its measuring accuracy is generally less than 2 cm. This type of sensor is much more expensive than 2D sensors, even if their prices are decreasing from year to year. For instance, Robosense [6] recently introduced its low-cost LiDAR RS-LiDAR for 7000 $. [Wang 2017] paper presents its comparison with the VPL-16 of Velodyne. However, despite the high price of these sensors, they produce an extremely accurate depth information.

  Multi-layer LiDARs has an important role in autonomous cars and their configuration, such as the location of each LiDAR, can influence the entire autonomous driving system. [Levinson 2011] was the first dealing with LiDAR's mounting location on the vehicle. They use the approach presented in [Levinson 2010b] to discover the optimal 6-DOF sensor pose. [Gordon 2013] use a reference model generated by terrestrial laser scanning to calibrate a multi-beam laser system mounted on moving platforms. Recently, [Mou 2018] proposes a generalized optimal Multi-layer LiDAR configuration which take into consideration the sparsity of these sensors. Figure 2.5 gives examples of some LiDAR configurations used by different self-driving cars.

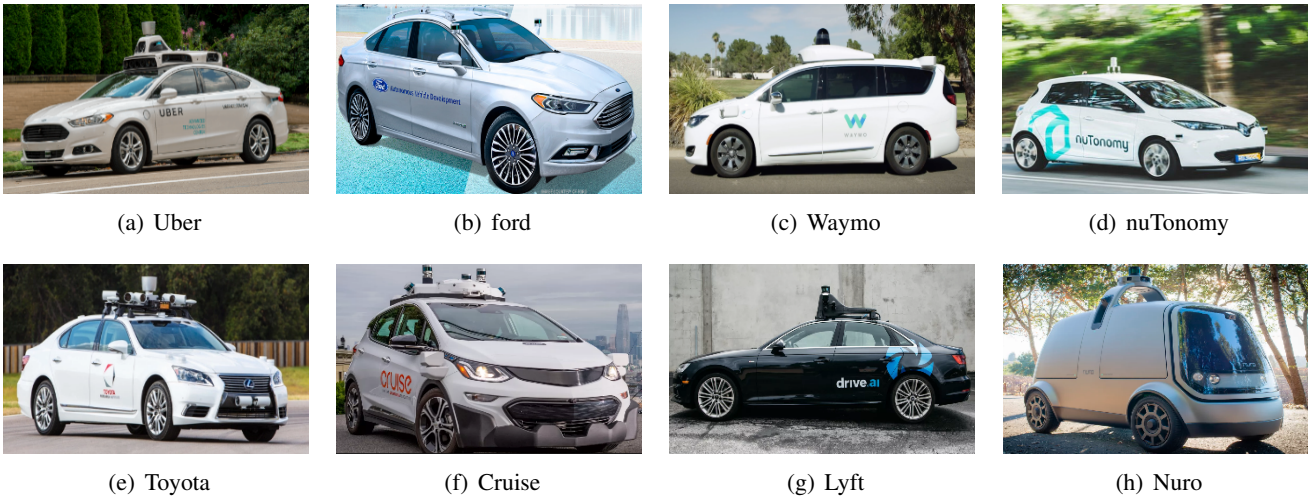
- The other types of LiDARs sensors are terrestrial laser scanning (TLS). This technology has been quickly adopted throughout the world for capturing three-dimensional survey data in a variety of industrial applications [Ford 2011, Biskup 2007]. They have a fast refresh rate and a large range such as the Leica P20, which we will use it in this work. This sensor consists of a single-point rangefinder and a two-axis motorized camera.

Other sensors such as the Kinect and Time-of-Flight Camera (TOF) provide depth measurement. However, the former is very disturbed in the outdoor environment, while the second remains marginal by the scientific community and suffers from its weak resolution.

2.2.2 Summary on sensors

In the previous section, we have reviewed the various sensors used for autonomous driving. Despite the fact that cameras are inexpensive and energy-efficient, these sensors are still suffering from environmental variations (their sensitivities to lighting conditions, noise, geometric illusions, etc.) or common known factors (lack of

[5] Source: https://velodyneliar.com/
[6] Source: https://www.robosense.ai/



(a) Uber (b) ford (c) Waymo (d) nuTonomy

(e) Toyota (f) Cruise (g) Lyft (h) Nuro

Figure 2.5: *Different Multi-layer Lidar configurations.*

overlap between images, texture-less surfaces, motion blur, etc.) [Sun 2018]. It has appeared that under very bright conditions (such as after sunrise and before sunset); it is apparently possible for some vision-implementations to not identify light objects against bright skies. This would have been a factor in the Tesla autopilot accident in May 2016 in Florida, which was the first death in a self-driving car. The onboard cameras of a Tesla Model S (level 2 autonomy) vehicle were not able to detect the side of a white truck due to the sunlight, and collided with it [Paul 2018]. Moreover, cameras are less useful for very close proximity assessment than they are for further distances. In addition, the image flow management remains a delicate and time-consuming operation. Another major drawback is that this type of sensor can only provide 2D information. In other words, any depth information is lost, unless using several sensors (stereovision) and the implication of very costly triangulation techniques to extract this type of measurement.

Ultrasonic sensors have very poor range, but are excellent for very near range. It alerts the driver before the collision during the parking maneuvers. We do not think that it can have any other use for the self-driving car.

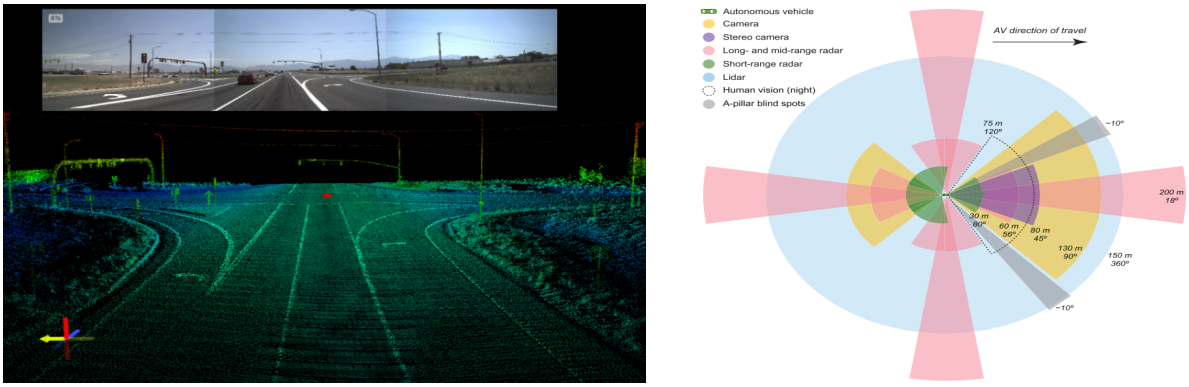
Table 2.2: *Comparison of exteroceptive sensors [Fridman 2018].*

	Resolution	Range	Proximity detection	Work in dark	Works in bright	Works in snow, fog, rain	Detect speed	Provide color/contrast	Sensor size	Sensor cost
Radar	✓✓	✓✓✓✓	✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	-	✓✓✓✓✓	✓✓✓✓✓
Ultrasonic	✓	–	✓✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	✓✓✓✓✓	-	-	✓✓✓✓✓	✓✓✓✓✓
camera	✓✓✓✓✓	✓✓✓✓	✓✓	–	✓	✓	✓	✓✓✓✓✓	✓✓✓✓	✓✓✓✓✓
LiDAR	✓✓✓✓	✓✓✓✓	✓✓	✓✓✓✓✓	✓✓✓✓✓	✓✓	✓✓✓✓	–	✓	✓
TLS	✓✓✓✓✓	✓✓✓✓✓	✓ ✓	✓✓✓✓✓	✓✓✓✓✓	✓✓	✓✓✓✓	-	✓	✓

In the case of RADARs, some of the standard vehicles already have them. Depending on their opening angle and their range, the RADAR is classified in long, Medium and short range. They have good range, but poorer resolution than other sensors. Also, they are less effective at very short distances.

Alternatively, LiDAR allows direct access to 3D information (3d point). Because it has a wide field of view,

and a long range, it is considered by many to be the most important of self-driving car sensors. However, LiDAR is better until significant atmospheric murkiness occurs with fog, snow, or heavy rain, but degrades under those conditions. Figure 2.7 shows a comparison between camera and LiDAR. From this comparison, LiDAR performs well than a camera for the full autonomy. The open question is whether LiDAR in the future of this technology can become cheap and its range can increase, because then LiDAR can win out. A lot of developments with a lot of startup LiDAR companies are promising to decrease the cost and increase the range of these sensors. One of the manufacturers of these sensors, which is Quanergy [7], has demonstrated a solid-state LiDAR system that should have a range of 150 meters, a cost of 250 dollars and an adequate resolution. The price and the expected performance will allow it to be a very competitive sensor if it reaches the production. Many well-known companies like Waymo, Toyota, Ford, and others like showed in Figure 2.5 use LiDAR as the primary and the dominant sensor. It is not by chance that these companies have chosen it for their self-driving cars. Of course, still other companies did not choose it because of its cost, such as Tesla, which prefers cameras. Nevertheless, for LiDAR, the noise associated with each distance measurement is independent of the distance and the lighting conditions, which is the opposite of the camera [Deschaud 2018]. This represents a great advantage for LiDAR with respect to cameras.



(a) Lidar image and corresponding photographic image of different sensors used in autonomous cars

(b) Estimates of coverage areas (field of view)

Figure 2.6: *LiDAR Superiority compared to other self-driving car sensors [Schoettle 2017].*

Beyond all LiDAR's advantages mentioned above, its high accuracy and high refresh rate make this sensor indispensable tools for localization in the field of autonomous vehicles. Indeed, it is able to discern a high level of detail (shape, size, etc.), especially for nearby objects and lane markings, as shown in Figure 2.6(a). Panoramic LiDARs allow the autonomous vehicle to "see" in all directions (Figure 2.6(b)). Nothing escapes neither pedestrians, nor cyclists, nor other vehicles. These powerful capabilities go far beyond the ability of other sensors, for all these reasons, it was the technological choice of this thesis.

Finally, each sensor has its strengths and its weaknesses as summarized in Table 2.2. We are aware that a single representation of reality from multiple sensors is necessary to avoid false positives and false negatives. In the work of this thesis, we are relying on the Velodyne HDL 32-E LiDAR as a unique and dominant sensor, along with the prospect of taking advantage of the benefits of other sensors in order to fill its gap, such as benefiting from the

[7] source: http://quanergy.com/

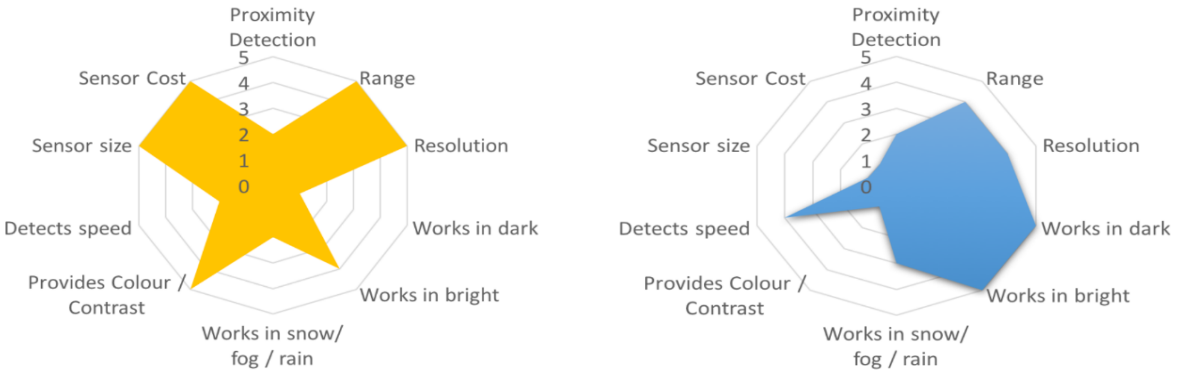


Figure 2.7: *Camera vs LiDAR: on these two figures, the main features for autonomous driving are represented on the right of each sub-figure, while the minor features are on the left. From this comparison, the LiDAR performs well than the camera for the full autonomy [Fridman 2018].*

information RGB from camera, odometry information or why not fusion with a RADAR for operation under all weather conditions.

2.3 Mapping

Maps are a vital element for autonomous systems; they facilitate the navigation process and add an extra layer of security and understanding. The vehicle uses its onboard sensors to compare what it "sees" at a given moment with what is stored in its memory. A priori maps allow the vehicle to better localize itself in the world by allowing it to focus its sensors and computing power only on moving objects. In this way, the vehicle has an idea about what should happen, it could see what is actually happening in real time, and therefore can make a judgment on what to do. Since the environment in which the vehicle operates is three-dimensional, the interest and demand for 3D mapping have been greatly increased in recent years. This is mainly due to the improvement of acquisition systems on the one hand and the growth of the range of potential applications on the other hand. Currently, 3D data can be obtained using two technologies: photogrammetry and laser scanning. The laser technology provides direct 3D data, while photogrammetry reconstructs 3D information by techniques such as triangulation from several images of the area under exploration. The advantage of direct 3D data acquisition makes the laser scanner popular for mapping the environment either indoors or outdoors [Caselitz 2016]. Moreover, localization can be done at different timescale compared to the mapping, which requires that the process of localization should be robust to the environment change (such as the lighting change) [Caselitz 2016]. For all these reasons, we focus in this search on laser technology.

2.3.1 3D data acquisition techniques

Several 3D data acquisition techniques used to create maps. These techniques can use a single sensor as the case of Mars rovers [Maimone 2007, Nuchter 2004], or a combination of sensors as the case of [Hentschel 2008] which

uses in addition to LiDAR, a GPS and an IMU to reference the data. Two main categories are stand out: fixed acquisition techniques and mobile acquisition techniques.

2.3.1.1 Static acquisition techniques

Static acquisition techniques use stationary systems for mapping the environment; usually they use Terrestrial Laser Scanners (TLS) (Figure 2.7). Static techniques allow having very precise maps, with a fine level of detail. Since the acquisition system does not move during scanning, this significantly reduces the noise caused by the mobility and the delay of reflected laser beams, which increases the accuracy. However, the acquisition time is generally very important. This is because the acquisition system is usually moved by hand. Moreover, the immobility of the scanner allows carrying out 3D scans with a high accuracy, which further increases the acquisition time.

Figure 2.8: *The 3D terrestrial laser scanner Leica P20 of Institut Pascal, as an example of static acquisition technique.*

TLS, as shown in Figure 2.8, can capture data from its front view and for a limit range. Thus, to scan a complete 3D object or a large area, the scanning process should be repeated from several locations and diverse angles, each expressed in its local frame. The resulting point clouds are assembled and merged into one consistent point cloud, through an operation called “Registration”, which we will see in detail in Chapter 4.

2.3.1.2 Mobile Acquisition techniques

These techniques are based on moving platforms (aircraft, train, car, etc.), which requires a georeferencing system to localize the data. The latter is necessary, because during the platform displacement, the acquisition system that is often composed of one or more LiDARs and/or cameras, provides data at a certain time interval (which depends on the sensor frequency). However, these data have no information on their positions at the moment of the acquisition, and therefore on their referencing. The referencing allow making the data usable by relating each point cloud or image to a ground system of geographic coordinates, which makes their registration performable. There are two types of referencing: local with respect to an arbitrarily chosen reference, usually the starting acquisition point, or global with respect to a world coordinate frame. Figure 2.9 shows several mobile acquisition platforms.

(a) Yamaha RMAX drone [Recchiuto 2017] (b) STEREOPOLIS II [Paparoditis 2012] (c) Segbot robot [Montemerlo 2006] (d) Portable backpack [Nüchter 2015]

Figure 2.9: *Some platforms used in mobile acquisition techniques.*

Mobile acquisition techniques include two approaches, “Stop-and-Go” and “On-Drive”. The first one, “Stop-and-Go” is quite similar to static acquisition techniques, the only difference being that the acquisition system is mounted on a vehicle and that it moves more quickly from scan station to another scan station. Regarding the “on-Drive” approach, the acquisition is performed during the movement of the vehicle. The displacement is considered as one of the scanning directions.

These techniques allow acquiring large volumes of data in a relatively short time, compared to static ones. However, these techniques currently respond only partially to the needs on the quality of the mapping results, particularly on the accuracy of the data.

2.3.2 Overview of Map Representation

An accurate representation of the environment is essential for any vehicle to operate successfully in a fully autonomous manner. Depending on their internal representation, there are three different types of maps:

2.3.2.1 Topological maps

The topological maps allow representing the environment of the vehicle by a graph. The graph’s nodes correspond to places (the positions the vehicle can reach). The edges linking the nodes mark the possibility for the vehicle

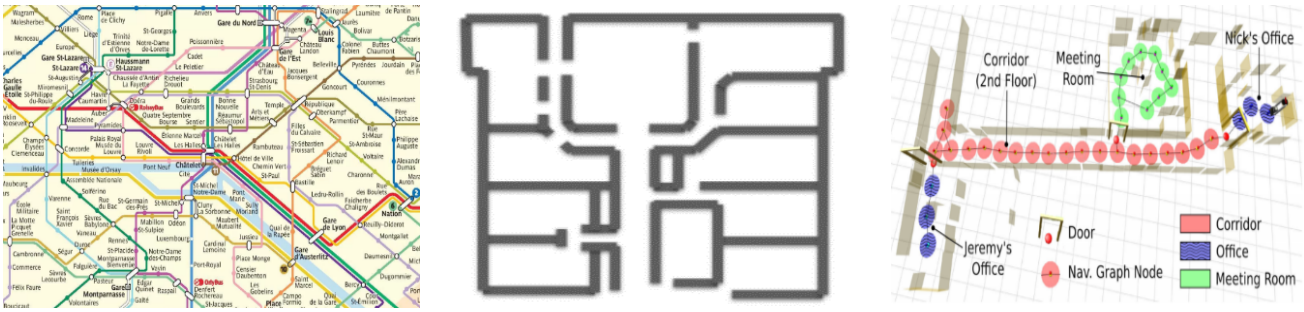
to pass directly from one place to another and memorize in general the way to achieve this passage. Topological maps are adapted to the representation of large spaces while keeping a sufficient level of abstraction. Very common examples of a topological map is the Subway map (Figure 2.10(a)), or road graphs.

2.3.2.2 Metric maps

In a metric map (Figure 2.10(b)), the environment is represented by a set of objects that are associated with positions in a metric space. Objects stored in these maps correspond to the obstacles that the vehicle will encounter in its environment. The map of the environment then corresponds to the space in which the robot can move. The main advantage of metric maps is to be able to represent the entire environment, not a small subset of places as the topological maps do. This complete representation allows to accurately and continuously estimate the position of the robot over its entire environment. Moreover, this complete representation is not limited to the physically explored positions, but extends to all areas that the robot has been able to perceive from the places it has visited. However, these representations are sensitive to noise.

2.3.2.3 Hybrid maps

They are mixed maps that contain both topological and metric information (Figures 2.10(c)). This in order to benefit from the advantages of both representations. Like the idea of connecting the different local maps with a topological graph. This allows to maintain the easy aspect of long-term navigation, with limitation in drifts accumulation [Lim 2012].



(a) Topological map (b) metric map (c) hybrid map

Figure 2.10: *examples of different map representations [Pronobis 2011, Filliat 2011].*

2.3.3 Maps for autonomous driving

Prior maps are of paramount importance for the autonomous driving. They can guide cars when the lane markings are erased or covered by snow or an object is blocking the car's view. With prior maps, car focus more on moving obstacles, which reduces the amount of software processes and allowing the vehicle to anticipate and avoid tricky situations. Moreover, while the sensors of the autonomous cars can detect a distance of approximately 50 to 70

meters, a car driving at high speed (as on a highway) can traverse this distance in less than 5 seconds, which leaves it a detection horizon of only two or three seconds. An insufficient time for the car to take the right decision in order to navigate safely. Alternatively, a detailed 3D map allows lengthening the car's vision on several kilometers ahead. It provides an additional context for sensors in real-time, allowing the vehicle to distinguish abnormal situations from normal driving conditions. In addition, autonomous cars will be able to improve their positioning by referring to the information contained in the map regarding what they perceive in real time. This is an additional level of reliability compared to the regular GNSS/IMU positioning.

For self-driving cars, the need for more accurate, complete and clear maps go far beyond basic turn-by-turn directions. Precision is very important; a few centimeters may lead to exhaustion of spirit. Completion should be reflected in the inclusion of these maps to the street signs, lane markings, and traffic signals.

At present, the most successful autonomous driving projects use the prior maps for a precise self-localization in the mapped environment [Vivacqua 2017]. Many companies, such as GoogleX of Alphabet Inc., Tesla, BMW, Uber, Honda, Toyota, Navya, and NuTonomy, use high definition street map for the navigation of their own highly automated vehicles.

Google has come a long way in designing a high-end system that combines a three-dimensional street map with cameras, 3D LiDAR, and artificial intelligence, based on its Google Street View project and on its international coverage. Up until now, the navigation strategy adopted by Google for its autonomous car is to use prior maps. Their fleet has traveled more than 2.4 million kilometers autonomously using this strategy [Van Brummelen 2018].

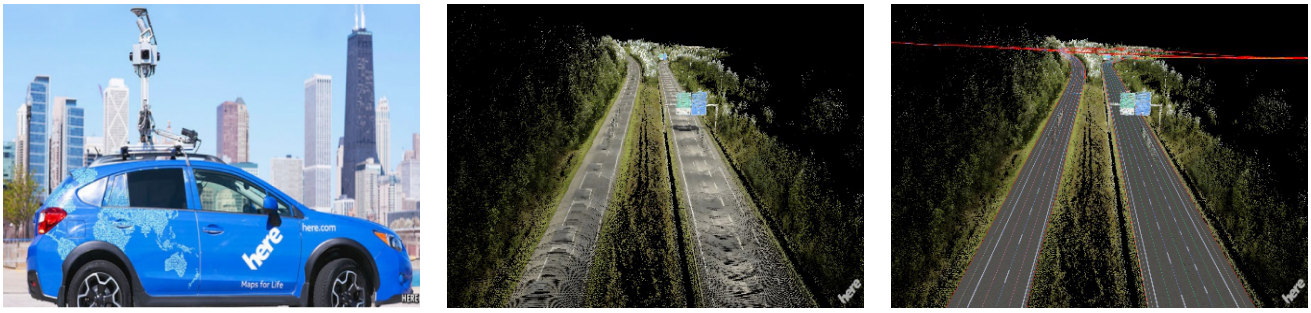
However, Google is not the only company in this field; almost all automakers are working to produce their own prior maps. Tesla collects navigation data through its Model S and Model X owners and creates a high-precision map, instead of sending a dedicated fleet of mapping cars. The data recorded by each camera and GPS of each vehicle are sent over the Internet to a cloud service that collects this information. Then, post-processing treatments are performed to improve this map and adapt it to the autonomous driving purpose.

Ford autonomous vehicle research group has invested in Civil Maps [8], a technology start-up with its own cameras, which works on the generation of HD Maps for autonomous vehicle purpose.

Mercedes in 2014, in the Bertha's Benz Memorial Road commemoration. The 100 km autonomous driving route, between Mannheim and Pforzheim, was carried out using very precise contextual road-maps as well as a geo-referenced landmark database for the vehicle localization [Ziegler 2014].

HERE [9] is a Netherlandic company specialized in mapping and navigation. Originally was the American company Naveteq, which was acquired in 2007 by Nokia. HERE has been bought by a German consortium composed of Daimler, BMW, and Audi. It uses a fusion of LiDAR and GPS data for the construction of highly precise and dynamic maps. These maps consist of tiled layers providing road and lane topology attribution. They capture important details such as the detailed geometry of the road (lane boundaries, slope, curvature, etc.), lane markings and roadside objects such as signposts. The collected point clouds have a relative accuracy of 20 cm over a distance of 100 m (Figure 2.11). Their other advantage lies in their constant updates. HERE affirms that their own worldwide map database receives 2.7 million updates every day [Jomrich 2017].

[8] Source: https://civilmaps.com/
[9] Source: https://www.here.com/en



(a) (b) (c)

Figure 2.11: *HERE HD maps. Left: HERE mobile mapping vehicle. Middle: initially LiDAR map of the Francilienne (N104) between A6 and A10 motorways in the south of Paris. Right: Live Roadmap combining details of the environment with traffic lanes and flow information.*

For the moment, technology is evolving and there is still a lot to do, between the cars of we know and the fully autonomous car, especially in the field of mapping and localization. The proposed maps are not defect-free and errors or imperfections can slip into the data. Many technical challenges remain:

1. Perform accurate 3D maps of the environment is extremely difficult. The enormous volume of data used in these maps is a dilemma. Another challenge is to keep them up to date, so that they provide the latest information to the cars.
2. Each company develops its own map, and considers it like an internal secret. This is a painstaking process and wastes many resources and forces every company to reinvent the wheel. These companies use different standards.
3. The problem of the change of the conduct-law from one country to another obliges the mapmakers to adapt the process of creation of maps to each country.
4. Map should be like a living organism, updating (refreshing) at regularly. Moreover, it should detect changes in real-time and update it (things like accident, path change due to a lane closures, street signs).

2.3.4 Maps for localization purpose

A prior map in autonomous driving can be used for several purposes, including path planning, obstacle avoidance, object detection and tracking, as well as localization. The latter is often the first step executed by the navigation system, because all the other tasks are based on it. This prior map can be of different shapes and of different natures. There are generally six families:

2.3.4.1 Feature maps

These are sparse metric maps, so a set of landmarks (or environmentally distinctive objects) that will allow the robot to localize in this environment. These kinds of maps are characterized by their easier processing and updating. The

landmarks are of different natures:

- These methods exploit laser-plane rangefinders (scanning only in a horizontal level parallel to the ground) in order to generate 2D maps. The landmarks correspond to the intersections of the scanning beams with the objects present in the scene; which are often 2D segments. The vehicle is located only in 2D $(positionX, Y, \Theta)$.
- With regard to vision-based maps, thus building maps from images, the landmarks are mostly 3D points. These points have no semantic interpretation; they simply have photometric properties that make their correspondents projected into images, are key-points (points "easy" to detect and match). Birem [Birem 2015] from Institut Pascal wrote a Ph.D. thesis about localization and loop closure detection for a mobile robot using visual saliency. The main drawback of these approaches is their computational complexity. For that, an implementation of the salient region detector on the reconfigurable platform DreamCam was carried out during this thesis. With regard to vision-based maps, thus building maps from images, the landmarks are mostly 3D points. These points have no semantic interpretation; they simply have photometric properties that make their correspondents projected into images, are key-points (points "easy" to detect and match). Birem [Birem 2015] from Institut Pascal wrote a Ph.D. thesis about localization and loop closure detection for a mobile robot using visual saliency. The main drawback of these approaches is their computational complexity. For that, an implementation of the salient region detector on the reconfigurable platform DreamCam was carried out during this thesis.
- Among the feature maps, we find also, laser reflectance-based maps, which contain landmarks of the reflectance of some key-points or part of interest, such as lane markings, traffic signs, etc.

2.3.4.2 Occupancy maps

Occupancy maps or "Occupancy Grid" [A.C. Schultz 1998, Elfes 1989]. These methods only work in flat terrain, often indoors. The ground plane is discretized in cells with a fixed resolution; Probabilistic Methods [Elfes 1989] (using the Bayes law generally) allow the fusion of observations into this map. Each cell has a probability of being either occupied or free. The localization is achieved by matching the overall map under construction, with the current observations, usually by building a local robot-centered map.

2.3.4.3 3D voxel maps

They have been proposed to also integrate observations acquired by a 3D LiDAR (such as Velodyne, or Hokuyo mounted on pan-tilt unit) or by a Kinect camera. The OctoMap method [Hornung 2013] developed by Freiburg, and available under ROS [10].

[10] **ROS**: **R**obotic **O**peration **S**ystem

Figure 2.12: *OctoMap [Hornung 2013]*

2.3.4.4 Raw data maps

That do not build specific maps, but keep the raw observations, possibly preprocessed and filtered. The map is, therefore, a base of images in vision, or 3D point clouds for the 3D sensor. The localization is obtained by indexation methods in vision [Raoui 2011], or in 3D data by registration methods exploiting generally a variant of the well-known ICP [11] algorithm [Besl 1992]. We will discuss in detail this algorithm in Chapter 4 and Chapter 5.

2.3.4.5 3D building model map

They are generally used for satellite-based localization. These maps contain approximate 3D building models, typically generated from a 2D map and Digital elevation model (DEM). The latter is necessary to determine the height of the buildings. The localization using these maps is obtained by detecting and correcting the wrong GNSS measurements thanks to the knowledge of the building forms. [Obst 2012] generates 3D building models, presented in Figure 2.13, from the 2D OpenStreetMap [12] and the DEM data. These models are used to detect whether buildings hinder the direct visibility of the vehicle GNSS receiver to certain satellites. Pseudo-ranges that are not directly observable will not participate in the position computation. Similar work can found in [Ben-Moshe 2014], and for more detail about these maps, please refer to a Ph.D. thesis of Kurdej [Kurdej 2015].

[11] **ICP**: **I**terative **C**losest **P**oints

[12] Source: https://www.openstreetmap.org

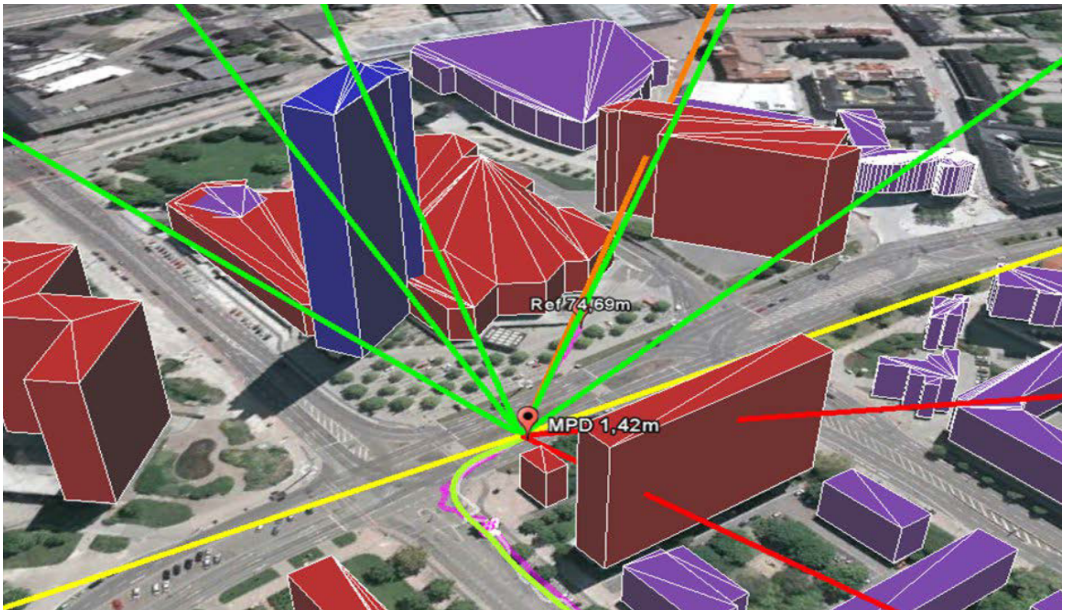


Figure 2.13: *3D building model map used for satellite based localization [Obst 2012]*

2.3.4.6 Semantic maps

These maps represent the last trend of research in the mapping field. The ultimate goal is to enrich the spatial data with other, higher-level concepts. These maps provide an abstraction of space and a means for human-robot communication. The localization is performed using object-based registration methods. We will detail this concept in Chapter 4, dedicated to registration techniques. An up-to-date survey of these approaches can be found in [Kostavelis 2015].

2.3.5 assessment on mapping

Since the origins of autonomous navigation, the mapping techniques have always been a subject of intense research. Many of research works use the prior map as a virtual sensor to overcome the problem of lacking information. In this thesis, we chose to exploit prior maps in order to estimate the vehicle position, because of their advantages that we have previously detailed. The localization is therefore done by matching between a 3D prior map and point clouds received as the vehicle moves.

In this section, we have addressed the mapping problem. After analyzing the 3D data acquisition techniques, we have discussed maps representation means. Thereafter, we have accomplished a state of the art of maps adapted to the localization purpose in autonomous driving.

We have retained to use 3D raw data map representation for our localization approach. In Chapters 6 to 8, we will detail how these maps are elaborated and the way in which localization is done.

2.4 Localization

The localization is a process of determining the pose of the vehicle with respect to the given map or the generated map (estimate its pose relative to its initial position). Accurate localization is essential to the safe and effective functioning of an autonomous vehicle. Several sensors can be used towards this goal. In this thesis, we are interested in LiDAR-based localization techniques. In the following, we drive a dedicated state of the art of this group of methods.

2.4.1 Lidar-based Localization Methods

LiDAR-based localization methods have been the object of intense research in recent years and obvious progress has been achieved. In the following section, we will propose an original classification of localization methods, by grouping them into four categories, depending on the surrounding environment and the robot's knowledge of this environment. The idea behind this classification is to better understand the localization problem as a whole. The state of the art presented later in this section is not exhaustive but lists, in my opinion, the main approaches proposed in the literature of laser-based localization. These methods can be classified into different kinds based on different situations (Figure 2.14).

• Depending on the environment where the robot is located:

Localization methods can be classified according to the following two types:

- Localization of the robot in a static environment that contains only static objects;
- Localization of the robot in a dynamic environment containing static and dynamic objects.

Each of the two types could be further divided into two subgroups:

• According to the whole knowledge of the environment:

- Localization in a completely known environment, in which the robot already knows the model of the environment before it starts to move. This category can also be known as "absolute localization" [Lothe 2010].
- Localization in a partially known or uncertain environment, in which the robot perceives the environment with the help of its sensors to acquire local information from its location. In this case, the robot must determine its position in the environment while mapping it as it moves. This problem is known as simultaneous localization and mapping (SLAM) [Durrant-Whyte 2006, Cadena 2016, Bailey 2006].

2.4.1.1 Localization in a Static and Known Environment

Localization in a static and known environment is the simplest case. The robot knows the entire information of the environment before it starts to move. As the environment configuration will not be changed for a long time,

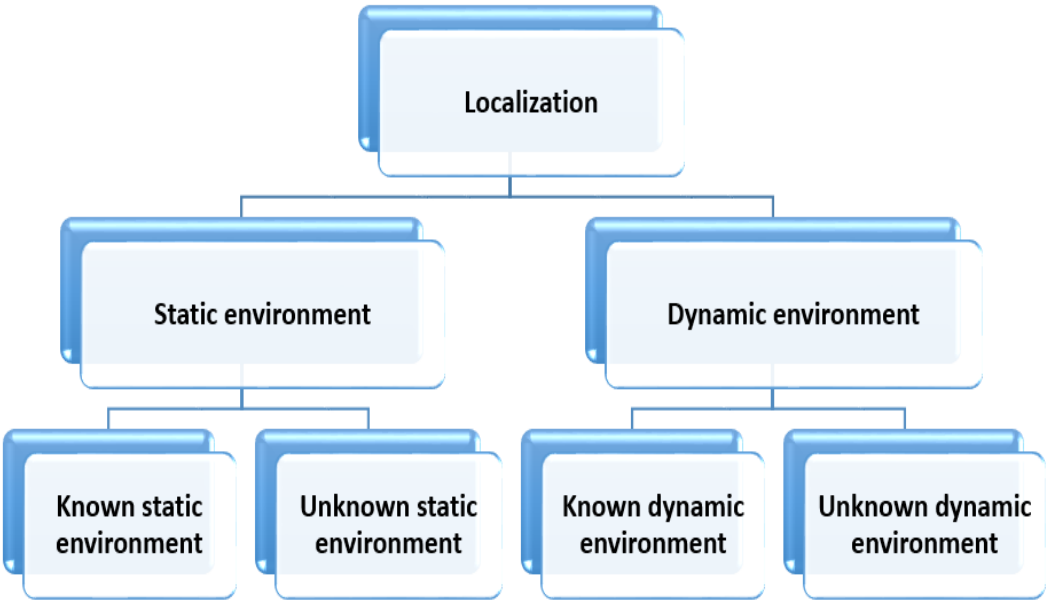


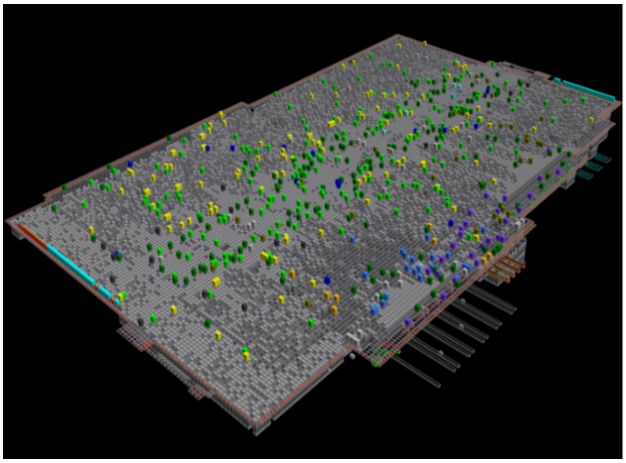
Figure 2.14: *classification of the vehicle localization methods*

the optimal trajectory could be planned offline for once. Then, for every robot's intervention, it executes this trajectory without any constraint. This circumstance is characterized in simple and well-structured environments, such as industrial environments. The British online-grocery company Ocado [13] is one of such environment. The company use air-traffic-control systems and AI [14] technology to coordinate 700 factory robots (Figure 2.15). It uses a unique grid system called "THE HIVE", where the robots assemble customer orders, before taking them to "PICK STATIONS" where human workers put the orders together. Each robot has a central cavity and a set of claws it uses to grab crates and pull them up into its interior. It can then move the crate to a new location or drop it down a vertical chute to a picking station. Each robot can reach the speed of 4 m/s and can pass within 5 mm from each other. They do very basic but efficient tasks that consist of three assignments "lift", "move", and "sort". Ocado has sold its technology to four supermarkets across Europe and Canada.

Regarding autonomous driving technology, driverless cars are already cruising our streets, although, for the moment, they are only prototypes, such as the Google car, Tesla, Uber, and many other companies that all rely on the design of high-tech driverless cars, while forgetting the old non-high-tech cars. Recently, Joukhadar and his colleagues brought this issue forward in their research [Joukhadar 2018] and suggested to upgrading these old cars instead of throwing them away and releasing them. They proposed a mechanical system that can be installed in any type of car and without any modification, which gave the ability to the vehicle to follow a predetermined path and be completely autonomous. This is at the expense of the autonomy of these cars, which operate in dedicated infrastructures that limits their use. In other works, the trajectory of the robot is plotted by a magnetic line as in the case of [Aghaboni 1981] or by an optical line as in the case of [Olivares-Mendez 2011]. Figure 2.16(a) represents the car used for testing above the optical line. The paint on this guide line has been produced with special paints that appear blue when illuminated with ultraviolet light, while remaining colorless under normal light. The circuit

[13] Source: http://www.ocadogroup.com/
[14] **AI**: **A**rtificial **I**ntelligent

(a) The hive-grid-machine



(b) A simulated view of the warehouse. Each dice represents a robot, with colors indicating the state of each robot.

Figure 2.15: *Ocado highly automated e-commerce picking warehouses.*

measuring 190 meters traveled by this car is shown in Figure 2.16(b). The localization in this type of environment is done by very basic systems, such as simple encoders or IMU, which calculate the current position by integrating velocity and acceleration measurements from an initial position. The first drawback of this type of methods is the

(a) The used vehicle

(b) Circuit representation on Google Earth

Figure 2.16: *Visual line guidance system for an urban vehicle [Olivares-Mendez 2011].*

need to install a whole infrastructure. Its second drawback lies in limiting the movement of the robot to specific places. Another disadvantage is illustrated by its inability to handle unexpected situations such as the crossing of its route by an obstacle. Finally, its biggest drawback, in my opinion, lies in the total stoppage of the system if the robot comes out of its previously defined circuit; this scenario requires human intervention to re-establish the robot to its path.

2.4.1.2 Localization in a Static and Unknown Environment

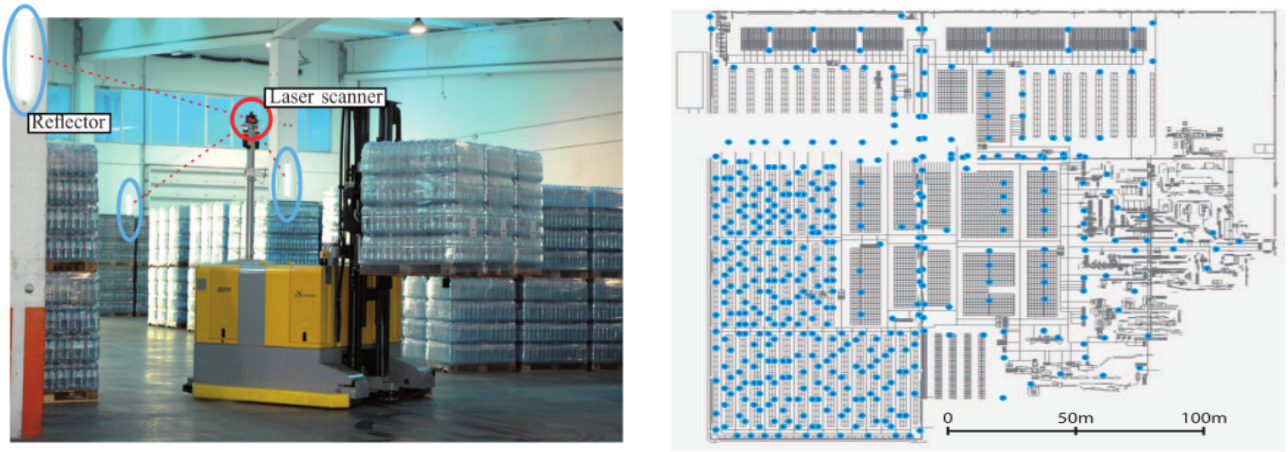
In a static and unknown environment, navigation will be more difficult than in the static case where the environment is known. This difficulty is due to an occasional variance in the environment. The latter is considered static because, at the time of the robot's intervention, it not undergoes any changes. Even if once in a while, it may be required to evolve (of course, outside the intervention phase of the robot). An example of such environments is represented by the case of a warehouse or storage shed. The environment in this case, can be changed every so often due to the movement of goods (moving a pallet between two robot interventions). Previously defined trajectories are no longer feasible in this case, and the crux of the issue is to determine the pose of the vehicle without any prior information on its location. Some approaches that adopt the easier and safer path suggest exploiting the landmarks present in the surrounding environment. While others that prefer the flexibility make use of simultaneous mapping and localization methods.

For the first type of approaches, there are two types of strategies that can be used:

- **The first strategy concerns the use of artificial landmarks:** such as beacons of known geometric shapes (usually cylindrical) previously placed in the sceneI [Loevsky 2010]. This common strategy in industrial environments is used for the localization of AGV [15]. In the work of [Ronzoni 2011] the vehicle equipped with a mono laser scanner as shown in Figure 2.17, measures the distance and angle relative to the various beacons scattered in the environment. Once a set of landmarks are properly detected and associated, the position of the robot will be established. In this instance, the localization is performed in regards to beacons placed at known positions in the environment and not with respect to environment infrastructure that can lead to being modified. The number of beacons used can exceed several hundred, as in this case (450 beacons). One of the drawbacks of this class of methods is that the beacons should be placed in the same height as the laser, as in almost all of these methods, a single-layer laser is used (scanning only in one plane), rendering these systems incapable to track movements other than in the laser plane. In addition, these methods only work for flat terrains, so usually indoors.

- **The second strategy concerns the use of natural landmarks:** many works (see for example [Tardos 2002], [Madhavan 2004], [Núñez 2008], and [Tipaldi 2014]) exploit natural characteristics of the environment such as (corners, walls and different geometric forms) to determine the position of the vehicle. These approaches are very powerful, but they require that a considerable portion of these landmarks remain stable and that each part of the environment is sufficiently distinct [Ronzoni 2011]. A condition that is not always guaranteed. Indeed, in industrial environments, important parts of these benchmarks can be changed. For example, a palette can occlude a landmark and change the geometry of the environment squarely. In addition, many parts of the environment can be very similar (in a symmetric environment like corridors and storage shelves) leading to certain ambiguities in the recognition of a landmark or area [Ronzoni 2011]. For the outdoor applications, landmarks such as tree-trunks or pole-like structures in orchards, rural environments, forest like areas or simply in urban situations without street markings are used to provide information about the vehicle localization. [Shalal 2013] proposes a local-scale orchard mapping based on tree trunk detection (Figure 2.18(a)). The obtained map consists of 2D locations of trees

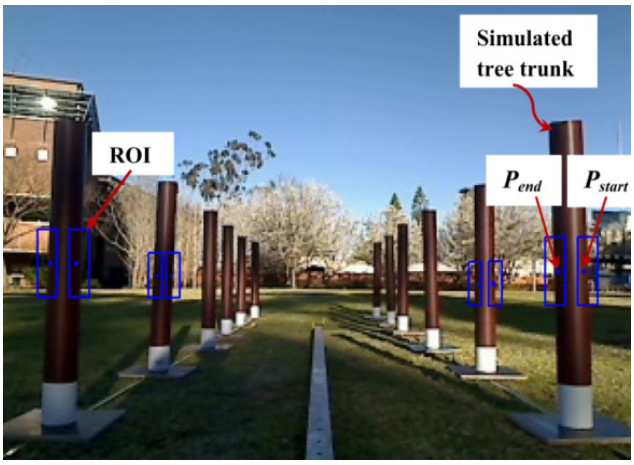
[15] **AGV**: **A**utomatic **G**uided **V**ehicle



(a) (b)

Figure 2.17: *the vehicle and the warehouse used in the work of [Ronzoni 2011].*

will be used as a prior map for localization purpose. [Jagbrant 2013] introduces a single-layer LiDAR-based localization of a mobile robot in an almond orchard. Segmentation of each tree is performed, and a descriptor is calculated on the profile of the tree that will serve as a beacon. [Kampker 2018] presents a landmark-based localization for automated driving (Figure 2.18(b)). In this work, pole-like structures are used as a reliable representation in the last mile and urban scenarios. To compute the pose of the vehicle an Adaptive Monte-Carlo algorithm has been implemented. This environment also has the same drawbacks as that of the indoor one.



(a) Simulated environment from the work of [Shalal 2013]

(b) Pole like structures as valid landmarks for localization [Kampker 2018]

Figure 2.18: *Natural landmarks used in outdoor static and unknown environment.*

To extract different landmarks, algorithms like RANSAC [16], SVM [17], CNN [18], Hough transform are used. RANSAC is often preferred for its efficiency in computing time.

The second type of approaches used in this kind of environment is SLAM methods. SLAM [19] is for simultaneous localization and mapping. The First time this acronym was used is by John J. Leonard and Hugh Durrant-Whyte in [Leonard 1991]. It was SMAL in the beginning but was later changed to SLAM. As its name suggests, the localization takes place simultaneously with the creation of the map. The first study of the SLAM problem has appeared for more than thirty years now. This started with the work of [Chatila 1985, Crowley 1989, Smith 1987]. Since then, wide varieties of different SLAM approaches have been developed. More details on these techniques are revealed in [Durrant-Whyte 2006, Bailey 2006] or more recently in a very good work collecting more over than 300 references in the field [Cadena 2016]. This third category is undoubtedly the category of localization methods where are the most contributions were produced. In these methods and thanks to the addition of new data iteratively, the uncertainty on the map decreases.

According to the recent SLAM survey of Cadena and their colleagues [Cadena 2016], the development of SLAM solutions has gone through three phases:

- **Classical age (from 1986 to 2004):** This period is characterized by the introduction of the main probabilistic formulation for SLAM. [Smith 1986] is the first work in this category. We can find also works like [Thrun 1993] that offer solutions for 2D displacement in static environments such as office buildings. Other off-road works require more degrees of freedom, are proposed by [Nuchter 2004]. This work develops a mobile robot for the exploration of abandoned mines. This phase is outlined by the emergence of filtering approaches rely on Bayes filtering such as Particle Filters [Dellaert 1999, Murphy 1999], or Gaussian filters such as Extended Kalman Filters [Castellanos 1999, Paskin 2003], and information filter [Thrun 2004]. [Durrant-Whyte 2006, Bailey 2006, Thrun 2008] are good surveys for a more comprehensive exploration of the matter.

- **Algorithmic-analysis age (from 2004 to 2015):** The main characteristic of this period is the study of the fundamental proprieties of the SLAM problem (the observability, the convergence, and the coherence). A number of many open-source SLAM libraries are also emerging, such as g2o [Kuemmerle 2011], GTSAM [Dellaert 2012], SLAM ++ [Salas-Moreno 2013]. [Dissanayake 2011] give a good overview of this period.

- **Robust-perception age (from 2015 to now):** This represents the new chapter of SLAM history. Which means a wide set of challenges as well as a broad range of opportunities to develop SLAM research a step forward. This will be centered around four main axes, according to the authors of this review: high-level understanding, robust performance, resource awareness, and task-driven inference. The drawbacks of SLAM-based methods summarized in the fact that they are local methods. They cannot get the global position of the vehicle, but only a relative one to the starting position.

[16] **RANSAC**: **R**andom **S**ample **C**onsensus
[17] **SVM**: **S**upport **V**ector **M**achine
[18] **CNN**: **C**onvolutional **N**eural **N**et
[19] **SLAM**: **S**imultaneous **L**ocalization **A**nd **M**apping

2.4.1.3 Localization in a Dynamic and Known Environment

By this category, we refer to methods that are able to determine the position of the robot in a priori known model, initially provided to the robot. This includes methods matching online data with a priori map. Although the vast majority of real-life environments are dynamic, these environments also contain static objects. The map originally provided to the robot contains representations of these static objects, such as buildings, road topology, sidewalks and so on. In other words, a priori map is a detailed representation of the surrounding static environment. These representations can be improved for the HD models (HD maps), through 3D modeling of buildings with a photorealistic display, and considering altitude to represent valleys, hills, and mountains. As well as with other information such as street signs, lane markings, traffic lights, etc.

To build a priori map, a specific vehicles driven by human are used. These vehicles are equipped with dedicated sensors that collect detailed data, such as precise 3D point clouds, images and GPS information. The fusion of all this data creates detailed maps that will be stored on large databases, so the vehicle can move autonomously on these mapped locations using these maps.

Localization is performed subsequently by detecting the similarities between the map and the current sensor data. While moving objects detection is achieved by observing the discrepancies between the map and the current sensor data. The goal is to track the position of the vehicle on the prior map. Such kind of localization is distinguished by being a global strategy, where the map serves as an additional "virtual" sensor that adds an extra layer of security and understanding.

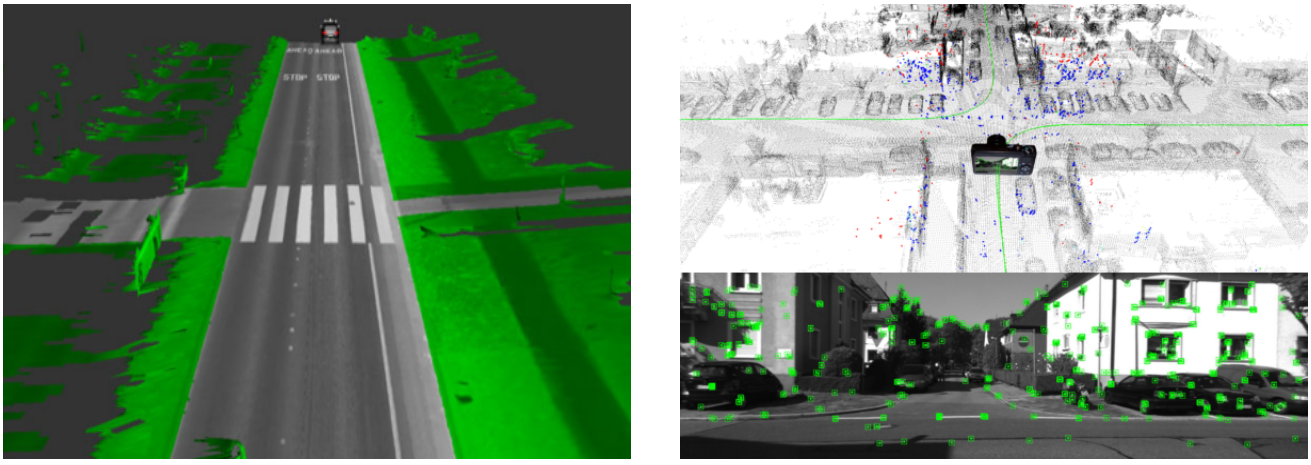
One of the approaches matching the LiDAR data with a previously known map is presented by Hentschel, Wulf and Wagner in [Hentschel 2008], which fuses the GPS pose with laser measurements and match the whole against a 2D reference map containing static characteristics of the environment. This method is dedicated to urban and rural localization. The main advantage of this approach is that when the quality of the GPS signal becomes bad (e.g. close to buildings), the sensors fusion allows the system to maintain a robust and precise localization of the robot.

Another approach introduced by [Levinson 2007] uses a particle filtering method to correlate the laser data acquired online with a pre-prepared 2D map. This map is elaborated using a dedicated car equipped with a GPS receiver, an IMU, an odometer and a laser sensor. They extract the ground points from the laser data and build a map of ground-points intensities. Figure 2.19 shows the process of matching the online data to the 2D map.

Thrun and his team propose in [Levinson 2010a] a robust localization of an urban vehicle using probabilistic maps, where the mean and the variance of the reflectance (the proportion of light reflected by the surface of a material) are stored. These 2D maps are generated from the GPS, IMU and LiDAR data. The impacts of LiDAR rays are projected onto the road contained in the map. Thereafter, the reflectance is used for the correspondence with the measured data. In this work, they used a histogram filter for the corresponding process. This method is the one applied in the Google Car.

Kuemmerle and his colleagues [Kuemmerle 2011] constitute a map of the likelihood field from an aerial photograph. A robot equipped with a single-layer laser navigates in the photographed environment and locates itself using this map.

Localization systems can also be based on occupancy grids. One of these approaches using digital maps and



(a) (b)

Figure 2.19: *(a) Process of correspondence between the online data and the 2D map from the work of [Levinson 2007]. (b) Monocular Camera Localization in 3D LiDAR Maps [Caselitz 2016]. The blue points are the good pairings, whereas the red points have no match. The Green line represents the trajectory of the camera.*

occupancy grids generated from the fusion of multimodal sensors are presented by Konrad and their colleagues in [Konrad 2012]. As a digital map, the authors used a GPS-based roadmap containing waypoints from the road network. The method estimates the width of the road by establishing the correspondence between the digital road map and the occupancy grid generated by the onboard sensors.

More modern efforts include visual localization in 3D LiDAR map. Welcott and his colleagues [Wolcott 2014] propose a method to localize a self-driving vehicle in an urban environment. The localization is achieved by comparing imagery from a monocular camera against a 3D LiDAR map augmented with surface reflectivity. This approach performs matching in 2D space and consequently, it provides only 3-DoF poses. [Caselitz 2016] extends a similar visual localization to estimate the 6-DoF pose. The method localizes and tracks a monocular camera in a 3D LiDAR map. It reconstructs a sparse set of 3D points from its input images via bundle adjustment. Then, the reconstructed points are aligned with the map points by using ORB-SLAM [Mur-Artal 2015]. However, both approaches required expensive image rendering supported by GPU hardware. Moreover, visual-based methods largely depend on environmental conditions and may fail due to poor environmental texture.

For self-driving cars, the most recent successful autonomous driving projects use the prior maps for a precise self-localization in the mapped environment [Vivacqua 2017]. Many companies, such as GoogleX of Alphabet Inc., Tesla, BMW, Uber, Honda, Toyota, Navya, and NuTonomy, use high definition street map for the localization of their own highly automated vehicles. The majority of these companies have their specific maps. Others can simply refer to specialized maps creation companies like HERE [20], TomTom [21] or GoogleX (Google Maps) and use their maps.

In this context, Google's vehicle fleet has traveled more than 2.4 million kilometers autonomously using map-based localization [Van Brummelen 2018]. Stanford has improved their maps with a priori list of traffic light

[20] Here is an American company that was bought by a German consortium consisting of Daimler, BMW, and Audi.
[21] Source: https://www.tomtom.com/fr_fr/

locations so that its vehicle, Junior, can detect traffic lights in different lighting conditions [Levinson 2011].

Mercedes in 2014, in the Bertha's Benz Memorial Road commemoration. The 100 km autonomous driving route, between Mannheim and Pforzheim, was carried out using very precise contextual road-maps as well as a geo-referenced landmark database for the vehicle localization [Ziegler 2014].

Ford autonomous vehicle research group has invested in Civil Maps [22], a technology start-up with its own cameras, which works on the generation of HD Maps for autonomous vehicle purpose.

In [Häne 2017], the authors use a multi-camera system to generate accurate dense maps, and then visually localize the car with respect to those maps. This system has been used successfully on the autonomous cars of the V-Charge project, demonstrating the practical feasibility of this system.

Moreover, the map can be used to anticipate road conditions. This is the case of [Anderson 2018], where the road information is extracted from the map in advance, and then the vehicle which is equipped with an active suspension changes its driving state according to the state of the road.

Naturally, the prior map can only improve the localization process, but it should be as accurate as possible. This requires the use of a dedicated infrastructure. In addition, map-based methods require mapping the entire navigable environment in advance. Nevertheless, its major drawback remains: in the exemption of the embedded sensors compatible with those used in the elaboration of the prior map, these methods are simply no longer applicable.

2.4.1.4 Localization in a Dynamic and Unknown Environment

In reality, the environment in which the vehicle evolves is generally dynamic and unknown. This environment changes unpredictably as in the case of pedestrians or cars [Tipaldi 2012]. In the same vein, as we have just seen in the previous class, we cannot previously make accurate maps of all places navigable by the vehicle, such as villages, rural environments, mountainous areas, etc. In such an eventuality, the vehicle has no knowledge of its environment, which is also continuously changing. This problem is much more difficult than the classes of the previously defined methods. In this localization class, we find the SLAM methods exposed in the second class, with an additional difficulty, which is the presence of dynamic objects.

Here we distinguish between a moving object and a dynamic object. The first one refers to objects that move during the acquisition process. While the latter defines the mobility of the object. That is, a dynamic object is movable even if it does not move during the acquisition, for example, the parked cars. Thus, a dynamic object is always a moving object, because it can move at any time during the acquisition. However, a moving object (moving during acquisition) is not necessarily a dynamic object (that is the case, for example, of a static objects moved by a human, as in the case of warehouse with moving a set of pallets) [Xiao 2015]. The difficulty here lies in the representation of a dynamic object on the map built as the vehicle progresses. How to represent a dynamic obstacle that sometimes present and sometimes not. This case is clearly found in Parking-type environments or in the streets where there are parked cars. [Tipaldi 2012] proposes a method for robot localization in a dynamic environment (parking surrounded by buildings). The approach is able to distinguish objects that exhibit fast dynamic behaviors such as cars and individuals, objects that can be moved and changed configuration (for instance: shelves, doors),

[22] Source: https://civilmaps.com/

and static objects that don't move, like walls. To represent the environment, they use a dynamic 2D occupancy grid, which implements hidden Markov models, in order to represent the occupation and the transition probabilities corresponding to each cell in that grid.

Finally, Note that even if these classes of methods seem to be well-specified, they are not entirely exclusive. Some localization methods can hardly be cataloged in a single class: SLAM-based localization methods can also be used with a priori maps.

2.4.2 Discussion

Many years of LiDAR localization research have demonstrated the maturity of several methods. However, some difficulties remain complex to circumvent [Merriaux 2016]:

- *Perceptual aliasing:*From the point of view of the sensor, several places in the environment seem to be similar. The perception is not rich enough, and the measures seem to be equivalent, which makes it impossible to distinguish precisely the ambiguities. This is much more common in the indoor environment, which often has symmetries and a relative monotony.
- *Dynamicity of the environment:*As we have seen in the previous section, the dynamic nature of the environment poses a real problem. This may lead to the fact that a priori maps are out of date. This can be caused by: the geometry of the environment is slightly changed (door open or closed). The second cause is all what is referred to dynamic obstacles. By definition, they are not present on the map, and will come to disturb the correlation between the map and the online measurement.
- *Accuracy:*

  Perform accurate 3D maps of the environment is extremely difficult. The enormous volume of data used in these maps is a dilemma. Another challenge is to keep them up to date, so that they provide the latest information to the cars.
- *Different maps standard:*

  Each company develops its own map and considers it like an internal secret. This is a painstaking process and wastes many resources and forces every company to reinvent the wheel.
- *Management of vast environments:*

  Without a priori knowledge on the position of the vehicle, localization approaches encounter a real problem, especially, in a large environment. Because, the ability to test many hypotheses to explore a large space remains very costly in computing time.

2.5 Conclusion

In this chapter, we focused on the issue of environment perception. In the first place, we have discussed sensor technologies used in autonomous driving. The methods for map creation and representation are then addressed,

as well as different maps used to enhance LiDAR-based localization. In the final section, we have reviewed localization techniques. This state of the art has shown that there are many and various techniques available. However, the use of prior maps presents a privileged direction. Indeed, this technique has several advantages. It allows to increase the overall localization accuracy and to detect the presence of disturbances. With prior maps, car focus more on moving obstacles, which reduces the amount of software processes and allowing the vehicle to anticipate and avoid tricky situations.

Prior maps are therefore unavoidable tools for the autonomous driving at the moment. The main weakness of these methods is that all navigable roads must be digitized. This is started to be achieved thanks to the involvement of different actors from the mapping world like TomTom and HERE, etc., and other automotive players such as Ford, Tesla, etc., as well as giants of technologies like Google, Apple, Uber, etc.

The next chapter is dedicated to an overview of mathematical theories used for the purpose of autonomous driving localization.

Chapter III

Fundamentals

CHAPTER 3

Fundamentals

Contents

In this chapter, we introduce the basic concepts and the notations necessary to the understanding of this thesis. After presenting basics geometric notions, we will recall the most important optimization techniques used, as well as the theoretical concepts used for the localization of autonomous vehicles. Finally, the clustering technique is addressed

3.1 Introduction

In the previous chapter, we have introduced the different techniques and maps used to localize an autonomous vehicle. As a result of this state of the art, we chose to use the prior maps as they allow to increase the overall localization accuracy and to detect the presence of disturbances. We also concluded to use a laser-based technique to have a better precision in 3D. The localization is therefore performed by scan matching between a 3D prior map and point clouds received as the vehicle moves. In this chapter, we propose to deepen this localization problem, by detailing its fundamental concepts. Concepts such as the representation and the proprieties of the rigid-body motion in 3D space, registration, clustering and optimization techniques will be considered. This is in order to enlighten some notions in first and secondly to make this document as autonomous as possible. The notation followed is the same used in [Ma 2003].

3.2 Notions of geometry

3.2.1 The Special Euclidian Space $SE(3)$

This section discusses the Special Euclidean group $SE(3)$ and its Lie algebra, which form the mathematical basis of rigid body transformations and velocities.

Consider a map f of a rigid-body motion:

$$f : \mathbf{R}^3 \longrightarrow \mathbf{R}^3; p \rightarrow f(p) \tag{3.1}$$

which preserves the distance and the orientation between two points p_1 and p_2

$$\|p_1 - p_2\| = \|f(p_1) - f(p_2)\| \qquad \forall p_1, p_2 \in \mathbf{R}^3 \tag{3.2}$$

$$f(p_1) \times f(p_2) = f(p_1 \times p_2) \qquad \forall p_1, p_2 \in \mathbf{R}^3 \tag{3.3}$$

This kind of map is called a Special Euclidean transformation. The collection of all such transformations in three-dimensional Euclidean space forms the special Euclidean group $SE(3)$.

- the above proprieties can be used to represent the motion of a rigid-body in a compact way,
- the transformation of a point with an attached coordinate frame is sufficient to specify the motion of the entire object.

3.2.2 Rigid-body transformation

The transformation is always with respect to a coordinate frame and can be decomposed into a translational part and a rotational part. The translation moves the object's coordinate frame in space and the rotation changes its

orientation. Thus, a rigid body motion has six degrees of freedom in total, three degrees for translation and three degrees for rotation.

More formally, let there be an orthonormal reference frame belonging to the Euclidean space, named reference frame, and F is an orthonormal frame named current frame. Let the homogeneous matrix $T \in SE(3) \subset R^{4\times4}$, belonging to the Euclidean special group, of dimensions 4×4 such as:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.4}$$

Where $R \in SO(3) \subset R^{3\times3}$ is a rotation matrix, belonging to the orthogonal special group [1] $SO(3)$ and $t \in R^3$ is a 3×1 translation vector.

The T matrix defines the rigid 3D displacement between the two frames F^* and F, or more commonly the pose transformation between the two frames.

Let $P^* = [XYZ]^T \in R^3$, a 3D point of the Euclidian space defines in the F^* coordinate system. The point P^* can be transferred by the rigid transformation T in the frame F by the matrix multiplication:

$$P = T\overline{P^*} = \begin{bmatrix} r_{11} & r_{21} & r_{23} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = RP + t \tag{3.5}$$

where $\overline{P^*} = [XYZ1]^T$ corresponds to the homogeneous coordinates of point P^*

The properties of the special orthogonal group allow defining the following equations:

- The inverse of a rotation matrix:

$$R^T R = 1 \tag{3.6}$$

- The inverse of a homogeneous pose matrix:

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

3.2.3 Velocity transformation

Motion is parametrized in the Lie group $SE(3)$ as a twist, i.e. each transformation matrix in the Lie group $SE(3)$ describing a rigid body motion has a representation in its associated Lie algebra with a 6×1 parameter vector.

Let $x \in R^6$ be a vector representing instantaneous translational velocity $\vartheta = [\vartheta_x \vartheta_y \vartheta_z]^T$ and rotation velocity $\omega = [\omega_x \omega_y \omega_z]^T$. To recover the instantaneous rotation and translation in Cartesian space,x, is integrated over time

[1] Various representations exist for the rotation matrix, the common one is $SO(3)$. The quaternions and the combination of a rotation angle and axis are other frequently used representations.

with an integration period of $\delta t = 1$:

$$x = \int_0^1 (\omega, \vartheta)\, dt \in SE(3) \tag{3.8}$$

The vector x is connected to a pose $T(x) \in SE(3)$ by the exponential matrix application:

$$T(x) = \exp([x]_\wedge) = \sum_{i=0}^{\infty} \frac{1}{i!}([x]_\wedge) \tag{3.9}$$

where the operator $[.]_\wedge$ is defined by:

$$[x]_\wedge = \begin{bmatrix} [\omega]_\times & \vartheta \\ 0 & 0 \end{bmatrix} \tag{3.10}$$

and the operator $[.]_\times \in SE(3)$ defines the skew symmetric matrix of the vector $\omega = [\omega_x \omega_y \omega_z]^T$ such as:

$$[\omega]_\times = \begin{bmatrix} 0 & \omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \vartheta = \begin{bmatrix} \vartheta_x \\ \vartheta_y \\ \vartheta_z \end{bmatrix} \tag{3.11}$$

The exponential matrix $\exp([x]_\wedge)$ has a closed form solution ([Ma 2003]):

$$e^{[x]_\wedge} = \begin{bmatrix} e[\omega]_\times & V_\vartheta \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.12}$$

Where $e^{[x]_\wedge}$ is computed used the Rodrigues' formula.

$$e^{[x]_\wedge} = I + \frac{1 - \sin(\|\omega\|)}{\|\omega\|}[\omega]_\times + + \frac{1 - \cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times^2 \tag{3.13}$$

And V is

$$V = I + \frac{1 - \cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times + + \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3}[\omega]_\times^2 \tag{3.14}$$

3.3 Notions on the localization theory

Let us consider the vehicle's trajectory as a set of poses,$X = \{x_i\}_{i=0}^n$, with typically $x_i \in R^6$. The purpose of any localization method is to determine the position and orientation of the vehicle at each instant t. This localization is achieved through the search for the transformation between a fixed global frame associated to the surrounding environment, and a moving frame associated to the vehicle, as shown in Figure 3.1. This is equivalent to determine the state vector x_i which expresses the three translations and the three rotations according to the three axes of the coordinate system.

$$x_i = [x\, y\, z\, \rho\, \theta\, \Phi]^T \tag{3.15}$$

To solve the localization problem, the vehicle can count on its sensory information, whether proprioceptive

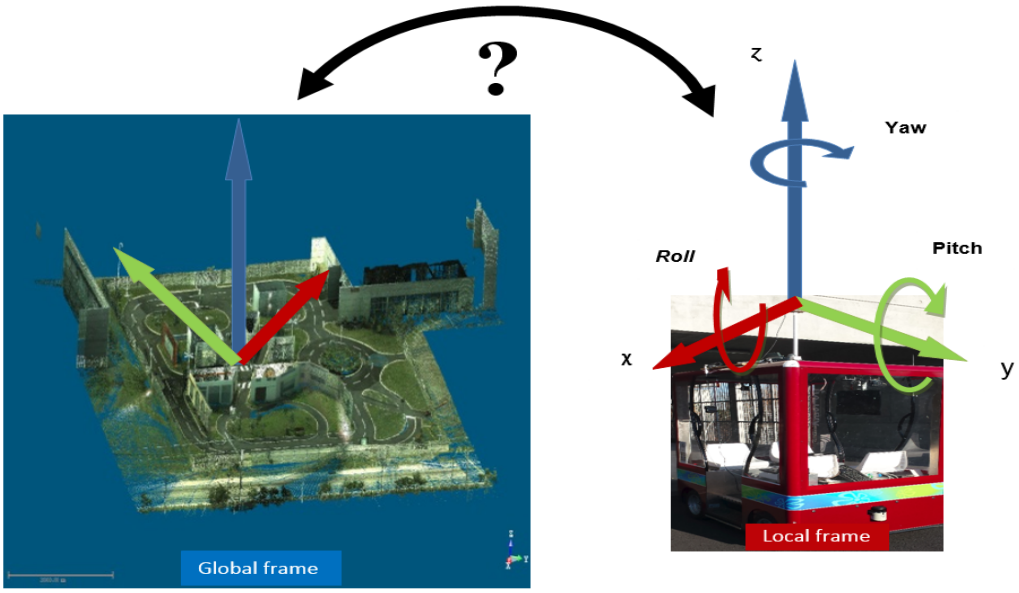


Figure 3.1: *Vehicle localization by calculating the transformation between a local and a global coordinate systems.*

measurements $U = \{u_i\}_{i=0}^{n}$, or exteroceptive measurements $Z = \{z_j\}_{j=0}^{k}$, and in some cases, in prior information that we note it M. At the first instant, we refer to all this information by the known parameter θ.

All localization methods, whatever their modalities, are intended to identify the most probable state vector x_i among several hypotheses. The Likelihood function expresses the probability $P(X|\theta)$ of obtaining a state x_i with a respect to the known parameter θ.

The Likelihood function can be written then:

$$L = \Pi_{i=0}^{k} p(x_i|\theta) \tag{3.16}$$

The product of these probabilities over all x_i is then the Likelihood of obtaining all states given the parameter θ. Now, when we maximize L with respect to θ, we obtain the Maximum Likelihood estimate $\hat{X}$ such as:

$$\hat{X} = \arg\max_{X} \Pi_{i=0}^{k} p(x_i|\theta) \tag{3.17}$$

At this stage and as the vehicle is evolving in a dynamic world that keeps changing, this puts us, according to the classification made in the previous Chapter 2.4, in the second category of localization methods. This category involves two strategies, either using prior maps for localization or localizing without any prior information.

3.3.1 Localization without a prior map

For the vehicle that operate in a priori unknown environment, it should estimates its pose and the map of the surrounding environment at the same time. The most used strategy in this case is the SLAM technique (please refer to 2.4). There are a lot of solutions that attempted to solve the SLAM problem. In this thesis, as we chose to

use raw data maps (2.3), we will consider to solve the SLAM problem in a metric framework. Formally, we search to find the maximum a posteriori estimate [2] of the vehicle's trajectory and map by evaluating:

$$\hat{X}, \hat{M} = \arg \max_{X,M} p(X, M|U, Z) \tag{3.18}$$

As we chose to use only exteroceptive information, which is LiDAR information, the previous equation becomes:

$$\hat{X}, \hat{M} = \arg \max_{X,M} p(X, M|Z) \tag{3.19}$$

Note that the map information is a generic representation that can encompass the different maps that we have seen in the previous chapter (2.3) such as feature map, semantic map, raw data map, etc. Each map has its own estimation requirements within the complete problem.

By considering the sensory measurements as Gaussian random variables,

$$z_j = f_j\left(x_{i_j}, M\right) + w_k, \qquad w_k \sim N\left(0, \sum_j\right) \tag{3.20}$$

Where $f_j(.)$ is a known function called the measurement model or the observation, the joint distribution can now be written as:

$$P\left(X, M, Z\right) \propto p\left(x_0\right) \prod_{j=0}^{k} P\left(z_j|x_{i_j}, M\right) \tag{3.21}$$

$$\propto \prod_{j=0}^{k} e^{-\frac{1}{2}\left\|f_j\left(x_{i_j}, M\right) - z_j\right\|^2_{\sum_j}} \tag{3.22}$$

The maximum a posteriori estimate has now become a product of exponentials. To simplify that, we take the negative log of $p(X, M, Z)$, which reduces the product to a sum of log terms. Therefore, to solve the SLAM problem defined in Eq. (3.19) the maximum a posteriori estimate can be found by minimizing the negative log of the joint probability:

$$\hat{X}, \hat{M} = \arg \max_{X,M} p(X, M|Z) = \arg \max_{X,M} p(X, M, Z) \tag{3.23}$$

$$= \arg \max_{X,M} \left(-\log\, p(X, M, Z)\right) \tag{3.24}$$

$$= \arg \max_{X,M} \left\{ \sum_{j=1}^{k} \left\|f_j\left(x_{i_j}, M\right) - z_j\right\|^2_{\sum_j} \right\} \tag{3.25}$$

[2] Source: https://en.wikipedia.org/wiki/Maximum_a_posteriori_estimation

3.3.2 Localization within a prior map

In the case where the prior map is available and initially provided to the robot, the problem becomes simpler than the full SLAM problem, leading to the following estimation goal:

$$\hat{X} = \arg\max_{X} p(X|M, Z) \tag{3.26}$$

The correspondence between the sensory data (in our case the LiDAR data) and the map is done by scan matching. To this end, several techniques can be found in literature, and regarding us, we focussed on the registration techniques. The purpose of these later is to align a reference point cloud to an input point cloud using an objective function based on a sum of squared differences similarity measure. Next, approaches to obtain an optimal solution for the registration problem are elaborated.

3.4 Registration

Registration algorithms assemble two representations of an environment in a single reference frame. The problem of registration has been dealt with extensively in several studies over the last 25 years. This started with geometric approaches leading to the appearance of the Iterative Closest Point (ICP) algorithm [Besl 1992, Chen 1991]. The strategy of the ICP algorithm consists in taking an optimistic assumption that there is a number of points in common between two point clouds taken from two different points of view, usually called source cloud and target cloud. In this way, the algorithm will have a good initial estimate of the rotation R and the translation t. Applying this assumption, the correspondence of a point will be the closest point to it. In this way, the algorithm will find the closest points of all source points corresponding to the points of the target cloud. Once it has these correspondences, it can improve the estimation of $\mathbf{R}$ and $\mathbf{t}$, by solving this optimization:

$$\mathbf{R}, \mathbf{t} = argmin \sum_{p_i, q_i} \|p_i, q_i\|_2 \tag{3.27}$$

where p_i and q_i denote the pairs of corresponding points in the two clouds and d represents the distance separating the points of each pair.

3.4.1 Cost function formulation

Consider $\overrightarrow{\mathbf{T}(\tilde{x})}$ defines the displacement between the points of the source scan p_i and the points of the target scan q_i, and x a vector belongs to R^6, representing linear velocities $\vartheta = [\vartheta_x \vartheta_y \vartheta_z]$ and angular velocities $\omega = [\omega_x \omega_y \omega_z]$. The convention is to apply $\overrightarrow{\mathbf{T}(\tilde{x})}$ to the source points in order to bring them to the best alignment with the target points.

Suppose now that only an approximation $\hat{\mathbf{T}}$ of $\overrightarrow{\mathbf{T}(\tilde{x})}$ is known. In this case, the registration problem consists in finding the incremental transformation $\mathbf{T}(x)$:

$$\overrightarrow{\mathbf{T}(\tilde{x})} = \hat{\mathbf{T}}\mathbf{T}(x) \tag{3.28}$$

Such that the differences between the positions of the source points registered by the transformation $\hat{\mathbf{T}}\mathbf{T}(x)$ and those of the target cloud, are zero.

$$\mathbf{E}(x) = \sum_{i=1}^{N} \|\overrightarrow{\mathbf{T}(\tilde{x})}p_i - q_i\|^2 = 0 \tag{3.29}$$

$\mathbf{E}(x)$ is the vector of dimensions $(m \times n) \times 1$ containing the errors associated to each point.

Note that the Eq. (3.29) represents the error function for the case of point-to-point metric. There exist also other cost functions, which we will detail in the next Chapter 4.

The linearization of the above cost function leads to a conventional closed-form solution given by a Least Mean Square (LMS) optimization.

3.4.2 Least-square optimization

In the case where the error distribution given by Eq. (3.29) is Gaussian, the least squares estimate corresponds to the estimation of the Maximum Likelihood. In this case, the solution obtained is optimal in the statistical sense. The approach used to solve this problem depends on the linearization of the $\overrightarrow{\mathbf{T}(\tilde{x})}p_i$.

3.4.2.1 Linear methods

When the $\mathbf{E}(x)$ is linear, there exists a matrix $\hat{\mathbf{T}}$ such as for any x,$\hat{\mathbf{T}}(x) = \mathbf{T}(x)$.

For this case and for the "point-to-point" error metric, closed form solution exist in order to estimate the rigid transformation, such as singular values decomposition (SVD) [Arun 1987], the use of units quaternions [Horn 1987], or dual quaternions [Walker 1991]. Eggert et al. [Eggert 1997] evaluated the numerical accuracy and stability of each of the previous techniques, concluding that the differences between theme are minimal. We are going to describe the SVD right after, as it is the commonly technique used in the linear minimization problems.

Singular Value Decomposition

As a consequence of the least-squares solution to Eq. (3.29), the point sets p_i and q_i should have the same centroid. That implies:

$$\bar{p} = \frac{1}{m}\sum p_i; \qquad \bar{q} = \frac{1}{n}\sum q_i; \tag{3.30}$$

where m and n are respectively the number of source points and target points.

The deviations of points from the centroid of each cloud are given by:

$$\acute{p}_i = p_i - \bar{p}; \qquad \acute{q}_i = q_i - \bar{q}; \tag{3.31}$$

The need at this stage is to find the rotation matrix $\mathbf{R}$ and the translation vector $\vec{t}$ in order to minimize the error of the Eq. (3.29). The latter becomes:

$$E = \sum_{i=1}^{N} \|\mathbf{R}p_i + \vec{t} - q_i\|^2 \tag{3.32}$$

Using the above definitions, this can be rewritten:

$$E = \sum_{i=1}^{N} \|\mathbf{R}\left(\acute{p}_i + \bar{p}\right) + \vec{t} - \left(\acute{q}_i + \bar{q}\right)\|^2 = \sum_{i=1}^{N} \|\mathbf{R}\acute{p}_i - \acute{q}_i + \left(\mathbf{R}\bar{p} - \bar{q} + \vec{t}\right)\|^2 \tag{3.33}$$

In order to minimize the error metric, the translation vector $\vec{t}$ must be chosen in such a way that it moves the source centroid to the target centroid.
Which gives:

$$\vec{t} = \bar{q} - \mathbf{R}\bar{p} \tag{3.34}$$

This simplifies the error expression:

$$E = \sum_{i=1}^{N} \|\mathbf{R}\acute{p}_i - \acute{q}_i\|^2 = \mathbf{R}\mathbf{R}^T \sum_{i=1}^{N} \|\acute{p}_i\|^2 - 2tr\left(\mathbf{R}\sum_{i=1}^{N} \acute{p}_i\acute{q}_i^T\right) + \sum_{i=1}^{N} \|\acute{q}_i\|^2 = \sum_{i=1}^{N} \|\acute{p}_i\|^2 - 2tr\left(\mathbf{R}\sum_{i=1}^{N} \acute{p}_i\acute{q}_i^T\right) + \sum_{i=1}^{N} \|\acute{q}_i\|^2 \tag{3.35}$$

Let:

$$N = \sum_{i=1}^{N} \acute{p}_i\acute{q}_i^T \tag{3.36}$$

To minimize the error E at this step, the trace $tr(RN)$ must be maximized. Suppose the rows of $\mathbf{R}$ and the columns of N noted by r_i and c_i respectively. The trace of RN becomes:

$$tr\left(RN\right) = \sum_{i=1}^{3} r_i.c_i \leq \sum_{i=1}^{3} \|r_i\|\|c_i\| \tag{3.37}$$

where the inequality is only a reformulation of the Cauchy-Schwarz's [3] inequality. Since the R-rotation matrix is orthogonal by definition, this implies:

$$tr\left(RN\right) \leq \sum_{i=1}^{3} \sqrt{c_i^T c_i} = tr\left(\sqrt{N^T N}\right) \tag{3.38}$$

Consider the singular value decomposition [4] of N

$$N = U\sum V^T \tag{3.39}$$

By choosing the rotation matrix $\mathbf{R}$ equal to:

$$\mathbf{R} = VU^T \tag{3.40}$$

[3] Source: https://en.wikipedia.org/wiki/Cauchy-Schwarz_inequality
[4] Source: https://en.wikipedia.org/wiki/Singular-value_decomposition

The trace of RN becomes:

$$tr\left(VU^TU\sum V^T\right) = tr\left(V\sum V^{-1}\right) = tr\left(\sqrt{V\sum^T\sum V^{-1}}\right) = tr\left(\sqrt{N^TN}\right) \tag{3.41}$$

which is as large as possible according to (3.38).

3.4.2.2 Non-linear methods

When $\mathbf{E}(x)$ is non-linear, it is possible to solve the Least Square problem of the Eq. (3.29) by using an iterative method. The hypothesis made by this family of methods is that the $\mathbf{E}(x)$ is locally linear. Technically, non-linear optimization methods compute from the set of found pairs a transformation $\mathbf{T}(x)$ and smartly determine how it should be changed to decrease the cost function $\mathbf{E}(x)$. In other words, these techniques consist of calculating the transformation matrix that minimizes the metric error iteratively. At each iteration, the algorithm provides an estimate of this matrix that will be used in the next iteration until the convergence of the algorithm. The latter converges when the error metric is less than a certain threshold.

There are several nonlinear optimization algorithms, such as Gradient Descent, Gauss Newton, Levenberg-Marquardt, and others. These algorithms differ from each other in robustness and speed. In the following, we will detail the Gauss Newton algorithm, as it was the choice of this thesis.

Gauss Newton

In the case of point-to-point metric, the error function to be minimized is given by:

$$E\left(x\right) = \sum_{i=1}^{N} \|\overrightarrow{\mathbf{T}\left(\tilde{x}\right)}P_i - q_i\|^2 \tag{3.42}$$

Since an approximation of the displacement $\overrightarrow{\mathbf{T}\left(\tilde{x}\right)}$ is known, the increment $\mathbf{T}\left(x\right)$ is assumed to be small. In this case,it is possible to linearize the vector $\mathbf{E}(x)$ by performing a developing Taylor series around $x = 0$:

$$e(x) = e(0) + \mathbf{J}(0)x + \frac{1}{2}\mathbf{H}(0,x)x + O(\|x\|^3) \tag{3.43}$$

where $\mathbf{J}$ is the Jacobian matrix of the error vector $\mathbf{E}$, with dimensions $(m \times n) \times 6$ and represents the variation of $e(x)$ as a function of each component of x:

$$\mathbf{J}(x) = \nabla_x e(x) \tag{3.44}$$

and the matrix $\mathbf{H}(x_1, x_2)$ of dimensions $(m \times n) \times 6$, is defined $\forall (x_1, x_2) \in \mathbb{R}^6 \times \mathbb{R}^6$ by:

$$\mathbf{H}(x_1,x_2) = \nabla_{x_1}\left(\mathbf{J}(x_1)x_2\right) = \left[\frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}x_2 \frac{\partial^2 e_2(x_1)}{\partial {x_1}^2}x_2 \dots \frac{\partial^2 e_n(x_1)}{\partial {x_1}^2}x_2\right]^T \tag{3.45}$$

where each Hessian matrix $\frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}x_2$ represents the second derivative of $\mathbf{E}$ with respect to x.

The system of equations (3.43) can be solved with a least-squares method. This is equivalent to minimizing the following cost function:

$$O(x) = \frac{1}{2}\|e(0) + \mathbf{J}(0)x + \frac{1}{2}\mathbf{H}(0,x)x\|^2 \tag{3.46}$$

A necessary condition for the vector x to be a minimum of the cost function is that the derivative of $O(x)$ is zero at the solution, i.e. $x = \tilde{x}$:

$$\nabla_x O(x)|_{x=\tilde{x}} = 0; \tag{3.47}$$

In this case, the derivative of the cost function can be written:

$$\nabla_x O(x) = (\mathbf{J}(0) + \mathbf{H}(0,x))^T \left(e(0) + \mathbf{J}(0)x + O(\|x\|^2)\right) \tag{3.48}$$

The standard method for solving equation (3.47) is Newton's method. It consists of incrementally determining a solution x by:

$$x = -\mathbf{Q}^{-1}\mathbf{J}(0)^T e; \tag{3.49}$$

where the matrix $\mathbf{Q}$ is written:

$$\mathbf{Q} = \mathbf{J}(0)^T\mathbf{J}(0) + \sum_{i=0}^{N} \frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}|_{x=0} e_i \tag{3.50}$$

However, Newton's method requires the calculation of the Hessian matrices, which is expensive in computation time. Nevertheless, it is possible to approximate the matrix $\mathbf{Q}$ with a first order approximation by the Gauss-Newton method:

$$\mathbf{Q} = \mathbf{J}(0)^T\mathbf{J}(0) \tag{3.51}$$

For this genre of non-linear optimization problem, the Gauss-Newton method is preferred, because, on the one hand, it makes it possible to ensure a definite positive matrix $\mathbf{Q}$ and, on the other hand, to avoid the rather expensive calculation of the Hessian matrices.

Under these conditions, at each iteration, a new error $\mathbf{E}$ and a new Jacobian matrix $\mathbf{J}(0)$ are computed in order to obtain the new value of x by:

$$\mathbf{x} = -\left(\mathbf{J}(0)^T\mathbf{J}(0)\right)^{-1}\mathbf{J}(0)^T e(x) \tag{3.52}$$

and to update the rigid transformation by:

$$\hat{\mathbf{T}} \longleftarrow \hat{\mathbf{T}}\mathbf{T}(x) \tag{3.53}$$

In general, the minimization is stopped when the error: $\| e \|^2 < \alpha$ occurs, or when the calculated increment becomes too small: $\|x\|^2 < \varepsilon$, where α and ε are predefined stop criteria.

Other techniques

Several methods exist in literature to approximate matrix $\mathbf{Q}$ of the Eq. (3.50) with a positive definite matrix. These are listed as follows:

- Gradient decent:

$$\mathbf{Q} \cong \alpha I, \qquad \alpha > 0 \tag{3.54}$$

- Levenberg-Marquardt:

$$\mathbf{Q} \cong \mathbf{J}(0)^T \mathbf{J}(0) + \alpha I, \qquad \alpha > 0 \tag{3.55}$$

Generally, the non-linear methods work well for very small inter-scan incremental movements, but because the entire scan information is used, they take advantage of the massive data redundancy to produce a more robust and accurate pose estimation. Their second strength is that their objective functions can be modified in several ways. For example, they can be improved with weighting functions, which results in what is called robust optimization. That is what we are going to discuss now.

3.4.3 Iterative Reweighted Least Square

Since the equation (3.29) represents a least square problem whose contribution of each residue is quadratic, this implies that the more a point is outlier, the more its residue will be, and therefore the more its influence in the cost function will be great. Outliers may occur under different phenomena, for instance, occlusions, non-rigid entities such as vegetation, or simply due to sensor noise. As a result, local minima may appear, leading to the deviation of the optimization techniques from the global minimum. For this reason, it is necessary to consider outliers into the cost function to down-weight the contributions of high errors. This is also called the Robust Optimization.

The cost function is then rewritten:

$$O(x) = \frac{1}{2} \sum_{i=0}^{n} \rho\left(e_i(x)\right) \left(e_i(x)\right)^2 \cong \frac{1}{2} \|e(x)\|_\rho^2 \tag{3.56}$$

where the function $\rho\left(e_i(x)\right)$ is a weighted measure of the error $e_i(x)$. The robust solution of x then becomes:

$$x = -\left(\mathbf{J}^T D \mathbf{J}\right)^{-1} \mathbf{J}^T D e, \tag{3.57}$$

where D is a diagonal matrix of size $mn \times mn$

$$\begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix} \tag{3.58}$$

containing the weights $w_i \in [0, 1]$, which indicate the confidence associated to each point.

Many weighting functions have been proposed in the literature. We will present the two most used in Computer Vision [Huber 2009]:

$*$ **Huber**.

$$\rho_{Huber}(x) = \begin{cases} \frac{x^2}{2} & if|x| \leq c \\ c(|x| - \frac{c}{2}) & else \end{cases} \tag{3.59}$$

∗ **Tukey**

$$\rho_{Tukey}(x) = \begin{cases} \frac{c^2}{2}\left(1 - \left[1 - \left(\frac{x}{c}\right)^2\right]^3\right) & if|x| \leq c \\ \frac{c^2}{6} & else \end{cases} \tag{3.60}$$

The threshold c corresponds to the residual value from which the points are considered to be outliers. In certain state-of-the-art problems, this threshold is set at a precise value. In others, it is automatically estimated from the measured residuals. In particular, the median absolute deviant (MAD) allows the estimation of this threshold in cases where the distribution of the studied residuals can be assimilated to a Gaussian distribution [Lothe 2010].

3.5 Clustering

Clustering can be defined as the task of automatically identifying groups of similar characteristics from a given set of data points [Kaufman 1990]. It is an unsupervised machine learning technique that is used in many fields such as statistical data analysis (pattern recognition, data mining, etc.), data classification and compression, image analysis, etc. The clustering has a long and rich history in various scientific disciplines, including statistics, mathematics, bio-informatics, engineering, and informatics. As a result, many clustering algorithms have been proposed since the beginning of the years 1950. However, the quality of the results they provide always depends on three main factors: The data, the distance function and the nature of the algorithm itself. These algorithms can be categorized into two groups: hierarchical clustering and partitional clustering [Jain 2010]. Hierarchical algorithms recursively either find nested clusters, in a top-down (divisive clustering) or bottom-up (agglomerative clustering) fashion. In contrast, partitional algorithms find all the clusters simultaneously as a partition of the data and do not impose a hierarchical structure [Celebi 2013]. Most hierarchical algorithms are single-link and complete-link. This mean that, once connected, clusters cannot be partitioned again (agglomerative clustering) and the order in which clusters are formed is crucial (depends on the linkage criterion). In addition, hierarchical algorithms are sensitive to outliers that either leads to additional clusters or can cause other clusters to merge. Moreover, they have a quadratic or higher complexity in the number of data points [Celebi 2013] and therefore are not suitable for large data sets, while partitional algorithms often less complex. The most popular and the simplest partitional algorithm is K-means. Although it was first proposed more than 60 years ago, it remains one of the most used algorithms for clustering [Jain 2010]. Next, this algorithm is further elaborated.

3.5.1 k-means

k-means [Kaufman 1990, Arthur 2007] is one of the most popular clustering techniques. The fundamental principle of this method is based on the need to minimize the intra-cluster distance (the distance between points within each partition) and maximize inter-cluster distance (the spacing between the groups). This idea is formulated by discovering of the following parameters:

$$\Omega_k = \{\mu_j\}_{j=1,\ldots,k}$$

and the labels Γ, such that the following function is minimized:

$$L = \sum_{i=1}^{N} \sum_{j=0}^{K} 1\left[\gamma_i = j\right] \|x_i - \mu_j\|_2^2 \tag{3.61}$$

L is the fitness function of the classification method k-means, where is the set of centroid of each class $j = 1 \ldots K$. $1[.]$ is an indicator function of the associated condition and $\|.\|_2$ is the $L2$ norm or the Euclidean distance and N represents the number of points.

In order to cluster with the k-means method, the objective in equation (3.61) is evaluated iteratively until some convergence criteria are satisfied. Each iteration consists of assigning each point to the closest partition:

$$\gamma_i = \arg \min_{j=1,\ldots,K} \|x_i - \mu_j\|_2^2, \qquad i = 1, \ldots, N \tag{3.62}$$

And update the mean (center) of each cluster u_j:

$$\mu_j = \frac{\sum_{i=1}^{N} \mathbf{1}\left[\gamma_i = j\right] \mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{1}\left[\gamma_i = j\right]} \tag{3.63}$$

The k-means method starts by setting initial values of the positions of partitions centers Ω_k. These centers are set randomly. However, random initialization often generates a near-optimal solution because it cannot always guarantee the convergence to the global minimum. The convergence criteria applied in this method consists of setting a maximum number of iterations or when two successive iterations lead to the same partition.

3.5.1.1 Number of clusters estimation

The k-means clustering method is an extremely simple and efficient method. However, it assumes prior knowledge of the data to choose the appropriate number of clusters (K). The correct choice of K is often ambiguous, for that we use the Elbow method [Tibshirani 2001] (Algorithm 2), which consists of calculating L for different values of K.

The idea of the Elbow method for choosing the appropriate clusters number is to compare the L for a number of cluster solutions. The curve between the number of clusters along the abscissa and the cost function L on the ordinate shows that the error decreases as the extent as K becomes larger. Indeed, when the number of groups increases, the sizes of these groups become smaller, which also leads to distortion also be smaller. The appropriate cluster solution could be defined as the solution at which L decreases sharply. This produces an "elbow effect" in the graph, as shown in the following picture:

For the best of our knowledge, this technique works well in most cases, without being very precise. This

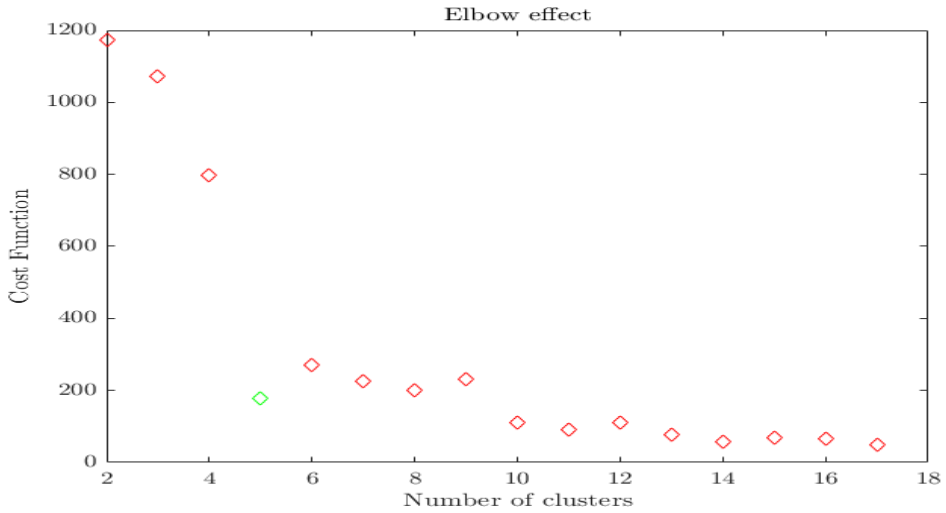


Figure 3.2: *Elbow method: in this graph example, the elbow point is marked in green.*

method still a heuristic that may not operate in some cases.

Algorithm 1: Elbow method

```
Cloud (X):
Let N = number of points in X
Init a list D, of size N/2
Let D[0] = 0
for k = 1 ... N/2 do
    cluster X with k clusters (e.g. with k-means)
    D[k] = L(k)
end
Define J of size (N/2) − 1, with J(0) = 0
for i = 1 ... N/2 − 2 do
    J(i) = D[i] − D[i + 1]
end
Return the k between 1 and N/2 that maximizes J
```

3.6 Conclusion

The localization is one of the fundamental functions of the navigating of any mobile platform. Knowing its position is a better way to understand the surrounding environment, but also to anticipate and plan the trajectory to follow to reach the desired destination. In this chapter, we have seen fundamental notions and concepts to elaborate this localization.

In the remainder of this manuscript, the different concepts defined in this chapter, such as the registration, the robust minimization and the clustering will be frequently used.

Part II

POINT CLOUD REGISTRATION

Through this part, we study the registration technique and more specifically the dense-to-dense registration methods based on the ICP technique. Then we propose a novel approach for the sparse-to-dense alignment.

CHAPTER 4 *addresses the registration issue by implementing and testing a large number of ICP variants. This chapter aims to understand this method that is considered to be one of the key components of mapping and robot localization*

CHAPTER 5 *describes the author's contribution in the domain sparse-to-dense registration. This study is conducted since the final localization strategy is performed using the map-matching between a sparse data and a dense map.*

Chapter IV

Registration

CHAPTER 4

Registration

Contents

In this chapter, a bibliographical and experimental study of the registration technique is presented. The goal is to understand this technique that represents one of the fundamental building block of mobile robotics. It forms an integral part of the processes of mapping, localization, object detection and recognition, loop closure and many other applications. Our primary concern consists of the adaptation of this method to our thesis work, based on 3D laser localization. As the registration is more like a concept than a single algorithm, it involves many steps and

each step can be implemented using different strategies, which gave rise to the appearance of several techniques. In this chapter, we chose to be more practical by implementing several techniques in each step, which resulted in 200 variants. The approach we adopt is to implement several strategies at each ICP stage. The variants will be formed from different combinations of each strategy implemented in each step of the algorithm. Three metrics are used to compare these variants: the number of iterations required for the convergence of the algorithms, the computational performance and the accuracy. We are focused here on only the registration of a single pair of scans, and leave the complete reconstruction (SLAM) to be dealt with in another chapter.

4.1 Introduction

Registration is the technique that allows two representations of an environment to be assembled. In the roboticist's jargon, this refers to the alignment of two images or two point clouds in a single reference frame. A technique that finds application in many robotic systems. A well-known application is that of self-driving cars that will soon share our roads with other cars, automated or not. For example, the project of "Google Self-Driving Car" [1] uses 3D LiDARs (Velodyne) for both navigating and building maps that are used for navigation. If Google has been choosing to use a Velodyne for its own car, this is because it represents one of the most accurate sensors for real-time maps building. It permits to monitor the continuous changes that are made as the environment progresses, which require the alignment of the online-generated point-cloud sequences in a common frame. The problem of registration has been dealt with extensively in several studies over the last 25 years. This started with geometric approaches leading to the appearance of the Iterative Closest Point (ICP) algorithm [Chen 1991, Besl 1992]. ICP is used to calculate the optimal transformation fitting two point clouds by a two-step process: matching of points and minimizing a metric describing the misalignment [Marani 2016]. These two-steps iterate to minimize the matching error and thus improve alignment. As Pomerleau [Pomerleau 2013a] points out, its easy implementation and simplicity that always attract researchers, are both strengths and weaknesses for this algorithm. This has led to the appearance of many variants of the original solution, adapted in many ways every year. Particularly in robotics, the number of possible adaptations becomes so important that it is difficult to make a decision on the choice of the appropriate algorithm for a given scenario. Therefore, we will wonder in the following on how to make such a choice. We first discuss this problem by studying a large number of variants published in literature. Then we will present the implementation of some of these variants by taking the care to test them on our own data. Then we propose a comparison between the implemented variants to achieve to the most suitable variant for our case.

4.2 Related Work

Registration is a crucial step in several applications, ranging from inspection in the medical domain [Markelj 2012], passing through the detection of objects in computer vision [Salvi 2007], to mapping and localization in mobile robotics [Pomerleau 2015], which is our main research interest. Registration is located in the front-end of the mapping pipeline [Pomerleau 2013a]. In recent years, the interest and demand for 3D mapping has been greatly increased. This is mainly due to the improvement of acquisition systems on the one hand and the growth of the range of potential applications on the other. Currently, 3D data can be obtained using two technologies: photogrammetry and laser scanning [Fabio 2003]. The laser technology provides direct 3D data, while photogrammetry reconstructs 3D information by techniques such as triangulation from several images of the area under exploration. The advantage of direct 3D data acquisition makes the laser scanner popular for mapping the environment either indoors or outdoors [Caselitz 2016]. Moreover, localization can be done at a different timescale compared to the mapping, which requires that the process of localization should be robust to the environment change (such as the lighting change) [Caselitz 2016]. In this study, we only focus on laser technology.

Registration algorithms assemble two representations of an environment in a single reference frame. The

[1] Source: https://waymo.com/

problem of registration has been dealt with extensively in several studies over the last 25 years. This started with geometric approaches leading to the appearance of the Iterative Closest Point (ICP) algorithm [Besl 1992, Chen 1991]. ICP is used to calculate the optimal transformation fitting two point clouds by a two-step process: matching of points and minimizing a metric describing the misalignment [Marani 2016]. These two-steps iterate to minimize the matching error and thus improve alignment. In the literature, three groups of registration methods are identified:

- sparse methods (approaches based on features extraction);
- dense methods (approaches exploit all the points in the cloud);
- approaches based on objects.

4.2.1 Sparse Approaches

Sparse methods are generally used in outdoor environments [Maddern 2016]. They are based on the use of features, which may be points that are easily identified by their apparent character (position, local information contents, mathematical definition, etc.) with respect to the other points. A good feature requires stability and distinctiveness [Serafin 2016]. In other words, detected features should be consistent in all the frames. They should be robust to noise and invariant to rotation, perspective distortion and changes of scale [Serafin 2016, Costa 2016, Feng 2016]. There are numerous sparse methods in the literature, each one is adapted to specific needs, but all of them share the same workflow. They begin by the identification of the feature estimation model, then the extraction of the set of relevant points (keypoints [Filipe 2014]) corresponding to the feature model. Afterwards, for every point, a local descriptor is computed collecting the shape and appearance of the neighborhood around each point. Finally, keypoints found in different frames are used to determine correspondences and align the different point clouds.

Among the well-known algorithms is the 3D Scale Invariant Feature Transform (3DSIFT), which is an extension of the 2D version proposed by Lowe in 1999 [Lowe 1999]. The 3D version was adapted by the PCL [Rusu 2011] community using the curvature of points instead of the intensity of pixels [Hänsch 2014]. The method uses a pyramidal approach to reach the scale invariance characteristic of features. To achieve invariance against rotation, it assigns orientations to keypoints.

A multitude of methods exploiting 2D information obtained from 3D points have emerged since. Ranging from FAST (Features from Accelerated Segment Test) [Rosten 2006], and going through SURF (Speeded Up Robust Features) [Bay 2008], until ORB (Oriented FAST and Rotated BRIEF) [Rublee 2011]. These methods, unfortunately, are less robust [Feng 2016] and are affected by parasitic phenomena such as illumination and weather conditions [Serafin 2016].

Normal Aligned Radial Feature (NARF) [Steder 2011] is a rotation invariant 3D feature that also operates in range image and has two goals: extract points from stable local surfaces that are near significant changes and from borders. The authors argue that working on range image makes borders explicitly identified by transitions from foreground to background. Indeed, borders usually appear as non-continuous traversals from foreground to

background. Still according to the authors, points from stable surface that represent a significant change in a local neighborhood represent robust points that can be detected and observed from different perspectives.

However, all those methods cited above, operate on 2D representation of 3D point cloud. Recently, a method developed by [Feng 2016] highlights the use of these two strategies together (2D representation and 3D information). It uses a 2D range image, as well as information calculated from 3D points such as normals and curvatures to extract 3D feature point from LiDAR data.

The last category of sparse methods exploits the 3D points directly. Most well-known approaches include the Point Feature Histograms (PFH) which was used in [Rusu 2008] to describe the local geometry around each point, in order to classify them, by means of a multi-dimensional histogram, according to its local nature (flat surface, corner, edge). PFH is a global feature descriptor as it computes a single descriptor for the entire cloud. Fast Point Feature Histograms (FPFH) [Rusu 2009b] modifies the mathematical model of FPH in order to reduce its computational complexity. Similar approaches are formed elsewhere, such as VFH (Viewpoint Feature Histogram) [Rusu 2010], CVFH (Clustered Viewpoint Feature Histogram) [Aldoma 2011], where features are determined based on the geometric information of 3D points. In the same vein, PCA (Principal Component Analysis) [Jolliffe 1986] are used in [Zhong 2009] and [Mian 2010] to establish 3D features for use in recognition and pose estimation.

In any case, sparse methods exploit information about some key points of the scene [Serafin 2016]. They are based on local characteristics of these points, often only geometric characteristics are taken into account [Weber 2015] although there are other descriptors such as color, intensity, etc. Methods which fall into this category do not require any prior knowledge [Serafin 2015]. Despite this advantage of sparse methods, many of them are not completely adapted to real-time applications [Costa 2016]. Indeed, features are generally cumbersome to determine, and it is unwise to compute them at each point [Costa 2016]. Some methods identify a few numbers of locations where their computing may be more efficient [Yang 2013], but the way these points are determined is time-consuming and hence are often not suitable for the applications that require efficiency. When using sparse methods, another problem occurs which is the necessity of very dense clouds in order to obtain good features, which compromises the use of sparse clouds [Agamennoni 2016, Serafin 2016, Yang 2016, Velas 2016]. More importantly, these methods are environment specific [Cadena 2016], which may result in the rejection of good data [Nieto 2006].

4.2.2 Dense Approaches

Dense approaches make use of all the points from both clouds, and require an initial guess (transformation) between the two clouds, which makes them sensitive to wrong initialization [Yang 2013, Serafin 2015, Costa 2016]. Despite the use of all the points, these methods are generally faster than sparse approaches [Serafin 2015].

ICP algorithm belongs to this class of methods. Its strategy consists of supposing an optimistic assumption that there are a number of points in common between the two clouds taken from two different viewpoints. In this way, the algorithm will have an adequate initial estimate of the translation and the rotation, which moves the points of the source cloud to correspond with the points of the target cloud. Applying this assumption, the correspondence of a point will be the closest point to it. In this way, the algorithm will find the closest points of the source cloud in the target cloud. After each iteration, better matches are found, which gradually produce better

registration. This is repeated until the convergence of the algorithm is reached. At this stage, the final translation and rotation between the two sets of points are obtained. As pointed out by Pomerleau [Pomerleau 2015], its easy implementation and simplicity, are both its strength and its weakness. This has led to the emergence of many variants of the original solution, adapted in many ways, throughout the years. Most well-known examples, Chen et al. [Chen 1991] improved the standard ICP by using point-to-plane metric instead of the Euclidean distance error. This approach takes advantage of surface normal information to reject wrong pairing. However, this approach fails when dealing with clouds of different densities, since normals computation are affected by the change in resolution, presence of noise and distortion [Das 2014, Holz 2015].

The Normal Distributions Transform (3D-NDT) [Magnusson 2007] discretizes the environment in cells, where each one is modelled by matrix representing the probability of occupation of its points (linear, planar and spherical). Then, a non-linear optimization is performed to calculate the transformation between the two clouds. Nonetheless, according to [Das 2014], the NDT is not suitable for systems with low computing power capability.

An efficient approach for dense 3D data registration was presented in [Segal 2009]. This probabilistic version of ICP called Generalized ICP (GICP) is based on a Maximum Likelihood Estimation (MLE) probabilistic model. It exploits local planar patches in both point clouds which leads to plane-to-plane concept. The authors in that paper show that this algorithm is a generalization of point-to-point and point-to-plane metrics, and the only difference lies in their choice of covariance matrices. Since this algorithm is point-to-plane variant of ICP, it has similar drawbacks, especially those related to normals computation. For instance in [Holz 2015], it is shown that the non-uniform point densities cause inaccurate estimates, which degrade the performance of the algorithm. Moreover, in [Pomerleau 2013b, Agamennoni 2016] the authors affirm that the GICP does not work well in outdoor and unstructured environment.

Serafin et al. [Serafin 2015] extended the GICP algorithm by using the normals in the error function and in the selection of correspondences, which according to the authors, increases the robustness of the registration. NICP [Serafin 2015] works on the projection of the two clouds on range images. For the reference cloud, this range image is recomputed at each iteration, which consumes time. These range images serve primarily for the selection of matched points. The Matched points are selected from the range image, so that they are points that share the same pixel and have compatible normals and curvatures.

4.2.3 Approaches based on Objects

Object-based methods have chosen to take advantage of higher-level representations, including 3D objects (solid shapes), 2D forms (plans), or 1D (segments). This concept allows them to benefit from a massive compression of information [Dubé 2016].

In [Salas-Moreno 2013], the authors propose a SLAM algorithm that combines recognition and 3D reconstruction of maps at the level of the object. During the navigation process, the algorithm uses prior knowledge of specific objects that are supposed to be in the environment, to perform a recognition task. These objects are used as top-level features to optimize the ICP-based pose refinement. However, this work is limited to the indoor environment and the specific known objects.

Fernandez-Moral et al. [Fernández-Moral 2016] propose a registration method based on planar surface. This

paper represents an extension of the work published in [Fernández-Moral 2013] which deals with the recognition places in indoor environments by extraction of planes. The extension is mainly focused on adding a probabilistic framework to account for the uncertainty model of the sensor. Whereas for [Dubé 2016], this approach is applicable only to small and indoor environments. The method uses the region growing technique [Georgiev 2011] to obtain the planar patches from the scene and represents them using a graph. Other techniques may also be used such as RANSAC [Fischler 1981] and Hough transform [Forsberg 1995] as in [Rusu 2011] and [Grant 2013]. However, for our proposal (dense-sparse registration), sparsity poses a real problem to get accurate segmentation [Grant 2013].

The segments are also used in the process of matching. In [Velas 2016], the authors introduce a Velodyne point cloud registration method based on line clouds. The algorithm starts by sampling the two clouds into sets of random segments, then the correspondence is made by a strategy similar to the ICP between the two sets of lines. Dubé et al. [Dubé 2016] use a segment-based method for a loop-closure purpose. This method has the advantage of compressing the point cloud into a group of distinct elements, which reduces false matches and optimizes the time required for correspondence.

Object based methods suffer from imperfect segmentation [Dubé 2016]. Their matching tends to reject a lot of potentially useful data, since they exploit information belonging to some simplistic geometric models [Nieto 2006].

4.3 Iterative closest point: The algorithm

The ICP algorithm is an iterative registration method, which consists in putting the points of the source cloud into the frame of the target cloud in order to generate a unique and consistent point cloud. To do this, a translation and a rotation, which make the points of the source cloud move to correspond with the points of the target cloud must be found by the algorithm. Moreover, this algorithm must identify the points of the two clouds that correspond to each other. The strategy of the ICP algorithm consists in taking an optimistic assumption that there are a number of points in common between the two clouds taken from two different points of view. In this way, the algorithm will have a good initial estimate of the rotation R and the translation t. Applying this assumption, the correspondence of a point will be the closest point to it. In this way, the algorithm will find the closest points of all source points corresponding to the points of the target cloud. Once it has these correspondences, it can improve the estimate of R and t, by solving this optimization:

$$R,t = argmin_{R_i,t_i} \sum_{p_i,q_i} \|p_i - q_i\|_2 \tag{4.1}$$

where p_i, q_i denote the pairs of corresponding points in the two clouds and $\|.\|_2$ represents the $L2$ norm (Euclidean distance separating the points of each pair).

After each iteration, better matches are found producing progressively better registration. This is repeated until the algorithm converges. It converges when the distance is less than a certain threshold. Once this convergence is reached, the final rotation and the translation between the two sets of points are obtained. Rusinkiewicz [Rusinkiewicz 2001] identify six distinct stages in this algorithm:

1. **Selection:** selecting a set of points from one or both clouds of the input point (source and target).

2. **Matching:** it is the pairing of the source and target points.
3. **Weighting:** assignment of weights to matched pairs of points.
4. **Rejection:** reject the pairs of points that do not contribute positively to the convergence of the algorithm.
5. **Error metrics:** it defines the objective function, which is minimized at each iteration of the algorithm.
6. **Minimization:** minimize the error metric to bring the points of the source cloud and align them with the points of the target cloud.

The algorithm terminates when a maximum number of iterations is reached or a variation relative to the error metric is reached. In many cases, the algorithm converges quickly but not necessarily towards the optimal solution. Several problems may arise, namely:

- noises and outliers that can cause biased results,
- partial overlap.

4.4 The ICP variants

There are many variants of this algorithm according to different optimization formulas, different ways of choosing pairing points, and different ways of rejecting bad points. Given the amount of work on this issue, we are not aiming to be exhaustive, but we will simply cite a few main approaches in order to get an overview on the subject. We resume the steps identified by Rusinkiewicz [Rusinkiewicz 2001], presented above, to give a brief overview of different works proposed at each stage.

4.4.1 Selection

This first stage seeks to reduce the number of points of the input clouds by applying one or more filters [Gressin 2012]. The way in which these points are selected has a direct impact on the convergence of the algorithm, and especially on the computation time necessary for convergence, in particular, when handling very dense datasets. There are several selection approaches:

- **Use all available points:** in the work of [Besl 1992], it was considered that the points selection is not necessary when there is a considerable overlap between the two clouds and the number of outliers is not significant.
- **Uniform sampling:** for dense-point clouds, sampling is required to make the computing time acceptable. Among the works that used a uniform sampling, we find [Chen 1991] and [Zhang 1994].
- **Random sampling:** for this strategy, it is to sample differently at each iteration of the algorithm in order to avoid any bias towards outliers [Masuda 1996].

- **Sampling according to the orientation of normals:** his method uses the normals in order to better preserve the geometrical characteristics of the points. The authors [Rusinkiewicz 2001] show, through several studied approaches, that it is better to group the points according to the orientation of their normals then perform a uniform sampling on each group instead of random sampling.
- **Sampling based on point density:** this approach presented by [Lalonde 2006] consists of removing the points whose density is below than a given threshold. This allows discarding outlier points in low-density areas.
- **Sampling based on the eigenvalues combination:** in [Gressin 2012], the authors propose two methods for selecting different points, based on the combination of eigenvalues of the covariance matrix. The first method is based on "dimensionality", which aims to select points with linear behaviors. These points correspond to the boundary between the surfaces and the thin objects. It is best to remove them from the next steps of the algorithm. The second method is to select points with higher entropy values.
- **Outliers filtering:** this method [Costa 2016], based on the suppression of outliers exceeding a distance threshold, is proving to be one of the most widely used and effective approaches. The purpose of the method is to remove points that do not have a nearby neighbor within a specified distance threshold.

Other point selection methods may be exist such as the statistical sampling [Rusu 2009b].

4.4.2 Matching

This step represents the key operation in the ICP algorithm. It consists in coupling corresponding points from both clouds. These correspondences are obtained by seeking, for each point of the source cloud, the nearest point in the target cloud. The definition of the "nearest point" determines the matching technique used [A. Donoso 2016]. Some authors [Besl 1992] use a strategy based on the Euclidean distance. The latter remains the main method used in the ICP variants.

Finding the nearest points is usually the most greedy step in terms of computational time [Holz 2015]. Several techniques of nearest-neighbor-search (NNS) are used to optimize the time of this step, as it used to be the most demanding step in terms of computation time [A. Donoso 2016]. The authors of [Elseberg 2012] and [A. Donoso 2016] assert that "k-d trees" is the best technique to find the nearest neighbor. For this reason, we use it in our implementation.

Further enhancement of the closest point search is made possible by using the normal vectors [Jacques Feldmar, Nicholas Ayache 1994]. As part of this approach, the search space for pairing points goes from three to six dimensions. Other improvements include the contributions of [Schutz 1998, Sharp 2002] and [Akca 2005] which cover the application of curvature, invariant moments, color and intensity that are taken as descriptors applied to Matching process.

Alternatively, the pairing of points can be done by finding the inter-section line-Surface [Chen 1991]. This technique is commonly referred to as "normal shooting" [Rusinkiewicz 2001]. Other pairing techniques using heuristics are sometimes used. Pulli [Pulli 1999] proposes a constraint based on the difference between the angles

of the normal points, allowing only the pairing if the difference of the angles of the normals of the two points is less than 45 degrees. A similar approach is used by [Godin 1994] to match points, if and only if, their intensity is greater than a given threshold.

4.4.3 Weighting

It is the assignment of weights to matched pairs of points. It aims to strengthen the contribution of correspondences believed to be correct and mitigate the effect of false matches [Holz 2015]. The pair of matched points can be weighted differently depending on its compatibility in a certain sense. Concretely, this means multiplying each term in the error metric by a specific factor. The latter could be based on distance, color, curvature or normal directions. To weight the pair of points according to their distance, Godin [Godin 1994] suggested:

$$\rho = 1 - \frac{dist(p,q)}{dist_{\max}} \tag{4.2}$$

Here $dist_{\max}$ is the maximum distance between the points of all pairs.

Luck et al. [Luck 2000] use a version of the ICP algorithm with a weighting of paired points based on the median squares of the distances between these points. The weighting of the pairing points according to their normals n_p and n_q can be performed as in [Rusinkiewicz 2001] using the weight:

$$\rho = \vec{n_p}.\vec{n_q} \tag{4.3}$$

Similarly, for points with curvature values c_p and $c_q$in the range $[0,1]$, where we find a Gaussian weight:

$$\rho = e^{-(c_p - c_q)^2} \tag{4.4}$$

Khoshelhama et al. [Khoshelham 2013] propose a weighting method based on the variance in the depth axis of an image given by a Kinect sensor.

There are also other more advanced weighting strategies such as Huber weighting [Huber 2009] and Tukey weighting [Huber 1981]. We have given the mathematical details of these two methods in the fundamental Chapter 3.

4.4.4 Rejection

This step that is fairly related to the previous is intended to reject the supposed false matches, such as outliers, occluded points (points that are not visible in one of the acquisitions) or unpaired points (points of one cloud that do not find correspondents in the second cloud). Generally, reject the pairs of points that do not contribute positively to the convergence of the algorithm. In the following, some rejection approaches that have been proposed in the literature:

- **Distance-based rejection:** this is the most basic way to eliminate wrong-pairings. This simple and powerful strategy consists of eliminating the correspondences which have distances greater than a given

threshold [Zhang 1994, Pulli 1999].

- **Statistical rejection:** this can be done on a statistical evaluation of the distances of the nearest neighbors and thus reject $10\,\%$ of the bad matches, as was suggested by [Pulli 1999].
- **Rejection based on the normals orientation:** comparing the orientations of the point normals provides a rejection method similar to that based on distance. The difference in the orientations of the normals must not exceed a certain threshold. [Pulli 1999] proposes $45°$ as a threshold. A big difference indicates that the points in question do not share the same local geometry.
- **Edge points Rejection [Rusinkiewicz 2001]:** for point clouds that have partial overlap, points that have edge matches may be rejected. This requires that at least one of the point clouds should be triangulated, which is costly in terms of computational time.

4.4.5 Error metrics

The error metric defines the objective function that is minimized at each iteration of the algorithm. Three metrics are commonly used:

- **point-to-point:** This criterion is the sum of quadratic distance errors between paired points [Besl 1992]. This can be represented as follows:

$$E = \sum_{i=1}^{N} \|\mathbf{R}p_i + \vec{T} - q_i\|^2 \tag{4.5}$$

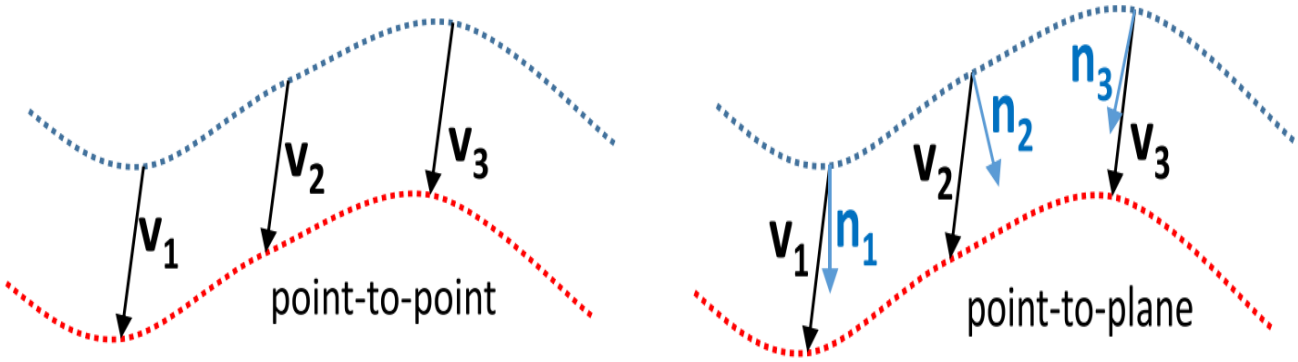
  For an error metric of this form, there are closed-from solutions [Rusinkiewicz 2001]. We have outlined an example of these solutions based on the SVD decomposition in the previous chapter.

- **point-to-plane:** This criterion is based on the sum of the squares of the distances between the points of a cloud and the planes containing the points of the second cloud [Chen 1991]. Its mathematical formulation is:

$$E = \sum_{i=1}^{N} \left[\left(\mathbf{R}p_i + \vec{T} - q_i \right) . \vec{n_i} \right]^2 \tag{4.6}$$

  where $\vec{n_i}$ represent the normals to the plans estimated at each iteration. For this error metric, there is no closed-from solutions [Rusinkiewicz 2001].

- **plane-to-plane:** Segal et al. [Segal 2009] present the "Generalized ICP" as a generalization of the previous two metrics. This metric assumes that the points of the two clouds are locally Gaussian distributions. The maximum likelihood estimation is used to iteratively compute the transformation (R,T) minimizing the distances between pairs of matched points.



(a) The point-to-point metric aims to find the transformation that minimizes the sum distance between the matches' points of the two surfaces (i.e. $\|V_1\|^2 + \|V_2\|^2 + \|V_3\|^2$, where vi are the translation vectors linking source-to-target matches).

(b) The point-to-plan metric is designed to minimize the perpendicular distances from source points to the target surface tangent of their respective target points (i.e. $(V_1^T n_1)^2 + (V_2^T n_2)^2 + (V_3^T n_3)^2$ where n_i are the normal surface vectors)

Figure 4.1: *Different ICP error metrics [Al-nuaimi 2017].*

4.4.6 Minimization

Iteratively, the algorithm consists of computing the transformation matrix that minimizes the error metric. At each iteration, the algorithm provides an estimate of the transformation matrix, which will be used in the next iteration until the convergence of the algorithm. The latter converges when the error metric is less than a certain threshold.

- For the "point-to-point" error metric, closed form solution exist in order to estimate the rigid transformation. In the previous chapter, we gave some examples of solutions, as well as detailed SVD solution.
- The "point-to-plane" error metric is a non-linear optimization problem that can be solved using the Gradient descent, Gauss Newton (GN) or Levenberg-Marquardt algorithm. The Gauss-Newton method is often preferred because it avoids the costly calculation of Hessian matrices; we have outlined the details of this solution in the previous chapter. This metric also has a closed form solution after the linearization of the rotation matrix R. Note that the latter is a nonlinear function of the rotation angles $\alpha, \beta and \gamma$ around the three axes X, Y and Z, respectively. To linearize it, it is assumed that the rotation angles are small, so that the cos (θ) can be approached to 1 and the sin (θ) by θ. This suggests a reasonable hypothesis for closely shifted scans. The "point-to-plane" minimization has been proposed and derived by [Chen 1991]. The details of the derivations of the system after linearization are well exposed in the article [Low 2004] of Kok-Lim Low. We referred to this paper in our implementation.
- Regarding the "plane-to-plane" minimization [Segal 2009], it assumes that the points of the two clouds are locally Gaussian distributions. In other words, it exploits local planar patches in both point clouds which leads to plane-to-plane concept. The maximum likelihood estimation is used to iteratively compute the T-transformation minimizing the distances between pairs of matched points.

Other optimization techniques can be used, such as the genetic algorithms [Silva 2005], or simulated annealing [Luck 2000], to address the problems of local minima whose the ICP suffers from. Additionally,

various deep learning and regression techniques were proposed recently to improve registration techniques such as [Nicolai 2016, Pfeiffer 2017, Li 2017].

4.5 Implementation

After a few weeks of manipulation of ICP variants implemented in PCL, a careful study of the algorithm's functioning proved to be indispensable. Once we understood the mathematical logic of the technique, everything seems less complex and simpler to deal with. We started with the implementation of a single ICP technique, and then several implementations were chained afterwards.

In the following, we will present the variants of the ICP that we have implemented. Up to now, more than 200 variants of ICP are implemented and tested. The approach we adopt is to implement several strategies at each ICP stage. The variants will be formed from different combinations of each strategy implemented in each step of the algorithm.

4.5.1 Variant selection:

Table 4.1 summarizes the implemented strategies that serve as bricks for the composition of the different variants.

Table 4.1: *Variant selection.*

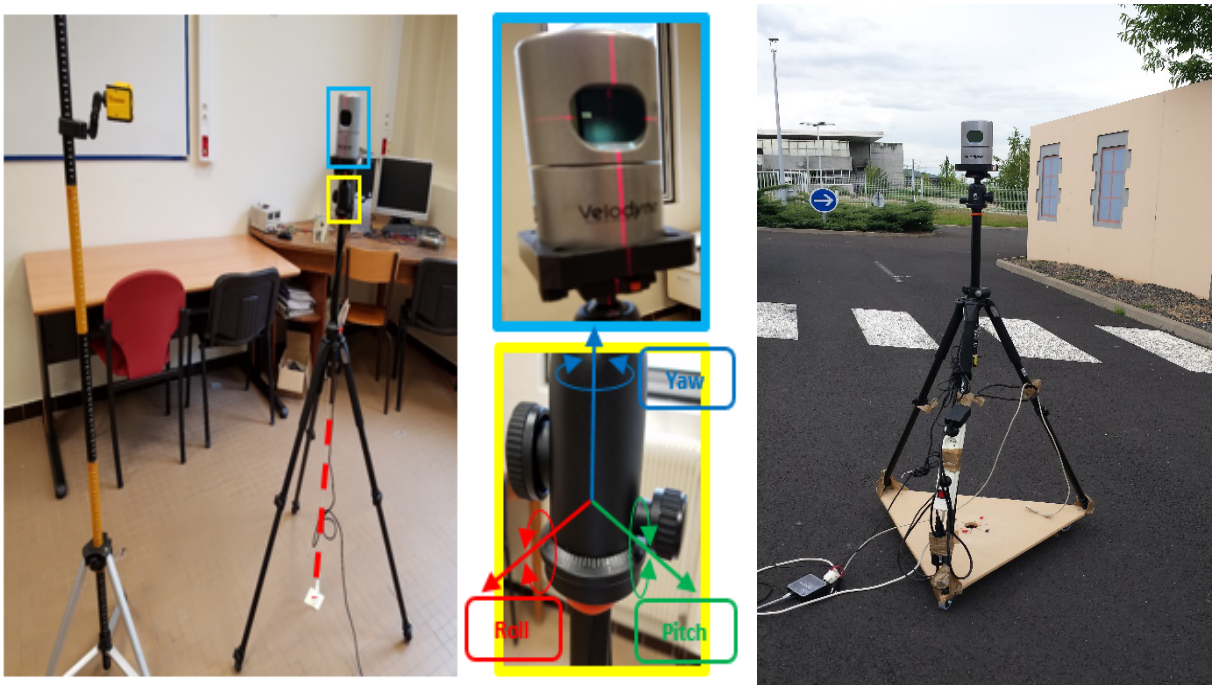
Algorithm steps	Strategies
Selection	– Use all the points – Uniform sampling - Random sampling – Outliers filtering – Statistical sampling – Normal-based sampling
Matching	– Nearest neighbors
Weighting	– No weighting – Huber – Tukey
Rejection	– No rejection – Distance rejection – Rejection based on the normals orientation
Error metric	– point-to-point – point-to-plane
Minimization	– SVD minimization - Gauss-Newton minimization

$$\begin{aligned} Total\ of\ variants &= C_6^1 \times C_1^1 \times C_3^1 \times C_3^1 \times C_2^1 \times C_2^1 \\ &= 6 \times 1 \times 3 \times 3 \times 2 \times 2 = 216 \end{aligned} \tag{4.7}$$

4.5.2 Used Dataset

The different strategies are tested with indoor and outdoor data taken using a Velodyne HDL-32E sensor mounted on a tripod at a height of 1m55 (Figure 4.2). Each point cloud contains approximately 70 000 points. Only the geometry is used for the registration, no color, and no intensity.

The experimental is set up as shown in Figure 4.2. The center of the sensor between two acquisitions is perfectly superimposed with the help of the STANLEY Cubix cross line laser. From one acquisition to another, the sensor is physically displaced and rotated by known translations and rotations from the graduated set up in order to perturb the 6 degrees of freedom transformation. Data acquisition is then performed under different scenarios in order to test the different variants.



(a) (b)

Figure 4.2: *Acquisition with the Velodyne.*

4.5.3 Evaluations metrics

The performance of each variants is evaluated using three metrics: the accuracy, the number of iterations required to the convergence of the algorithm, and the computation time.

- **The accuracy** The accuracy describes the evolution of the root-mean-square point-to-point distance; this

can be expressed mathematically as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|E_i\|^2} \tag{4.8}$$

where n is the number of points and is the distance error between the source points and its correspondent in the target cloud in each iteration. This can be expressed as follows:

$$E_i = \sum_{i=0}^{m} p_i - q_i \tag{4.9}$$

where m is the total number of points in the sparse cloud and which represent two points of the source and target cloud, respectively, whereby is transformed in the reference frame of.

- **The number of iterations** The maximum number of iterations for each test is set to 200. We consider that the algorithm has not converged if the maximum number of iterations is reached.
- **Computation time**

  The computation time for each variant quantifies the time of ICP iterations up to convergence. This means that the time of the point normals computation or sampling if it takes place is not included in this time.

4.6 Experimental results

4.6.1 Variants evaluations

We will examine the ICP variants implemented for each of the steps listed in section 4. We evaluate and compare the influence of each strategy on the convergence of the algorithm. In order to limit the scope of the problem, we choose a basic algorithm, and for each test, we only change the strategies of the stage concerned. This algorithm incorporates the following strategies:

- Use all the points + nearest neighbors + no weighting + no rejection + point-to-plane + Gauss-Newton minimization.

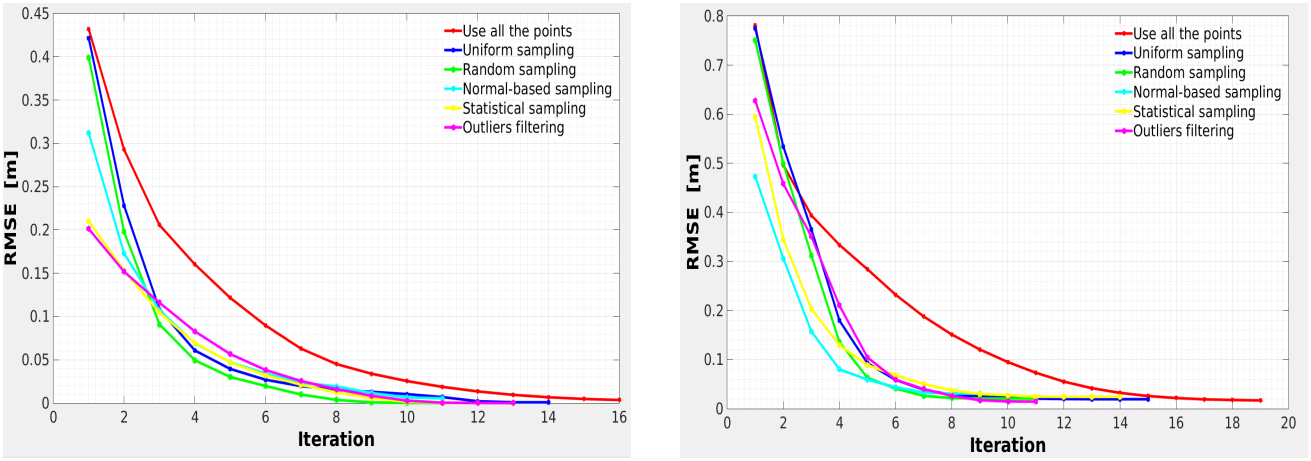
These implementations are accomplished in C++. They are tested on an Intel Core i7-4800MQ processor, 2.7 GHZ with 32 GB of RAM.

a. Selection strategies

The choice to use a selection strategy is motivated by the desire to optimize the computing time. Because, depending on the application, the registered clouds can become quite large (this is the case of the SLAM for example). On the other hand, most of the points are often redundant or useless for the registration task. Therefore, the use of sub-sets of points can give better results.

The uniform selection strategy is implemented using the voxelisation technique detailed in [Tazir 2016], taking a single point (voxel center) as representative of all the points of this voxel, similarly, for the random selection strategy that differs from the latter by choosing a random number of points between 0 and 3 as representatives. With regard to the sampling according to the orientation of the normals, we use the one implemented in PCL. The outlier filtering consists of deleting the points that do not contain a nearest neighbor within a specified distance threshold.

The statistical sampling method is based on the calculation of the distance distribution between each point and its neighbors. Assuming that the resulting distribution is Gaussian with a mean and a standard deviation, all the points whose mean distances are outside an interval defined by the average of the global distances can be considered as outliers and are therefore removed from the points cloud.



(a) Office scene (b) Pavin scene

Figure 4.3: *Convergence rate comparison of different selection strategies.*

As we can see in Figure 4.3, there is an improvement in the convergence profile and the number of iterations achieved at convergence. This shows that the point selection step is an important aspect of ICP. In addition, the appropriate level and type of selection are related to the scene and its representation, this can be seen by the difference of the results after selection between the two scenes, indoor (Figure 4.3(a)) and outdoor (Figure 4.3(b)). However, no strong preference is emerging among the alternatives.

The technique based on the normals orientation gives slightly better results, especially in the outdoor scene. This is because, it ensures the removal of points that do not share the same local geometries with its neighbors, which significantly improves the efficiency of the algorithm.

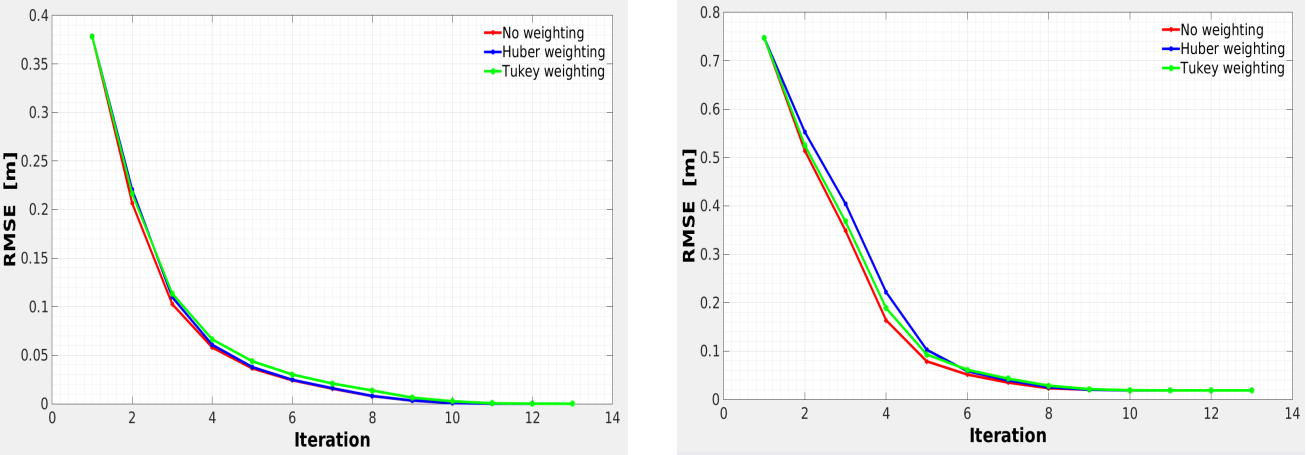
b. Matching strategies

Based on our bibliographic research, the best technique identified is the k-d trees (Section 4.4.2). We use the k-d trees implemented in PCL [Rusu 2011] directly, which is based on the FLANN library [Holz 2015]. The correspondence is obtained by finding, for each point of the source cloud, the nearest point in the target cloud. This is accomplished using L_2 norm

c. Weighting strategies

Here, we study the effect of assigning different weights to the pairs of points found by the previous two steps. We consider three different strategies for assigning these weights:

- no weighting,
- Huber,
- Tukey.



(a) Office scene (b) Pavin scene

Figure 4.4: *Convergence rate comparison of different weighting strategies.*

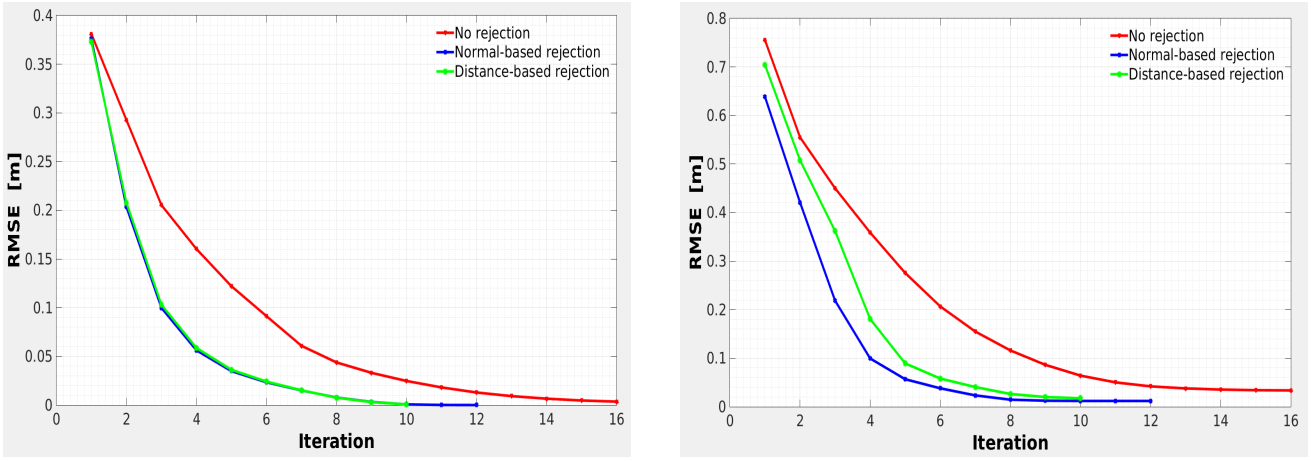
However, the tests (Figure 4.4) showed a minimal influence of these strategies on our data, whether on indoor or outdoor. Consequently, on our implementation, we do not use any weighting strategy. This is the same conclusion as found in [A. Donoso 2016], which affirms that the weighting stage might be removed from the ICP algorithm.

d. Rejection strategies

Figure 4.5 compares the performance of the algorithm when:

- rejection is not used,
- distance-based rejection is used,
- Normals-based rejection is used.

This test shows that variants that use a form of rejection, either based on distance or based on normal orientation, outperform the variant that does not use rejection in convergence speed and number of iterations. This reflects the importance of this step in the registration process.



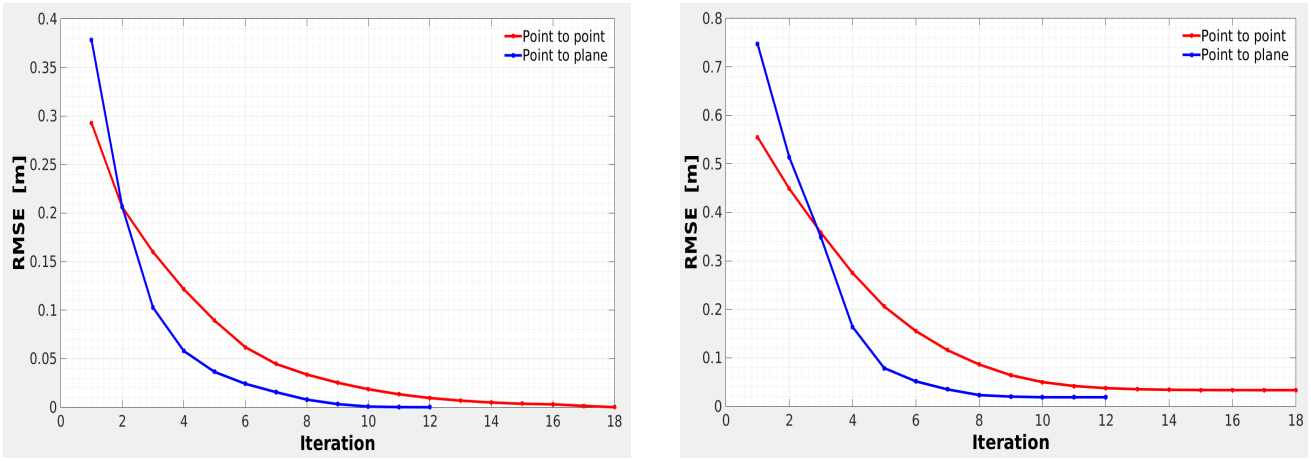
(a) Office scene (b) Pavin scene

Figure 4.5: *Convergence rate comparison for different rejection strategies.*

The performance of the two used techniques gives the same results in the indoor environment (Figure 4.5(a)). While in the outdoor environment (Figure 4.5(b)), it is the variants based on the normals orientation that gives better results.

e. Error metric

Two errors metrics (the most identified in the literature) are used in our implementations, point-to-point and point-to-plane. Figure 4.6 shows the convergence rate of two scans from indoor and outdoor environments with these two metrics.



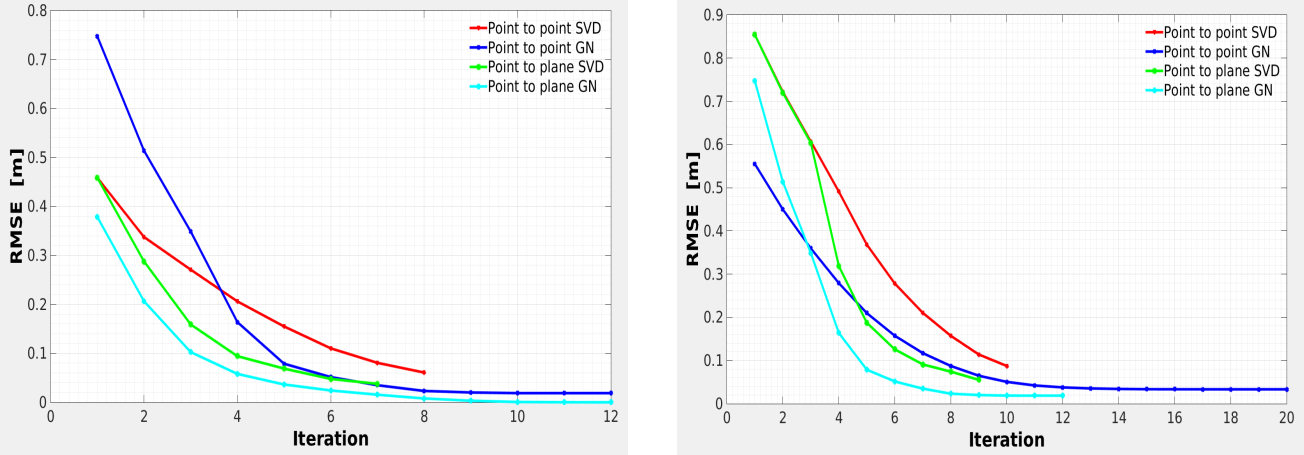
(a) Office scene (b) Pavin scene

Figure 4.6: *Comparison of convergence rate for the different metrics implemented.*

The convergence rate of the metric "point-to-plane" is better than "point-to-point". We can see that from its steeper convergence slope. The convergence speed (number of iterations) is also improved in the case of this metric. Moreover, the point-to-point metric performs poorly on the outdoor scene (Figure 4.6(b)). This is particularly apparent from the RMSE at the convergence, which is a bit higher in the case of this metric. This is perhaps due to the geometric variation of the content of this scene. While the point-to-plan metric, where points are matched with surfaces, allows the ICP algorithm to be more robust to these variations.

f. Minimization strategies

Two strategies are implemented: the SVD and the Gauss-Newton optimization. Both of these optimization techniques have been implemented for the previous two metrics.



(a) Office scene (b) Pavin scene

Figure 4.7: *Convergence rate comparison for different minimization strategies.*

Figure 4.7 shows that Gauss-Newton optimization is more robust than the SVD optimization.

4.6.2 Summary and observations

According to these tests, the main variant elected at the end is:

- Normal-based sampling + NN matching + distance-based rejection + point-to-plane + GN minimization.

In addition, there are several observations that could be drawn from these evaluations:

- There is no clear preference among point selection methods and weighting techniques.
- The sensitivity of ICP performance to the composition of the algorithm. This can be seen on the change of the convergence curves for each variant. The challenge is to identify those combinations of methods that are best performed across the both scenes.

- The performance of ICP variants depend on the scene

  This can be seen in the final convergence errors between the two environments. The indoor scene has always the accurate results. To further examine this observation, we conduct an experiment using the same different variations on indoor and outdoor data. Figure 4.8 shows the results of this experimentation based on the accuracy metric, which is represented here by the RMS error. According to this figure, a point close to the Y-axis represents a precise variation on the indoor data, and the same for a point that is close to the X-axis, which represents a precise variant on outdoor data. If the point is close to the two axes, this means that this variant performs well across the data of both environments.

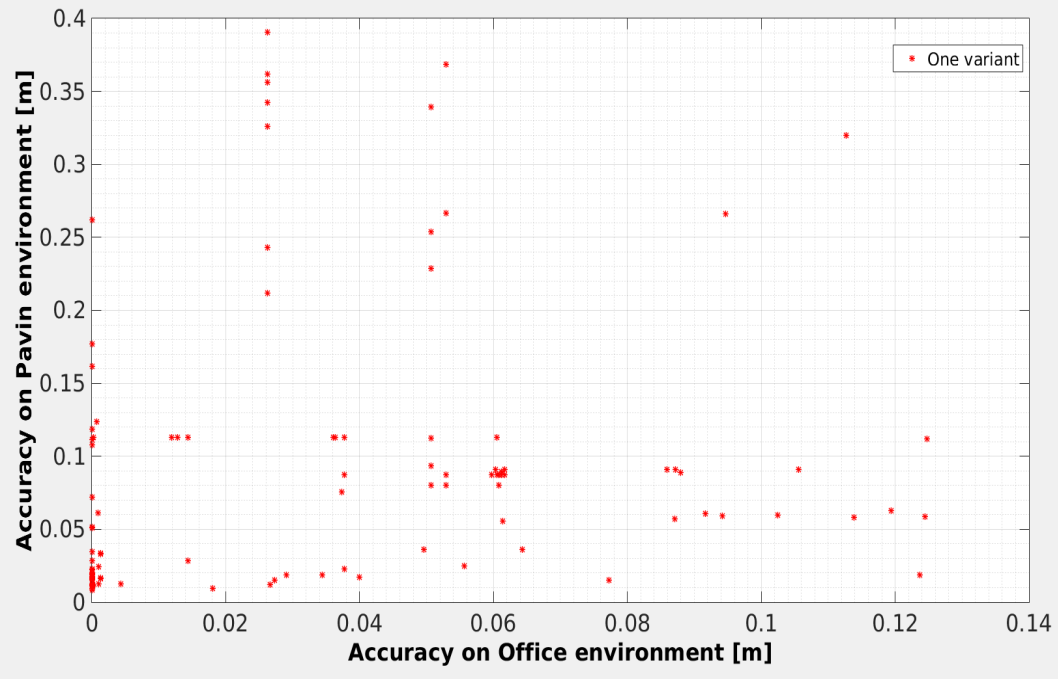


Figure 4.8: *Comparison of the performance of ICP variants on both indoor and outdoor environments in terms of accuracy performance metric. The variants that returned an accuracy less than 0.5 m on both scenes are presented.*

This test has reinforced the observations made before. In particular, that the variants which use a form of selection, a form of rejection, and based on point-to-plane metric are the most accurate.

4.6.3 Robustness tests with respect to translation and rotation

We have tested the selected variant from the previous test in different situations. The first test is relative to the pure rotation (with a change in rotation around a single axis). The second is with respect to the pure translation (here, it is only the translation along a single axis that changes), and the last is with respect to the change in rotation and translation at the same time.

For the sake of comparison, we have chosen another variant consisting of:

- Normal-based sampling + NN matching + rejection based on normal orientation + point-to-plane + SVD minimization.

The results in terms of number of iterations, computing time and quadratic mean error (RMSE) are presented in Tables 4.2, 4.3 and 4.4 respectively.

a. Test of robustness in translation

Table 4.2: *Test of robustness in Translation.*

	Point-to-plane SVD			Point-to-plane GN		
Offset (cm)	# iteration	time (s)	RMSE (m)	# iteration	Time (s)	RMSE (m)
10	4	***0.143***	0.000875	5	0.168	**0.000004**
20	6	***0.218***	0.000334	7	0.233	**0.000005**
30	7	***0.271***	0.000346	8	0.3	**0.000006**
40	8	***0.314***	0.000372	9	0.347	**0.000004**
50	9	***0.378***	0.000348	10	0.393	**0.000020**
60	9	***0.394***	0.000443	11	0.43	**0.000002**
90	11	***0.647***	0.000434	12	0.591	**0.000031**
150	14	***0.945***	0.000011	16	0.951	**0,000002**
200	15	***1.249***	0.000452	19	1.418	**0,000002**

The best results of the RMSE for all the methods are underlined in bold. Similarly, for the computation time, the best results are highlighted in bold italic. As already mentioned above, the maximum number of iterations for each test is fixed at 200, beyond which the algorithm is considered as not having converged (failed).

The results from the previous table show that the first variant based on point-to-plane metric and Gauss-Newton minimization is the most accurate. In terms of computational time, techniques based on Gauss-Newton optimization lose ground in front of SVD optimization based techniques and this is due to the costly calculation of the Jacobian matrices for the former. Figure 4.9 shows the case of a translation of 2 meters. The point to plane algorithm with Gauss Newton's minimization technique is able to align this critical situation with a quadratic error that do not exceeds the 0.02 millimetres.

b. Test of robustness in rotation

According to Table 4.3, methods based on the metric point-to-plane are highly influenced by the rotation. This is due to the influence of the rotation on the normals. This observation is justified by the fact that the variants based on the point-to-point metric are able to align two scans with a deviation of 75 degrees. This critical situation is shown in Figure 4.10.

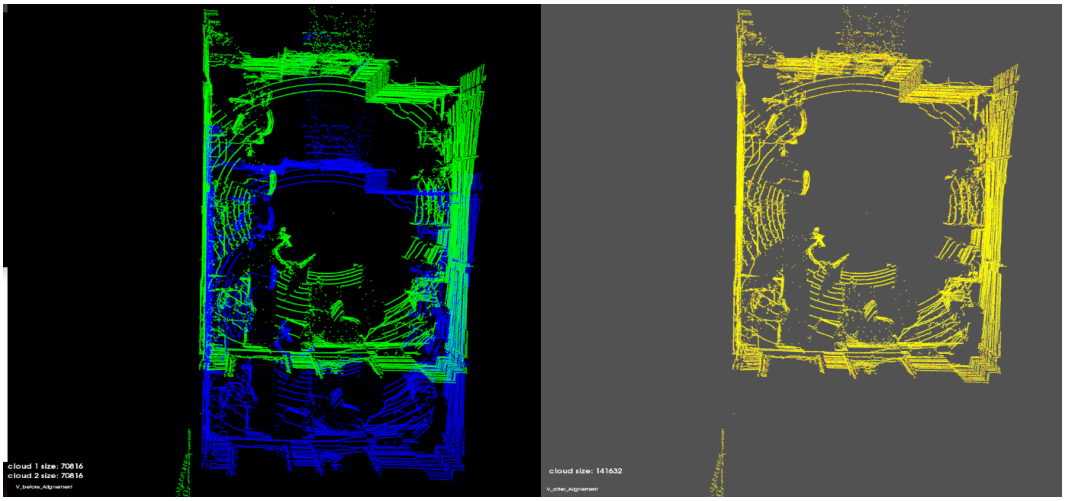


Figure 4.9: *Robustness in translation.*

Table 4.3: *Robustness in rotation.*

rotation	**Point-to-plane SVD**			**Point-to-plane GN**		
Degrees	# iteration	time (s)	RMSE (m)	# iteration	Time (s)	RMSE (m)
5	5	***0.183***	0.001230	6	0.21	**0.000007**
20	8	***0.354***	0.001574	14	0.556	**0.000006**
30	10	***0.5***	0.000017	20	0.807	**0.000006**

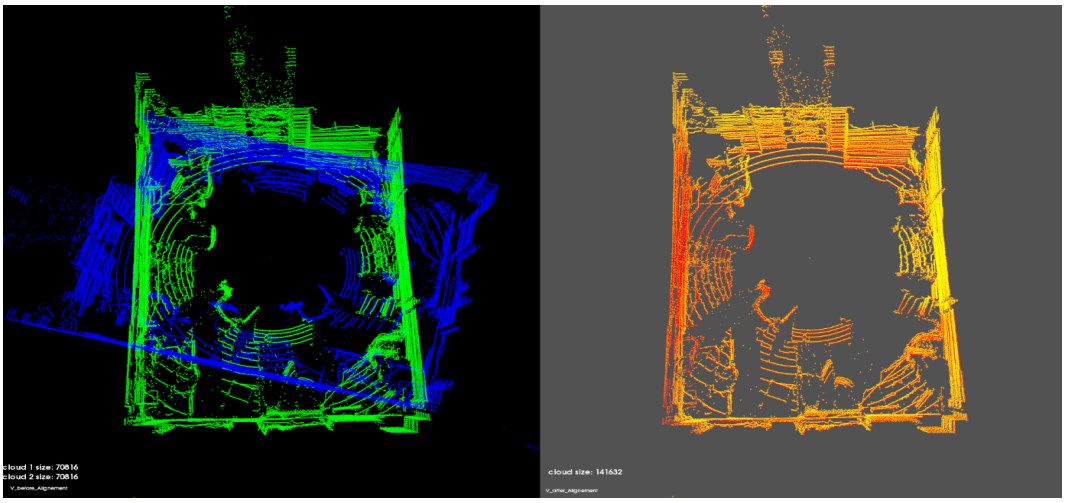


Figure 4.10: *Robustness in rotation. Case of 75 degrees of deviation. The variant based on point-to-point metric with Gauss Newton's minimization technique is able to register this critical situation with a quadratic error of 2 millimetres.*

c. Test of robustness in translation + rotation

In the case of the variation in translation and rotation, the registration process is harder, because the minimization is done on 6 degrees of freedom (DoFs), three for the rotation and three for the translation. In the case of Table 4.4, we vary only two DoFs, a translation along the X-axis and rotation around the Z-axis.

Table 4.4: *Robustness in translation + rotation.*

	Point-to-plane SVD			Point-to-plane GN		
Offset	# iteration	time (s)	RMSE (m)	# iteration	Time (s)	RMSE (m)
10 cm 5°	5	***0.174***	0.056456	8	0.246	0.040301
20 cm 10°	8	***0.296***	0.087945	13	0.419	0.056314
30 cm 15°	13	***0.5***	0.132591	20	0.717	**0.068863**
40 cm 20°	16	***0.685***	0.153472	23	0.941	**0.0717**
50 cm 25°	15	***0.877***	0.175892	26	1.491	**0.079288**
60 cm 30°	failed			29	***1.694***	**0.088925**
70 cm 35°				27	***1.819***	**0.095609**

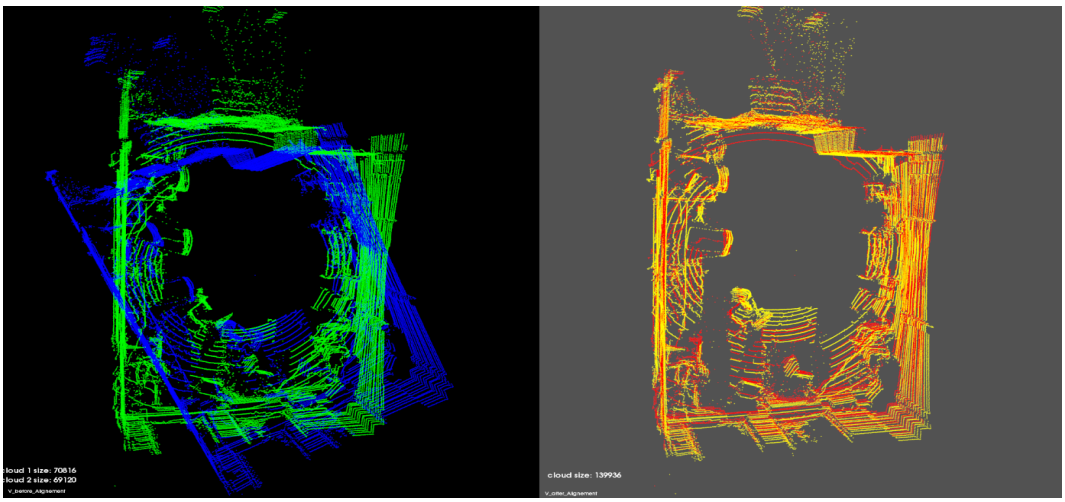
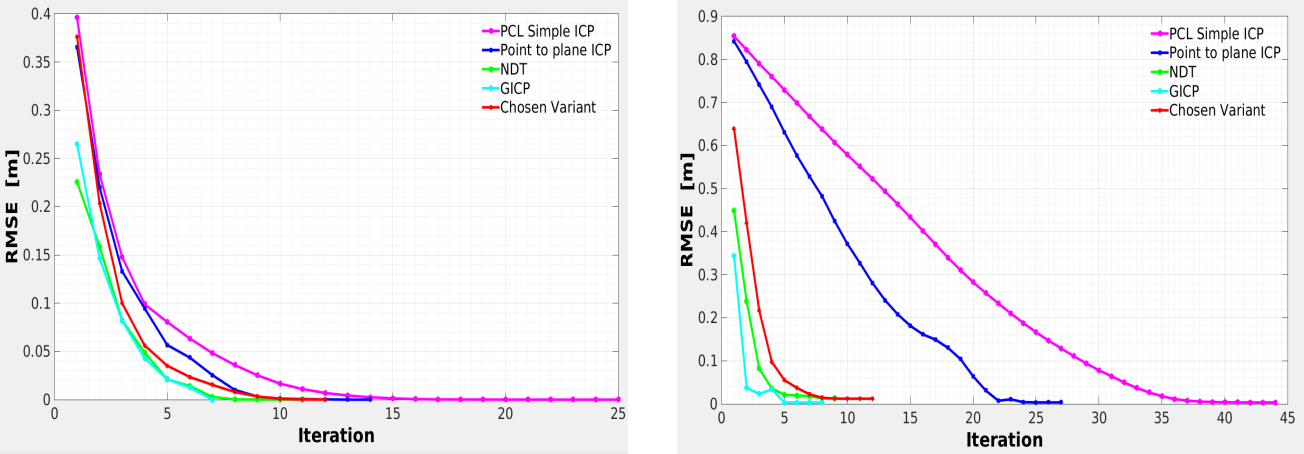


Figure 4.11: *Robustness in translation + rotation.*

The previous figure (Figure 4.11) shows the results of the registration of the point-to-plane GN algorithm in the case of a translation of 50 cm and a rotation of 25°. The final RMSE, in this case, is 8 centimetres.

4.6.4 Comparison with PCL Variants

Order to compare the chosen variant with the existing state-of-the-art methods, we use implemented routines of PCL library for the NDT, GICP, ICP point-to-plane, and ICP point-to-point. Figure 4.12 shows this comparison.



(a) Office scene

(b) Pavin scene

Figure 4.12: *Convergence rate comparison of implemented PCL routines against the chosen variant.*

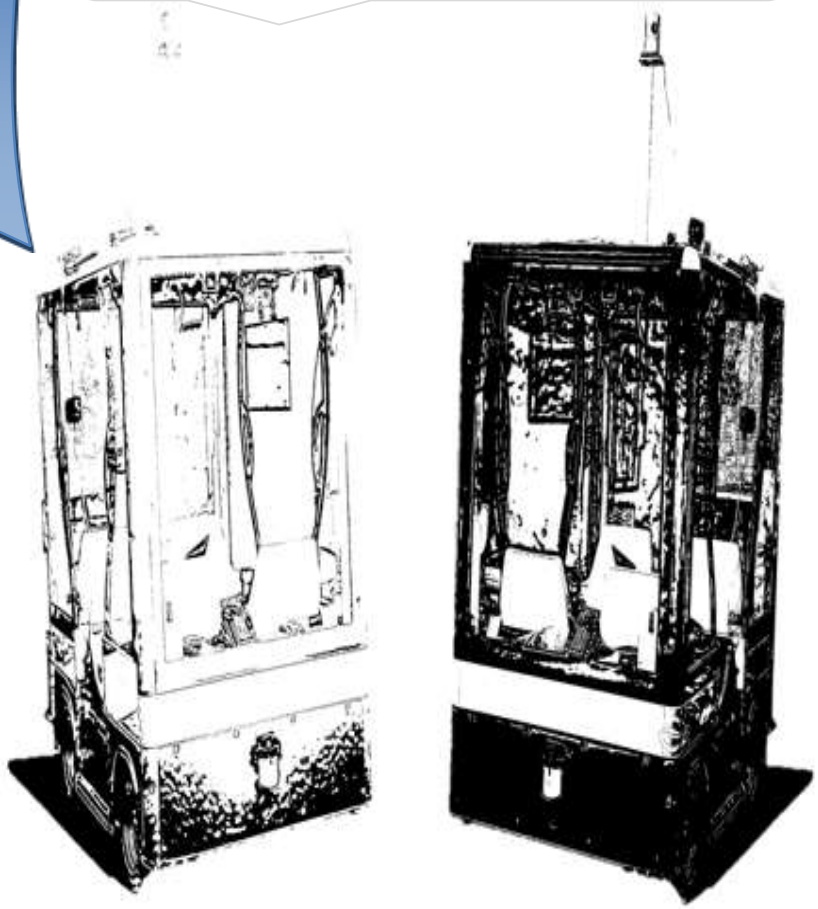
According to this comparison, the good results are given by the GICP and NDT. However, there is no significant difference between the chosen variant and these two algorithms. As a result, this variant will be chosen to build the incremental map. In addition, we prefer use our own implementation since we have more control over the internal functioning of the algorithm, instead of using a library with parameters sometimes enforced.

4.7 Conclusion

Through this chapter, we have shown our implementations of the ICP variants, supported by some results that have been used to validate these implemented techniques. This implementation issue seems crucial to my thesis. In concrete terms, this part of my thesis allows me to say that the ICP algorithm does not present a single solution, but rather a composition in which multiple variations and algorithms can be used to solve the problems of registration. In other words, there is not a single variant that can be accurate and reliable for all type of environments and scenarios. However, the best-performing variant used a strategy of outliers filtering, used local surface geometry in the rejection step, used the distance between points in the source cloud and the corresponding surface of target cloud as a measure of the fit between the two point clouds, and used a Gauss-Newton minimization. This variant will be used later for the construction of the reference maps.

Chapter V

CICP: Cluster Iterative Closest Point



CHAPTER 5

CICP:Cluster Iterative Closest Point

Contents

Throughout the years, registration has been addressed in different ways, based on local features, global descriptor or object-based. However, all these techniques give meaningful results only if the input data are of the same type and density (resolution). Recently, with the technological revolution of 3D sensors, accurate ones producing dense clouds have appeared as well as others faster, more compatible with real-time applications, producing sparse clouds. Accuracy and speed are two sought-after concepts in every robotic application including

those cited above, which involves the simultaneous use of both types of sensors, resulting in sparse–dense (or dense–sparse) point cloud registration. The difficulty of sparse to dense registration lies in the fact that there is no direct correspondence between each point in the two clouds, but rather a point equivalent to a set of points. In this chapter, we present the Cluster Iterative Closest Point for sparse to dense point clouds registration. The main novelty of CICP consists in matching points representing each local surface of source cloud with the points representing the corresponding local surfaces in the target cloud. Experiments and comparisons with state-of-the-art methods show that CICP gives better performance. It handles registration of point clouds of different densities acquired by the same sensor with varied resolution or taken from different sensors.

5.1 Introduction

The problem of dense-sparse registration has received less attention from the scientific community in the past [Agamennoni 2016]. The majority of research has been focused on dense registration [Bellekens 2014, Pomerleau 2015], or sparse registration [Velas 2016, Zhang 2015, Razlaw 2015, Grant 2013]. Recently, the need for sparse to dense registration has come to the limelight, and this is due to the emergence of sensors that produce sparse data like Velodyne [1] LiDAR (Light Detection And Ranging), which is widely used in autonomous vehicles (Google car [Patidar 2016], DARPA Grand Challenge [Thrun 2007]), because of its ability to provide 3D data at a high refresh rate and at a long range [Grant 2013]. Accurate sensors producing dense clouds also achieved a technological leap with the appearance of 3D laser sensors like Leica P20[2], Riegl VZ400i[3] or Trimble TX8[4], etc. Furthermore, multiple sensors that allow the change of scanning resolution have recently appeared on the market. These sensors can produce point clouds of different densities depending on the chosen resolution. Nevertheless, the difference in cloud density is generally due to the change of the sensor. For example, two different sensors generate two clouds with different point densities, which requires a calibration step between the two sensors in order to exploit the resulting clouds. Calibration is necessary whenever the two sensors are moved, which is a hard and tedious task. On the other hand, the main shortcomings of available point cloud registration methods are their lack of speed due to the increase of input data or their lack of precision due to the decrease in density [Grant 2013] whereas, for most robotic applications such as localization, these two attributes are highly desired. A recent trend is to use both kinds of sensors [Wolcott 2014, Caselitz 2016, Maddern 2016] to achieve these two sought-after concepts simultaneously, highlighting the importance of dense to sparse registration techniques.

As with all registration methods, an overlapping between the two clouds, usually called source cloud and target cloud, is necessary to determine the rigid transformation between these two clouds perceived from different viewpoints. With the conventional methods that use coherent data from the same sensor, what changes are the representation of points pertaining to the two views. For sensors that also provide intensity or color, these two attributes can be changed if the two clouds are acquired at two different times. Otherwise, except for the noise, nothing else can be changed, neither the number of points nor the spacing between the points. Regarding dense-sparse data, the degree of sampling changes, affecting the number of points and the distance separating these points. In other words, for the same part of the scanning environment, taken from two different viewpoints, the dense cloud will exhibit a larger number of points with a smaller distance separating them, as opposed to the sparser cloud. This change affects all the local characteristics of the points (normals, curvatures), making the conventional methods unsuitable for this type of registration [Holz 2015, Razlaw 2015].

Recently, Agamennoni et al. [Agamennoni 2016] addressed the sparse-dense registration issue and proposed a method that improves the standard point-to-point ICP [Chen 1991, Besl 1992] by introducing a probabilistic model for data association. The main idea of this work is to align each point from the sparser cloud with a set of points from the denser cloud. The association with each point is weighted taking into account the uncertainty of the transformation estimate. The problem is formulated as an Expectation-Maximization procedure, during

[1] Velodyne LiDAR: http://velodynelidar.com/

[2] Leica P20: http://leica-geosystems.com/

[3] Riegl VZ400i: http://www.riegl.com/nc/products/terrestrial-scanning/produktdetail/product/scanner/48/

[4] Trimble TX8: http://www.trimble.com/3d-laser-scanning/tx8.aspx

which, weights are calculated throughout the E-step, whilst during the M-step, the rigid transformation is updated from current associations. However, the weakness of this method is that the associations do not change at each iteration. The iterations serve only to optimize the weights. That is why the method should be executed several times, using as input the solution of the previous run, which consumes a lot of time. Moreover, the fact that the point associations do not change at every iteration, makes this method very sensitive to the initial data association.

In this work, we propose a method to align dense and sparse clouds to achieve accuracy and convergence speed. This method surpasses the notion of density by replacing the points sharing the same local surface of the two clouds by a single representative point for the matching step. It is not about sampling, but only for the matching process, the points most likely to match each other are selected. Then, the resulting transformation is used to transform all the source points. The process evolves iteratively in an ICP-like framework, starting with a selection process followed by a pose estimation process. The main contribution of this novel approach lies in the selection points for the matching process. First, a voxelization is performed on both clouds to maintain the topological details of the scene. Then for each voxel, a normal-based classification of its points is done. Thereafter, only one point of each local surface is maintained for the associating step. As a result, fewer points are used for the matching process, but they are most likely to be associated. Thereby, improving convergence and accuracy simultaneously.

5.2 Related Work

In the previous chapter, we have provided a complete state of the art of the registration technique. From this state of the art, three main categories are distinguished, which are feature-based methods, dense methods, and object-based methods.

CICP differs from these three sub-categories in the nature of its data and how these data are used. As input, it takes point clouds of different resolution, gathered by different sensors, or with the same sensor. It is based only on the geometric information of points, which makes it independent of weather and illumination conditions. The proposed approach aims to cluster points of the same surface as one topological pattern, and replace all the points held by this model by one representing point for the matching step. The main algorithm is based on point-to-point matching alignment.

Table 5.1 summarizes the main differences between the proposed method and the three sub-categories identified in the prior related work (4.2).

5.3 Contributions

In this chapter, a novel approach for sparse to dense (or dense to sparse) registration is introduced, exploiting normals differently. The desired contributions are as follows:

1. A new selection strategy is proposed by keeping only points which are most likely to be associated in the matching phase, thereby improving on convergence and accuracy simultaneously.

Table 5.1: *Main differences between CICP and the state-of-the-art methods.*

	Dense methods	Sparse methods	Object-based methods	CICP
No prior information required	x [Magnusson 2007] [Serafin 2015]	✓ [Holz 2015] [Pandey 2011]	✓ [Lenac 2017]	✓
Use all points	✓ [Serafin 2015]	x [Rusu 2010] [Aldoma 2011]	x [Velas 2016]	✓
No risk of loss of good data	✓ [Chen 1991] [Besl 1992]	x [Mellado 2014] [Zeng 2016]	x [Grant 2013]	✓
Based on geometric information	✓ [Yang 2013]	✓ [Feng 2016] [Rusu 2008] [Aldoma 2011]	✓ [Serafin 2016]	✓
Use of normals in correspondences choice	✓ [Segal 2009] [Chen 1991] [Serafin 2015]	✓ [Zhong 2009] [Pandey 2011]	✓ [Fernández-Moral 2016]	✓
Compress point cloud in a set of distinct elements	not concerned	x [Rusu 2009b]	✓ [Dubé 2016] [Fernández-Moral 2013] [Li 2016]	✓
Not environment specific	✓ [Magnusson 2007]	x [Cadena 2016]	x [Salas-Moreno 2013] [Fernández-Moral 2016]	✓
Not affected by imperfect segmentation	not concerned	not concerned	x [Fernández-Moral 2013] [bib 2015]	✓
Does not require a very dense cloud	✓ [Razlaw 2015]	x [Sehgal 2010]	✓ [Serafin 2016]	✓

2. The proposed method is totally independent of the density (amount of points, scanning resolution) of the two clouds, scanning patterns (nature of sensors). It takes as input point clouds of different resolution, gathered by different sensors, or with the same sensor.

3. A novel mathematical definition of sparse and dense concept is proposed to accomplish these objectives.

All the considerations outlined in this section will be demonstrated in the results section and these claims are further consolidated in the Discussion section (cf. Section 5.7).

5.4 General Formulation

5.4.1 Mathematical Definition

Dense and sparse are terms used to describe the state of points within a cloud. This includes their quantity, distribution and resolution. The distinction between these two terms is rather vague, and depends on the context.

Suppose we have two clouds of the same environment, with the same dimensions and taken from the same viewpoint, they will probably have been taken by different sensors or by the same sensor with varied resolutions. Let us assume that there is a large degree of variation between their densities. The dense cloud will exhibit a larger number of points with a smaller distance separating them, as opposite to the sparser cloud. The concept of density in 3D is always linked to a given volume. To get the same volumes, the two clouds are divided into voxels (subdivisions) of the same size. At this stage, the main clouds integral characteristics are the set of voxels V and the set of points P. The relation between these two sets determines whether the cloud is sparse or dense.

Below, we give some basic definitions, and we introduce the "voxelic density" definition in theoretical and practical cases:

Definition 1 (Voxel in $\mathbb{R}^3$)**.** *A voxel v with center $\omega = (x_0, y_0, z_0)$ and rayon r, is a set of points $P(x, y, z)$ if:*
$v = \{(x, y, z) : \max\{|x - x_0|, |y - y_0|, |z - z_0|\} <= r\}$

Definition 2 (Set of all voxels in P)**.** *Let v a voxel of V_P, we say that V_P is a set of all voxels in P if:*
$V_P = \{\forall v \in V_p, v \subset P\}$

Theoretically, a dense cloud is:

Definition 3 (Voxelic density)**.** *P is dense $\leftrightarrow$*
$\forall v \in V_p, \exists p \in P : p \in v$

According to this last definition, a point cloud is called dense, if and only if, there is always at least one point belonging to a voxel, whatever the voxel size.

Unfortunately, for practical reasons, this definition is not always verified. Because of this, we propose definitions 4 and 5:

Definition 4 (Sparse Cloud)**.** *A sparse cloud is a cloud $C = (V, P)$ in which:*
$|P| = \mathcal{O}(|V|)$.

Definition 5 (Dense Cloud)**.** *A dense cloud is a cloud $C = (V, P)$ in which:*
$|P| = \mathcal{O}(k.|V|)$, *with* $k > 2$.

$\mathcal{O}$: proportionality operator.

Definitions 4 and 5 are proposed to frame the notions of sparsity and density of point clouds. The voxel size is set according to the number of points in the sparse cloud, so that each voxel contains at least one point. This choice ensures a significant difference in density between the two clouds. A dense cloud, in our case, contains at least twice as many points as the sparse cloud. Otherwise, they are considered as equivalent.

5.5 Proposed Method

This work proposes a novel approach that deals with dense-sparse registration. We adopt the Rusinkiewicz decomposition and propose a new selection strategy, which aims to improve the pairing process and make it reliable for the purpose of this technique. Figure 5.1 illustrates the workflow of the proposed method.

CICP starts with the estimation of normals of the two clouds. Then, it takes the target cloud first and subdivides it into small voxels. The points of each voxel are subjected to a classification process based on their normals, giving rise to different groups of points, according to the geometric variation of each voxel. Each group of points represents a local surface since they share the same normal vector. Next, from the points of each small local surface, a single point is chosen to represent this surface during the matching process. In our case, we take the closest point to centroid of each local surface. Regarding the source point cloud, its points are transformed with their normals by the initial guess of the relative transformation. Then, this cloud undergoes the same steps as the target cloud: voxelization, normals-based classification, designation of point's representatives. At the end of these steps, the method comes up with two sets of points from the two clouds. Each set contains the most probable points to match with the points of the second set (this is specifically in the overlapping area of the two clouds, as it reflects the same geometry seen from two different viewpoints).

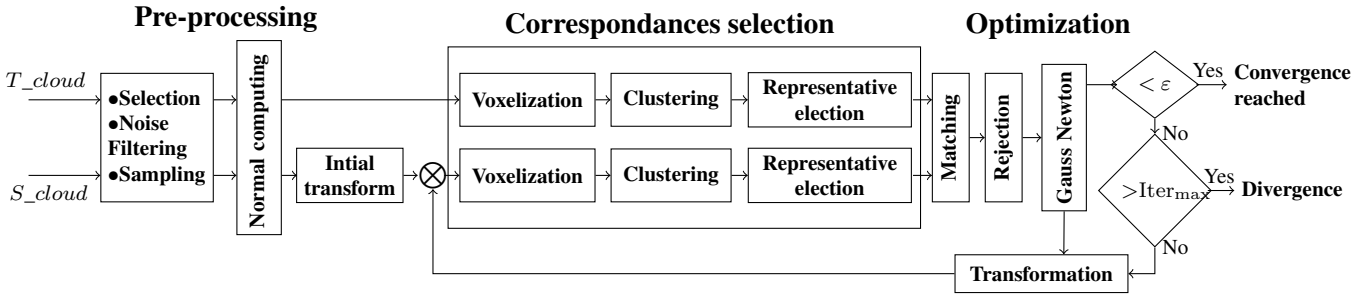


Figure 5.1: *Overview of the CICP pipeline. Given two point clouds, CICP starts by computing the surface normals of the two clouds. It looks for points sharing the same local properties, and then elects one representative point from each local cluster. This election process is based on 3D position of points and their normals. It consists of three sub-tasks: (1) Voxelization: a set of 3D cubic regions (voxels) is generated where all voxel points have very close spatial positions. (2) Clustering: classify all points of each voxel according to their normals. (3) Representative election: once this grouping step is completed, the last task consists in selecting one point from each cluster (local surface) in each voxel. Representative points serve as candidates for correspondence process. As a result, few points are used in the matching process, but which are most likely to be associated, thereby, improving on convergence and accuracy simultaneously.*

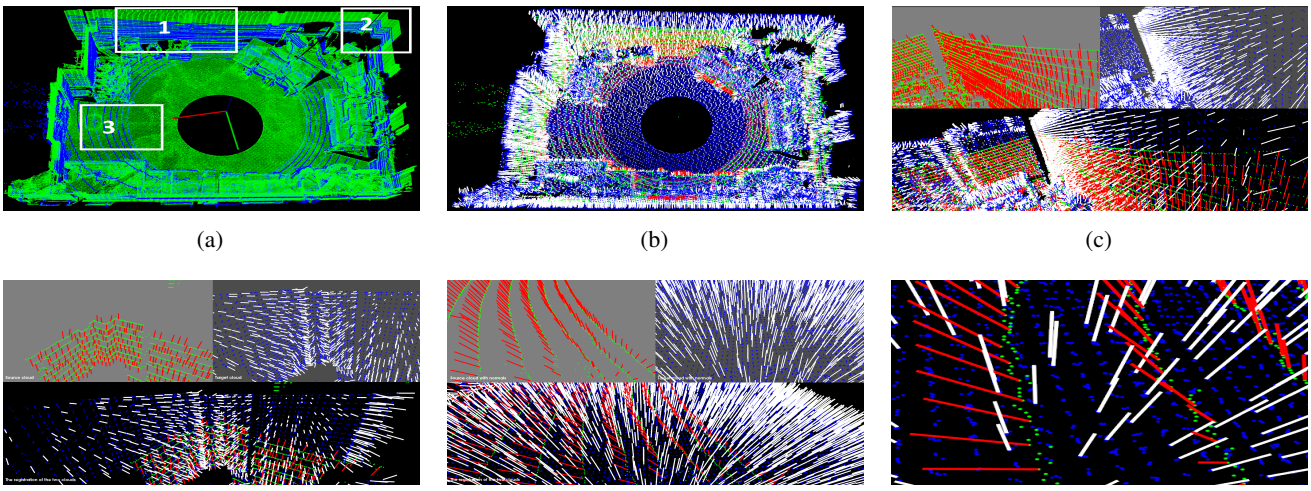
5.5.1 Selection

The main contribution of this approach is the proposal of a new selection strategy. As mentioned above, instead of matching point-to-point as the classical ICP variants, points pass through an election process, which gives rise to one representative point for each small region. These representatives appear as the most likely points to be matched between each other. These good matches ultimately result in an accurate motion between the two clouds. This election process is based on 3D position of points and their normals. It consists of three sub-tasks: (1) voxelization, (2) clustering and (3) Representative election. The first task performs a spatial grouping which attempts to preserve the topological information based on the 3D position of the points. A set of 3D cubic regions (voxels) is generated where all points within the voxel have very close spatial positions. The second task bundles all points of each voxel based on their normals. Once this grouping is done, we perform the last task, which selects one point for each cluster (local surface) in the voxel for the matching process. But before that, normals need to be calculated.

5.5.1.1 Normal Estimation

Normal segmentation of geometric range data has been a common practice integrated in the building blocks of point cloud registration. Most well-known point to plane and plane to plane state-of-the-art registration techniques make use of normal features to ensure a better alignment. However, the latter is influenced by noise, pattern scanning and difference in densities. Consequently, the resulting normals in both a source point cloud and a target point cloud will not be perfectly adapted, thereby influencing the alignment process, due to weak inter-surface correspondences. In order to support these claims, an illustration of sparse to dense registration is given in Figure 5.2. A dense point cloud is obtained from a 3D LiDAR Leica P20 scanner whilst the sparser one is extracted from an HDL-32E Velodyne. Figures 5.2 (c), (d), (e), (f) are samples of various places in a scene. The 3D points of the source and target clouds are represented in blue and green respectively, whilst their normals are in white and red. These figures depict the dissimilarity between normals pertaining to the same surface, which theoretically

should have the same orientations. This change is due to the different disturbances mentioned above. This is the major problem of the methods that use geometric features according to [Serafin 2016, Holz 2015]. Additionally, according to the authors of [He 2016], the calculation of normals on a large dataset is computationally expensive. To overcome this problem, we use normals only to distinguish the different local surfaces (group each surface



(a) (b) (c)

(d) (e) (f)

Figure 5.2: *Dense to sparse registration: (a) registration of a dense point cloud obtained from the Leica P20 LiDAR with* $88\,556\,380$ *points and a sparse point cloud obtained from an HDL-32E Velodyne with* $69\,984$ *points using our proposed method; (b) normal vectors corresponding to (a); (c), (d) and (e) are exploded views of places indicated in (a); (f) is a close up view of (e).*

alone). For the rest of the algorithm, we use x, y and z coordinates of each point without having recourse to their normals. In other words, we use normals only to distinguish the different surfaces, but we do not use them in the alignment process.

Normals are computed once for each cloud at the beginning of the algorithm. Source normals are transformed at each iteration by the transformation found. We use Principal Component Analysis (PCA) [Jolliffe 1986] to determine normals vectors of point cloud, as it is the most performant method used to compute normals according to [Donoso 2017] and [Klasing 2009]. PCA-based algorithm is usually used to analyze the variation of points in the three directions. Normal vector corresponds to the direction with minimum variation. We can also imagine the use of the dominant directions of the points as a characteristic of designation of the cluster within each voxel, instead of normals. We have chosen to use normals, as they are classical and common features.

From the eigen decomposition of the covariance matrix of considered nearest neighbors, the eigenvector corresponding to the minimum eigenvalue represents the normal vector. The covariance matrix can be calculated from the following equation:

$$\mathbf{C} = \frac{1}{k}\Sigma_{i=1}^{k}(p_i - \bar{p})^T(p_i - \bar{p}) \tag{5.1}$$

where k is the number of considered nearest neighbors; p_i, $i = 1 : k$ are kNN points and $\bar{p}$ is the mean of all k neighbors.

Algorithm 2 shows how to calculate the normals with the PCA method.

Algorithm 2: Compute normals with PCA.

Input: pointXYZ P, num_neighbors
Output: vector normals
Initialize: vector normal, vector neighbors, vector $\bar{P}$, matrix Q, H, U, V
begin
  foreach *point p in P* **do**
    // Extract the neighbors
      $neighbors = nearestKSearch(p, num\_neighbors, P)$
    // Calculate the centroid of neighbors
      $\bar{P} = sum(neighbors)/num\_neighbors$
    // Compute the covariance matrix
      $Q = (neighbors - \bar{P})$
      $H = Q.transpose() * Q$
    // Compute the eigenvectors
      $[\text{U, V}] = svd\_decomposition(H)$
    // Sort the eigenvectors by decreasing eigenvalues
      $U = sort\_decrease(U)$
    // Extract the normal
      normal $[0] = U(0, 2)$
      normal $[1] = U(1, 2)$
      normal $[2] = U(2, 2)$
    // Stack normal in container
    $normals.emplace\_back(normal[0], normal[1],$
    $normal[2])$
  end
  return *normals*
end

5.5.1.2 Voxelization

The voxelization is applied in order to maintain the topological details of the scrutinized surface. As normal computation depends on the number of neighbouring points and as the resolution of points of the two clouds is different, voxelization with the same voxel size aims to generate equivalent local regions in the two clouds. A common criterion of comparison now becomes feasible. Therefore, the voxel size parameter is of paramount importance for our technique and it should be chosen carefully in order to keep the fundamental characteristics of both point clouds; be it dense or sparse with topological details. In our case, the set of rules mentioned previously in the Section 5.4.1 must be verified.

At the beginning, the procedure applies a bounding box to the entire sparse cloud by finding the minimum and maximum positions of points along the three axes X, Y and Z. The number of voxels for this bounding box is determined by the number of points and the voxel size is deduced. The same procedure is applied to the dense cloud. For more details, see voxelized point clouds in Algorithm 3.

1. Voxel assignment: each voxel is identified by a unique linear index. If i, j, k represent the voxel indices in the X, Y, Z dimensions, respectively, $numDivX$, $numDivY$ are the number of voxels along X and Y axes, the formula to encode the linear index [Tazir 2016] is:

$$idx = i + j \times numDivX + k \times numDivX \times numDivY \tag{5.2}$$

   According to (5.2), we assign an index idx to each point. This relationship allows direct access to the desired voxel, thereby avoiding a linear search [Wiemann 2014].

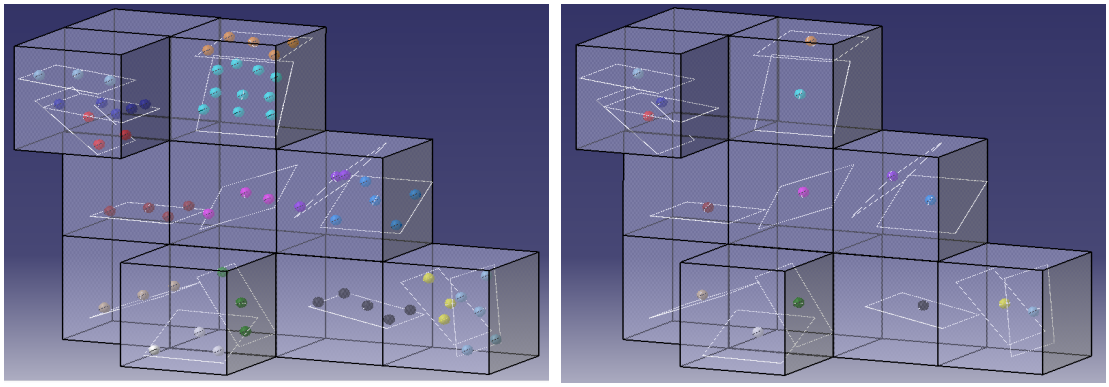
2. Voxels suppression: as the shape of the point cloud is arbitrary, the step of delimiting points by a bounding box creates many empty voxels which are later pruned out. Eventually, voxelization helps to filter noise from voxels where there is insufficient occupational evidence.

5.5.1.3 Clustering

The process of electing one point from each local surface makes them good candidates for point correspondence searching, thereby rejecting wrong matches impacting alignment accuracy. At first, all the "voxelized" points are taken and a classification method is applied to identify points belonging to the same surface. In our work, k-means clustering [Arthur 2007] is used as the classification technique based on the normal of each point. The appropriate number of clusters (local surfaces) within each voxel is determined using the Elbow method [Tazir 2016, Tibshirani 2001]. An illustration of the described approach is given in Figure 5.3.

Grouping the point clouds using their normal aims at:

- improving the robustness of the matching step by only allowing the pairing of compatible points,
- reducing the amount of data to be processed during the matching stage.



(a) Voxelized and clusterized point cloud

(b) Electing one point from each cluster for the matching phase

Figure 5.3: *Voxelized/normal-based clustering for matching process.*

Algorithm 3: Voxelization of a point cloud.

Input: pointXYZ P, voxelSize
Output: vector voxels
Initialize: minX = maxX = $P[0].x$, minY= maxY= $P[0].y$, minZ = maxZ = $P[0].z$, numDivX= numDivY =0
begin
  // Create a bounding box for all the points of P
  foreach *point p in P* **do**
    if $(p.x < minX)$ **then** $minX = p.x$;
    if $(p.x > maxX)$ **then** $maxX = p.x$;
    if $(p.y < minY)$ **then** $minY = p.y$;
    if $(p.y > maxY)$ **then** $maxY = p.y$;
    if $(p.z < minZ)$ **then** $minZ = p.z$;
    if $(p.z > maxZ)$ **then** $maxZ = p.z$;
  end
  // Calculate number of voxels along each axis
    numDivX = $(maxX - minX)/voxelSize + 1$
    numDivY = $(maxY - minY)/voxelSize + 1$
    numDivZ = $(maxZ - minZ)/voxelSize + 1$
  // Assign each point p to its corresponding voxel
  foreach *point p in P* **do**
    $i = (p.x - minX)/voxelSize$
    $j = (p.y - minY)/voxelSize$
    $k = (p.z - minZ)/voxelSize$
    $idx = i + j \times numDivX + k \times numDivX \times numDivY$ (5.2)
    voxels $[idx].push\_bach(p)$
  end
  // Erase empty voxels
    voxel.erase $(remove\_if\ (voxels.begin, voxels.end, container\_is\_empty), voxels.end)$
  return *voxels*
end

5.5.2 Matching

The clustering process generates a reduced, but different number of points in both clouds. These two sets of points are used for matching. Based on our bibliographic research, the best technique identified is the k-d trees (Section 4.4.2). We use the k-d trees implemented in PCL [Rusu 2011] directly, which is based on the FLANN library [Holz 2015]. The correspondence is obtained by finding, for each point of the source cloud, the nearest point in the target cloud. This is accomplished using L_2 norm. Since the number of points used for matching is different in the two pairing sets, there will be some wrong correspondences. This is handled in the rejection step, which aims to reduce these false matches.

5.5.3 Weighting

The aim of weighting is to reduce the influence of outliers on the alignment process. We tested two weighting strategies: Huber weighting [Huber 2009] and Tukey weighting [Huber 1981]. However, the tests showed a minimal influence of these strategies on our data. Consequently, on our implementation, we do not use any weighting strategy. This is the same conclusion as found in [Donoso 2017], which affirms that the weighting stage might be removed from the ICP algorithm.

5.5.4 Rejection

We opted for the distance-based rejection [Weber 2015, Costa 2016], as it is the most basic way to eliminate the wrong pairings. This simple and powerful strategy consists in eliminating the correspondences which have distances greater than a given threshold [Zhang 1994, Costa 2016]. This aims to reject the pairs of points that do not contribute positively to the convergence of the algorithm, such as unpaired points (as the number of points used for matching is different in the two pairing sets) and outliers.

5.5.5 Error metrics

After the matching step, which results in a selection of an equivalent number of representative points in both clouds, the three metrics commonly exploited in the literature, namely point-to-point, point to plane and plane to plane, can be used. However, for the sake of simplicity, we use the point-to-point metric:

$$\mathbf{E} = \sum_{i=1}^{N} \|\mathbf{R}p_i + \vec{\mathbf{t}} - q_i\|^2 \tag{5.3}$$

5.5.6 Optimization

The optimization is used to determine the transformation from the set of finding pairs. Given a set of correspondences, rotation and translation between the two frames are calculated using Gauss-Newton iterative least square algorithm [Bjorck 1996] (Algorithm 4).

In the case of a point-to-point metric, the error function to be minimized is given by:

$$E\left(x\right) = \sum_{i=1}^{N} \|\vec{T}\left(\tilde{x}\right) p_i - q_i\|^2 \tag{5.4}$$

The localization problem of a sparse to a dense point cloud (or vice-versa) resolves to estimating the relative transformation $\vec{T}(\tilde{x})$ between point clouds $\{p, q\} : \forall\{p_i, q_i\} \in \mathbb{R}^3$. The principle of rigid body motion is applied where the transformation of a point tethered to a coordinate frame represent the whole compact body motion. For any point pair lying on the body, metric properties such as distances and orientation are preserved. This kind of body motion, discussed subsequently forms part of the special euclidean group $\mathbb{SE}(3)$.

Inter-frame incremental displacement is further defined as an element of the Lie groups applied on the smooth differential manifold of $\mathbb{SE}(3)$ [Blanco 2010], also known as the group of direct affine isometries. Motion is parametrized as a twist (a velocity screw motion around an axis in space), denoted as $\mathbf{x} = \{[\boldsymbol{\omega}, \boldsymbol{\upsilon}] | \upsilon \in \mathbb{R}^3, \hat{\omega} \in so(3)\} \in se(3)$: $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}$, $\boldsymbol{\upsilon} = \begin{bmatrix} \upsilon_x & \upsilon_y & \upsilon_z \end{bmatrix}$, with $so(3) = \{\hat{\omega} \in \mathbb{R}^{3\times3} | \hat{\omega} = -\hat{\omega}^\top$, where ω and υ are the angular and linear velocities respectively. The reconstruction of a group action $\widehat{\mathbf{T}} \in \mathbb{SE}(3)$ from the twist consists of applying the exponential map using Rodriguez formula [Ma 2004].

Equation (5.4) is solved iteratively in a Gauss Newton fashion, where at each iteration, a new error E and a new Jacobian matrix $J(0)$ are computed in order to obtain the update x by:

$$x = -\Big(J(0)^T J(0)\Big)^{-1} J(0)^T e(x) \tag{5.5}$$

and the rigid transformation is updated as follows:

$$\hat{T} \longleftarrow \hat{T}T(x) \tag{5.6}$$

Minimization is stopped when the error: $\| e \|^2 < \alpha$ occurs, or when the calculated increment becomes too small: $\|x\|^2 < \varepsilon$, where α and ε are predefined stop criteria.

Algorithm 4: CICP Algorithm.

Input: targetCloud, sourceCloud; voxelSeize, $\hat{T}$
Output: Optimal T
Intialize: NormalXYZ T_normals, S_normals; PointXYZ T_match, S_match
begin
  T_normals = normalComputing (targetCloud)
  S_normals = normalComputing (sourceCloud)
  T_match = normalClustering (targetCloud, T_normals, voxelSeize)
  while $(iteration < iter\_\max \| |x| > \varepsilon)$ **do**
    sourceCloud = transform (sourceCloud, S_normals, $\hat{T}$)
    S_match = normalClustering (sourceCloud, S_normals, voxelSeize)
    EstablishCorrespondences (T_match, S_match)
    distanceRejection (distThreshold)
    compute the Jacobian **J** (3.44)
    compute the error vector $e(x)$ (3.43)
    compute the increment x (5.5)
    update the pose **T** (5.6)
    iteration ← iteration + 1
  end
  return T
end

5.5.7 Analysis of the cost function

In this section, a more in-depth analysis of the cost function is carried out. The cost function is based on a sum of squared error (SSE) term as shown by equation (3.42). This is an undeniable problem considering the fact that for a non-linear optimization problem (as is our case), whose domain is non-convex, it may contain several local minima. The local convexity of the SSE estimator around the solution is impacted by several factors; sensor observability, sensor noise, uncertainties induced whilst taking measurements. Therefore, a mathematical condition for convergence is generally difficult to establish.

However, the optimization domain can be sampled to provide a qualitative analysis of the convexity of the estimator. This is illustrated in Figure 5.4 where the root-mean-square error ($RMSE$) (in meters) is shown versus two groups; translational and rotational couples. It is observed that for a typically chosen subsampled point cloud set (dense: $926\,725$ points, sparse: $71\,584$ points), the estimator is convex for the translation as inferred from its formulation and hence, the result is not directly concerned by the initialization of the algorithm. However, the minimiser, though it exhibits a globally convex profile, contains one or several local minima along the way. This implies that initial values for the relative rotation must be carefully given locally around the solution.

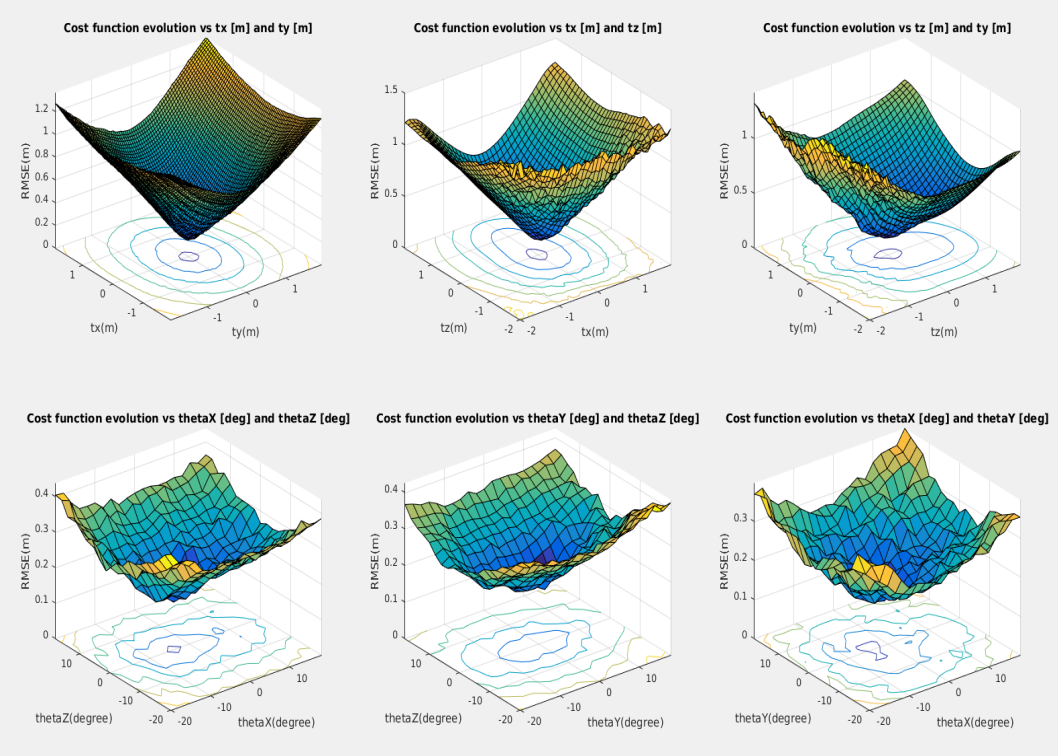


Figure 5.4: *Convergence domain for the office scene. First row represents the $RMSE$ with respect to the three possible DoFs in translation. $t_x, t_y, t_z \in [-2\text{ m}, 2\text{ m}]$ with a descretization of 10 cm. The second row shows the rotation domain where each DoF takes values in $[-20°, 20°]$ with a step of $2°$.*

5.6 Results

We implement our CICP approach in C++ without code optimization, and we conduct multiple experiments to evaluate it. Two different data sets are used: (1) point clouds acquired by different sensors; (2) point clouds generated by a single sensor by varying the scan resolution. These two datasets are carried out on indoor and outdoor environments. The indoor scene is represented by a typical office environment, which is symbolized with walls, desks and chairs. And the outdoor environment (PAVIN) is an experimental site for the development of automated vehicles in realistic urban environment. These different datasets provide a good platform to investigate the performance of the proposed method. We first show its results for the registration of two sparse and dense clouds, acquired with two different sensors, and enumerating different indoor and outdoor environments (Section 5.6.1). Then, we compare our results with the existing sparse and dense methods (Section 5.6.2). Thereafter, we perform registrations between multiple clouds from the same sensor, but with different resolutions (Section 5.6.3). Finally, registrations between clouds from different sensors are carried out (Section 5.6.5).

The computational efficiency of the algorithm is beyond the scope of this work. We would rather focus on the methodology.

The experimental is set up as shown in Figure 5.5. The center of the two sensors; Velodyne HDL32 and that of the Leica P20 are perfectly superimposed with the help of the STANLEY Cubix cross line laser. The velodyne is then physically displaced and rotated by known translations and rotations from the graduated set up in order to perturb the 6 degrees of freedom transformation. Data acquisition is then performed under different scenarios in order to test our CICP algorithm. Table 5.3 below summarizes the various experiments performed in a controlled environment. For each experiment, CICP is initialized at Identity, i.e. $x = [0, 0, 0, 0, 0, 0]$.

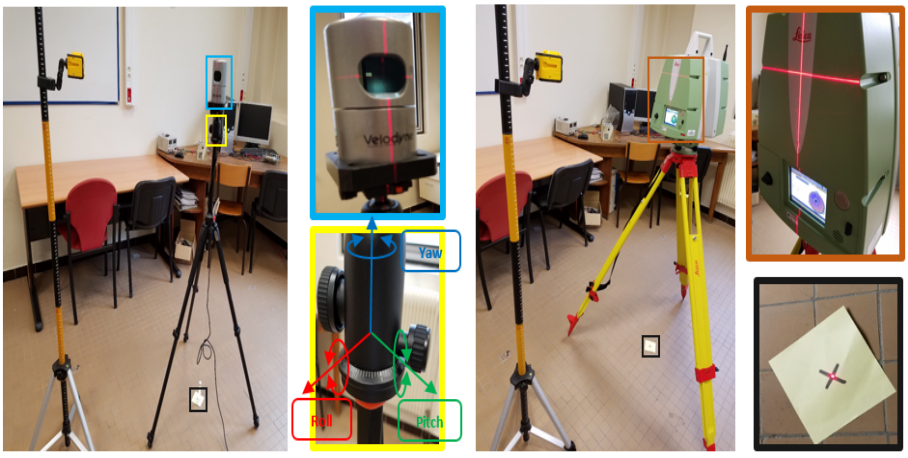
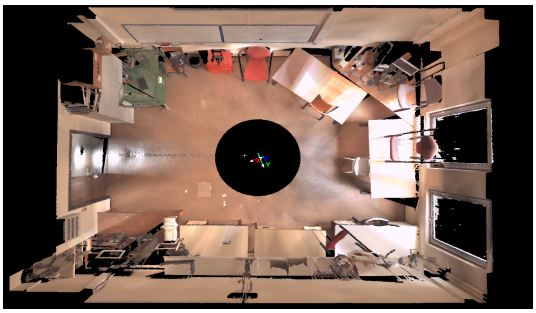


Figure 5.5: *Experimental set up for data collection from Velodyne HDL32 (left) and Leica P20(right) sensors.*

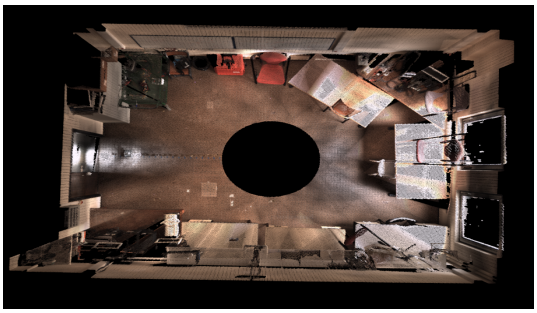
5.6.1 Dense-Sparse Registration with CICP

The purpose of this first experiment is to evaluate our CICP method. For that, we choose two clouds acquired with different sensors; the denser cloud produced by a 3D LiDAR Leica P20 laser scanner and the sparser cloud with

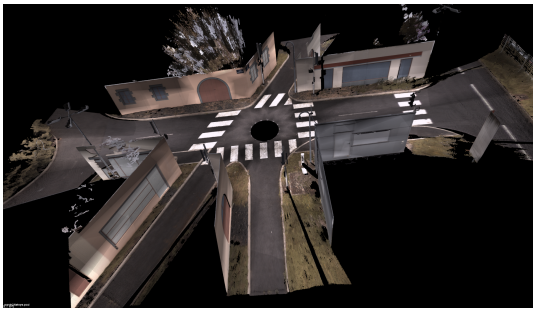
an HDL-32E Velodyne LiDAR sensor. A Leica P20 generates very detailed and dense point clouds as shown in Figures 5.6(a), 5.6(c). Depending on the resolution chosen during the scanning process, these clouds can exceed 100 millions of points for a single scan. For reasons of compatibility with the available computational equipment, which provided an Intel Core i74800MQ processor, 2.7 GHz, and 32 GB of RAM, we perform a sampling process using method described in [Tazir 2016] in order to reduce the number of points to the order of few millions without losing useful information. By compressing data, we provide a more compact 3D representation of point clouds whilst maintaining the notion of density and without affecting the initial structure of the scanned subject. Figures 5.6(b), 5.6(d) illustrate the output of the sampling process with 986 344 and 2 732 783 points for the office and PAVIN scenes, respectively. As for the Velodyne HDL-32E, this sensor generates sparse point clouds that do not exceed 70 000 points. This represents a ratio of 14 times between the two clouds from the first environment and a ratio of 40 times, for clouds of the second environment.



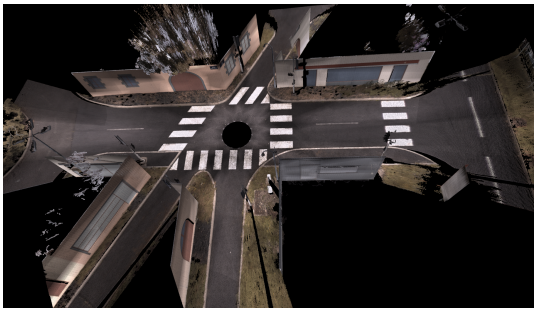
(a) Indoor point cloud delivered by the LiDAR Leica P20 with 88 539 380 points



(b) Indoor point cloud after sampling containing 986 344 points



(c) Outdoor point cloud delivered by the LiDAR Leica P20 with 72 947 044 points

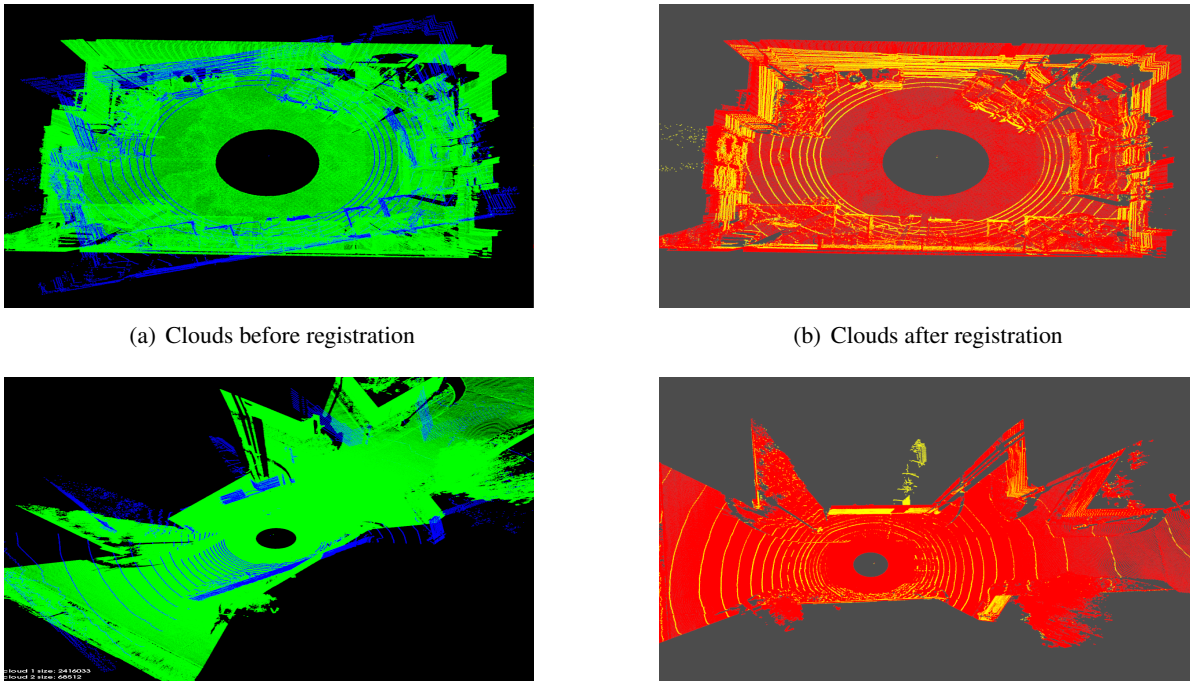


(d) Outdoor point cloud after sampling containing 2 732 783 points

Figure 5.6: *Point cloud sampling.*

Figure 5.7 shows the registration process of such point clouds using the CICP method. On the left, the green cloud is from the LiDAR Leica P20 and the blue cloud is from the Velodyne HDL32-E. The corresponding results are shown on the right. Table 5.2 includes the various parameters that manage the registration. The voxel size is set according to the number of points in the sparse cloud in order to verify the definition proposed in Section 5.4.1. In order to optimize the computing time, it is better to choose the sparse cloud as the source cloud, since the latter is transformed and clustered at each iteration.

In order to verify the convergence of the optimization, a comparison between the two clouds at the start and



(a) Clouds before registration

(b) Clouds after registration

(c) Clouds before registration

(d) Clouds after registration

Figure 5.7: *Registration results with CICP algorithm.*

Table 5.2: *CICP configuration parameters.*

Parameter	Value
Voxel size	8 cm
Rejection distance	$0.2 - 0.5$ m
Max iteration	500
Translation tolerance	10^{-3} m
Rotation tolerance	10^{-4} ◦

at the end of the registration process is recorded together with the convergence profile obtained. It is summarized in the evolution of the $RMSE$ as a function of the number of iterations (see Figure 5.8). As expected, the convergence error draws to a minimum until the imposed stopping condition is reached. It is normally a tolerance on the translation and rotation rates. In our case, this tolerance is 10^{-3} and 10^{-4} for the translation and rotation, respectively. We run the algorithm for several indoor and outdoor scenes, with different viewpoints, as depicted in Table 5.3 and shown in Figure 5.8. Each experiment is performed more than 20 times. For instance, for the experiment 1 (`Expt_1`), the displacement between the two clouds is $[150, 170, 35, 5, 0, 0]$, where the first three values correspond to the translation in millimeters and the last three to the rotation in degrees. As for the fourth experiment, the displacement is $[65, 45, 200, 0, 7, 5]$, which took 34 iterations for the algorithm to converge. A more complete analysis is summarized in Table 5.3, along with the $RMSE$ recorded and the number of iterations achieved at convergence. The analysis of this table allows us to identify three elements that influence the registration results. Namely, the inter-frame displacement and the difference in density between the two clouds, as

Table 5.3: *CICP registration applied to mainly two compiled datasets; OFFICE and PAVIN. The resolutions of corresponding dense and sparse cloud are given along with the initial physical measured transformation from our set up given by the first row of each experiment, whilst the second row depicts the results output by our algorithm. Convergence is evaluated from the $RMSE$ and the number of iterations required for full registration.*

Expt	Envir -onment	# dense cloud	# sparse cloud	t_x (mm)	t_y (mm)	t_z (mm)	θ_x (°)	θ_y (°)	θ_z (°)	$RMSE$ (m)	# iter.
1	Office 1	411924	69952	150.0 155.0	170.0 172.9	35.0 34.8	5.0 4.7	0.0 0.4	0.0 0.1	– 0.0198	– 40
2	PAVIN 1	665260	67488	0.0 0.2	30.0 29.8	200.0 191.9	5.0 4.8	5.0 4.8	0.0 0.1	– 0.0188	– 36
3	Office 2	986344	69984	350.0 357.3	350.0 342.8	0.0 0.3	0.0 0.3	0.0 0.1	0.0 0.8	– 0.0199	– 39
4	PAVIN 2	1364245	67768	65.0 64.1	45.0 45.7	200.0 200.9	0.0 0.3	7.0 6.9	5.0 4.9	– 0.0184	– 34
5	Office 3	2550564	69728	20.0 21.7	110.0 111.1	70.0 66.9	10.0 10.1	5.0 4.2	0.0 0.1	– 0.0191	– 39
6	PAVIN 3	3218879	67936	0.0 0.5	0.0 0.1	0.0 0.3	10.0 9.4	10.0 9.8	0.0 0.2	– 0.0184	– 55
7	Office 4	4490859	69996	30.0 27.8	470.0 470.3	300.0 307.3	20.0 19.6	0.0 0.1	0.0 0.1	– 0.0190	– 57
8	PAVIN 4	5025457	67904	50.0 52.1	50.0 48.2	50.0 53.3	5.0 5.3	5.0 7.5	5.0 5.3	– 0.0152	– 56
9	Office 5	7076192	69760	50.0 55.2	50.0 50.6	50.0 53.9	5.0 5.5	5.0 4.2	5.0 4.6	– 0.0190	– 35
10	PAVIN 5	19615433	67488	0.0 0.1	50.0 49.8	200.0 200.3	10.0 9.7	0.0 0.1	5.0 5.2	– 0.0169	– 48

well as the nature of the environment (indoor or outdoor). For the displacement, we can observe that the larger the initial displacement, the more difficult the registration is. We would like to draw the reader's attention to the fact that the displacements experimented here are quite large, keeping in mind that dense techniques generally require an inter-frame displacement since the cost function is linearized around $x = 0$.

Continuing with our discussion on the influence of initial displacement, let us take the example of experiments `Expt_3` and `Expt_6`, which represent a pure translation and a pure rotation, respectively. These two experiments, as a sample of several experiments that we carry out, show that generally, the pure rotation requires more energy to reach the convergence with respect to the case of pure translation. Tables 5.4 to 5.9 demonstrate further rigorous testing of the cost function with our selection strategy. All six degrees of freedom (DoF) are activated and tested either independently or permuted arbitrarily. The results show the convergence values against the actual values, always with an initialization at identity, i.e. $x = [0, 0, 0, 0, 0, 0]$. It should be noted that, although the actual values are subjected to systematic errors of $\pm 2°$ in rotation and ± 1 cm in translation, they do not affect, one way or the other, the correct functioning in the various steps of our method. The true discrepancy between the two corresponding point clouds at convergence is measured using the $RMSE$. Regarding the influence of density, a

quick look shows that the denser the clouds becomes, the more $RMSE$ decreases, leading to better registration. We will examine this parameter in detail in the Section 5.6.3. Finally, we observe that CICP performance depends on the scene. In fact, the impact of scene affects the performance of registration as observed by the difference of the number of iterations required to reach the convergence between PAVIN's and that of the office. It is clear that the indoor environment achieves better registration than the outdoor scene. This is possibly caused by the richness in planar regions of the former. It should not be overlooked that the outdoor environment contains a large amount of noise and outliers. This can be seen on `Expt_8` and `Expt_9`, in which the initial displacement is the same in both experiments. However, the alignment for the office dataset requires 34 iterations to converge instead of 56 for the PAVIN dataset.

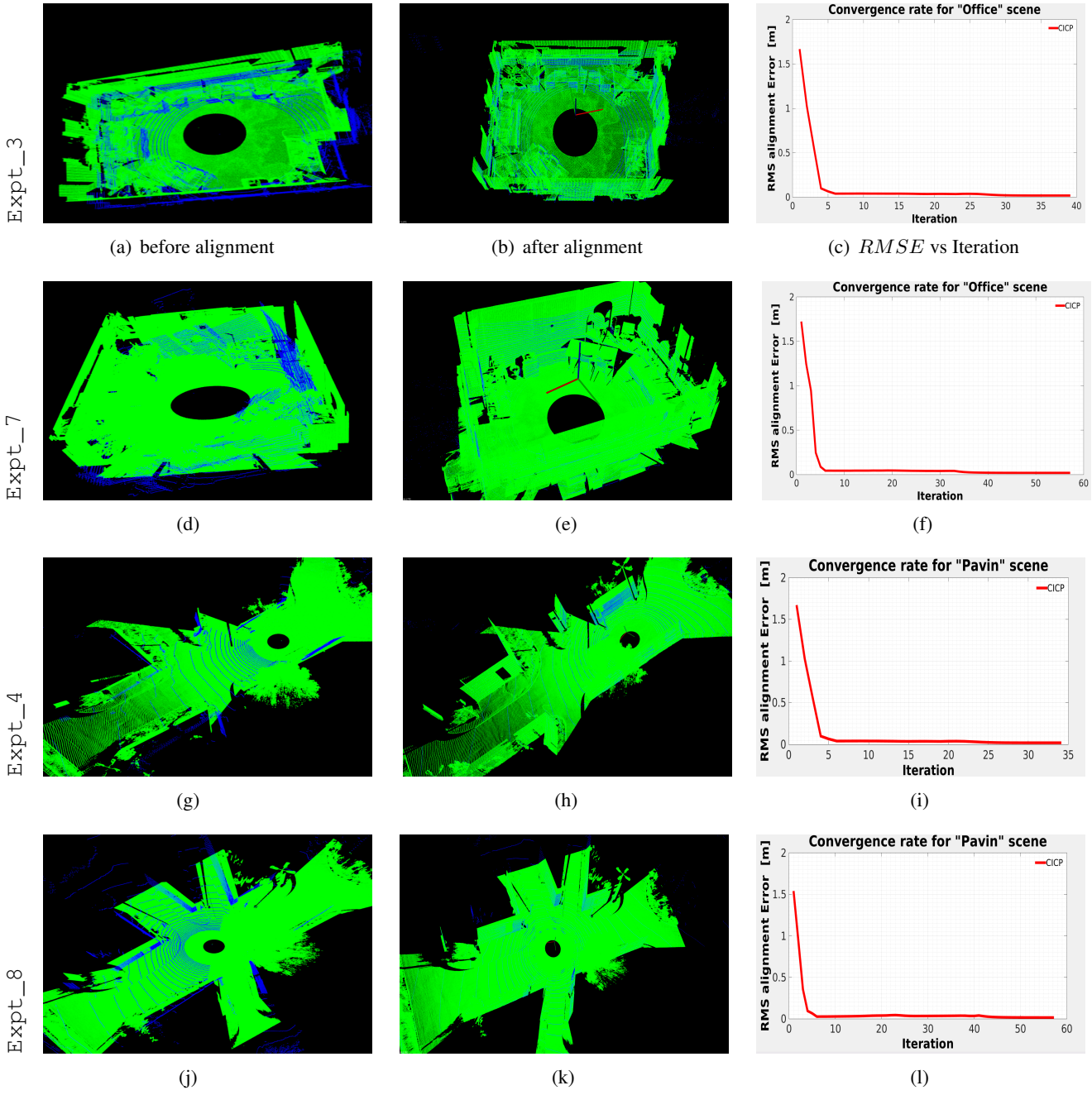
For the sake of illustration, we take four experiments arbitrarily (`Expt_3`, `Expt_4`, `Expt_7`, `Expt_8`), and show their state before and after the registration with their convergence profile in Figure 5.8. A closer look to the $RMSE$ curves in the third column of this figure shows that the residues which are far away are successfully minimized. However, the convergence begins very quickly and then stabilizes for a while before it reaches its minimum. This is mainly due to the fact that there is not a perfect point-to-point equivalence in the two pairing sets. This is quite logical and it can be explained by the large difference in density between the two clouds, the noise, and the clustering defects on the two clouds.

Table 5.4: *Variation only of rotation around one axis: θ_z. The second column represents the actual registration parameters and the third one represents the CICP estimated values.*

θ_z (°)	Actual	Final state	Iterations	$RMSE$
5	[0,0,0,0,0,5]	[0.00, 0.01, 0.00, 0.03, 0.05, 4.73]	64	0.0197
10	[0,0,0,0,0,10]	[0.00, 0.01, 0.00, 0.01, 0.15, 10.01]	99	0.0196
15	[0,0,0,0,0,15]	[0.00, 0.00, 0.00, 0.62, 0.13, 14.68]	95	0.0198
20	[0,0,0,0,0,20]	[0.00, 0.01, 0.01, 0.65, 0.10, 19.71]	157	0.0197
25	[0,0,0,0,0,25]	[0.00, 0.00, 0.00, 0.79, 0.25, 24.64]	220	0.0196
30	[0,0,0,0,0,30]	[0.00, 0.00, 0.03, 1.85, 0.12, 29.59]	303	0.022
35	[0,0,0,0,0,35]	[0.05, 0.03, 0.01, 0.95, 0.14, 36.05]	418	0.0221
40	[0,0,0,0,0,40]	[0.06, 0.03, 0.01, 0.85, 0.10, 40.64]	547	0.0215

5.6.2 Comparison with Existing Methods

In order to compare our method with the existing state-of-the-art methods, we use implemented routines of PCL [Rusu 2011] library for the NDT algorithm, GICP, point-to-plane ICP and simple ICP for dense methods. For the case of sparse methods (methods based on features extraction) we also use PCL implementations of SIFT3D and FPFH to extract characteristic points from the two clouds, and use simple ICP to perform matching. The performance of each method is evaluated using three metrics: the accuracy, the relative translational error and the relative rotational error. The former describes the evolution of the root-mean-square point-to-point distance; this



(a) before alignment (b) after alignment (c) $RMSE$ vs Iteration

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 5.8: *CICP results applied to different datasets from various environments.*

can be expressed mathematically as:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n} \parallel E_i \parallel^2} \tag{5.7}$$

Table 5.5: *Variation only of rotation around two axes.*

θ_y(°)	θ_z (°)	Initial	Final state	Iteration	$RMSE$
15	10	$[0,0,0,0,15,10]$	$[0.00, 0.00, -0.01, 0.09, 14.59, 9.62]$	83	0.0183
15	15	$[0,0,0,0,15,15]$	$[0.00, 0.00, -0.01, 0.50, 15.03, 14.71]$	140	0.0188
15	25	$[0,0,0,0,15,25]$	$[0.00, 0.00, 0.00, 0.65, 15.00, 24.92]$	387	0.0184
20	15	$[0,0,0,0,20,15]$	$[0.00, 0.00, 0.01, 0.89, 19.49, 14.91]$	167	0.0184
20	25	$[0,0,0,0,20,25]$	$[0.00, 0.00, -0.01, 0.82, 19.61, 24.68]$	337	0.0179
30	20	$[0,0,0,0,30,20]$	$[0.01, 0.00, 0.02, 0.93, 28.81, 19.30]$	394	0.01807
30	25	$[0,0,0,0,30,25]$	$[0.01, 0.00, 0.03, 0.91, 28.58, 24.68]$	616	0.01807

Table 5.6: *Variation only of the translation along one axis: t_y.*

t_z (cm)	Actual	Final state	Iteration	$RMSE$
10	$[0, 0.10, 0, 0, 0, 0]$	$[0.00, 0.10, 0.00, 0.23, -0.33, -0.08]$	25	0.0204
20	$[0, 0.20, 0, 0, 0, 0]$	$[0.00, 0.20, 0.00, 0.30, 0.04, -0.42]$	27	0.0205
30	$[0, 0.30, 0, 0, 0, 0]$	$[0.00, 0.30, 0.00, 0.02, 0.27, 0.78]$	35	0.0202
40	$[0, 0.40, 0, 0, 0, 0]$	$[0.00, 0.41, 0.00, 0.18, 0.26, 0.77]$	41	0.0205
50	$[0, 0.50, 0, 0, 0, 0]$	$[0.00, 0.51, 0.01, 0.37, 0.42, 0.90]$	65	0.0205
60	$[0, 0.60, 0, 0, 0, 0]$	$[0.00, 0.61, 0.01, 0.51, 0.41, -0.59]$	94	0.0228
70	$[0, 0.70, 0, 0, 0, 0]$	$[0.00, 0.71, 0.01, 0.37, 0.39, -0.01]$	117	0.0217
80	$[0, 0.80, 0, 0, 0, 0]$	$[0.00, 0.80, 0.01, 0.19, 0.48, -0.32]$	145	0.0211
90	$[0, 0.90, 0, 0, 0, 0]$	$[0.00, 0.91, 0.01, 0.09, 0.46, 0.75]$	160	0.0213
100	$[0, 1.00, 0, 0, 0, 0]$	$[0.00, 1.01, 0.00, 0.09, 0.30, 0.05]$	215	0.0212
120	$[0, 1.20, 0, 0, 0, 0]$	$[0.00, 1.20, 0.00, 0.04, -0.04, 0.89]$	278	0.0198
150	$[0, 1.50, 0, 0, 0, 0]$	$[0.00, 1.51, 0.00, 0.20, 0.33, 0.76]$	386	0.0209
170	$[0, 1.70, 0, 0, 0, 3]$	$[0.00, 1.70, 0.01, 0.11, 0.22, 0.16]$	477	0.0196
200	$[0, 2.00, 0, 0, 0, 4]$	$[-0.01, 2.00, 0.01, 0.26, 0.09, 0.83]$	618	0.0201

where n is the number of points and E_i is the distance error between the source points and its correspondent in the target cloud in each iteration. This can be expressed as follows:

$$E_i = \Sigma_{i=0}^{m} p_i - q_i \tag{5.8}$$

where m is the total number of points in the sparse cloud. p_i and q_i which represent two points of the source and target cloud, respectively, whereby p_i is transformed in the reference frame of q_i.

The second metric is the Relative Translational Error (RTE), which measures the translation gap between the

Table 5.7: *Variation only of the translation along two axes: t_x and t_y.*

$t_{x(\text{cm})}$	$t_{y\ (\text{cm})}$	Actual	Final state	Iteration	$RMSE$
30	30	$[0.30, 0.30, 0, 0, 0, 0]$	$[0.30, 0.31, 0.00, 0.17, 0.46, 0.36]$	33	0.0186
30	50	$[0.30, 0.50, 0, 0, 0, 0]$	$[0.31, 0.51, 0.00, 0.53, 0.48, 0.41]$	67	0.0198
30	70	$[0.30, 0.70, 0, 0, 0, 0]$	$[0.31, 0.71, 0.01, 0.54, 0.49, 0.41]$	92	0.0198
30	100	$[0.30, 1.00, 0, 0, 0, 0]$	$[0.32, 1.01, 0.02, 0.55, 0.49, 0.41]$	166	0.0198
50	50	$[0.50, 0.50, 0, 0, 0, 0]$	$[0.50, 0.51, 0.01, 0.35, 0.53, 0.55]$	87	0.0204
50	70	$[0.50, 0.70, 0, 0, 0, 0]$	$[0.50, 0.71, 0.00, 0.36, 0.53, 0.55]$	108	0.0204
50	100	$[0.50, 1.00, 0, 0, 0, 0]$	$[0.51, 1.01, 0.01, 0.37, 0.54, 0.55]$	185	0.0205
50	150	$[0.50, 1.50, 0, 0, 0, 0]$	$[0.51, 1.51, 0.02, 0.39, 0.54, 0.55]$	298	0.0204
100	100	$[1.00, 1.00, 0, 0, 0, 0]$	$[0.99, 0.99, 0.01, 0.66, 0.46, 0.26]$	262	0.0199
100	120	$[1.00, 1.20, 0, 0, 0, 0]$	$[1.00, 1.19, 0.01, 0.72, 0.50, 0.26]$	309	0.0200
100	150	$[1.00, 1.50, 0, 0, 0, 0]$	$[1.00, 1.49, 0.00, 0.72, 0.50, 0.26]$	450	0.0199
100	200	$[1.00, 2.00, 0, 0, 0, 0]$	$[1.00, 1.99, 0.00, 0.72, 0.50, 0.26]$	590	0.0200

Table 5.8: *Variation only of the translation along three axes: t_x, t_y and t_z.*

$t_{x(\text{cm})}$	$t_{y(\text{cm})}$	$t_{z(\text{cm})}$	Actual	Final state	Iteration	$RMSE$
30	30	30	$[0.30, 0.30, 0.30, 0, 0, 0]$	$[0.31, 0.31, 0.29, 0.50, 0.47, 0.42]$	74	0.0197
50	50	50	$[0.50, 0.50, 0.50, 0, 0, 0]$	$[0.50, 0.52, 0.50, 0.37, 0.47, 0.55]$	124	0.0205
100	100	100	$[1.00, 1.00, 1.00, 0, 0, 0]$	$[1.00, 1.01, 1.00, 0.64, 0.44, 0.27]$	408	0.0200

Table 5.9: *Variation of translation along one axis (t_y) and rotation around one axis (θ_z).*

$t_{y(\text{cm})}$	$\theta_{z(^\circ)}$	Actual	Final state	Iteration	$RMSE$
30	20	$[0.00, 0.30, 0, 0, 0, 0.20]$	$[0.00, 0.30, 0.00, 0.77, 0.07, 19.50]$	85	0.0199
30	30	$[0.00, 0.30, 0, 0, 0, 0.30]$	$[0.00, 0.29, 0.00, 0.46, 0.16, 29.30]$	246	0.0200
50	20	$[0.00, 0.50, 0, 0, 0, 0.20]$	$[0.00, 0.50, 0.01, 0.05, 0.08, 20.12]$	98	0.0206
50	30	$[0.00, 0.50, 0, 0, 0, 0.30]$	$[0.00, 0.50, 0.00, 0.48, 0.21, 29.92]$	350	0.0203
100	20	$[0.00, 1.00, 0, 0, 0, 0.20]$	$[0.00, 1.01, 0.01, 0.74, 0.10, 20.85]$	186	0.0206
100	25	$[0.00, 1.00, 0, 0, 0, 0.25]$	$[0.00, 1.01, 0.01, 0.67, 0.16, 25.89]$	227	0.0197
150	20	$[0.00, 1.50, 0, 0, 0, 0.20]$	$[0.00, 1.50, 0.01, 0.87, 0.17, 21.09]$	473	0.0204

ground truth ($\mathbf{t}_{GT}$) and the estimated ($\mathbf{t}_E$) translation vectors.

$$RTE = \|\mathbf{t}_{GT} - \mathbf{t}_E\|_2 \tag{5.9}$$

For the Relative Rotational Error (RRE), we use the metric defined on the tangent space of $\mathbb{SO}(3)$:

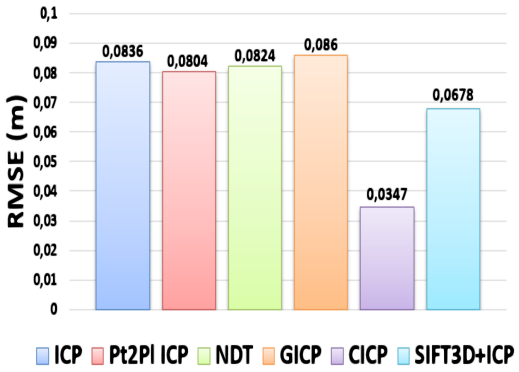
$$RRE = \| \log\mathrm{m}(\mathbf{R}_E^T \mathbf{R}_{GT}) \|_F \tag{5.10}$$

where $\log\mathrm{m}(.)$ is the matrix logarithm, $\mathbf{R}_E$ is the estimated rotation matrix, $\mathbf{R}_{GT}$ is the ground truth rotation matrix and $\|.\|_F$ is the Frobenious norm.

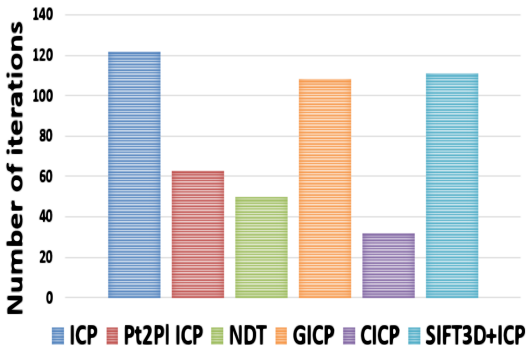
Table 5.10: *Comparison with the state-of-the-art methods.*

		Dense					Sparse	
		ICP	pt2pl ICP	NDT	GICP	CICP	SIFT 3D+ICP	FPFH +ICP
Office	$RMSE$ (m)	0.0602	0.0620	0.0636	0.0636	**0.0299**	0.0516	failed
$[\mathrm{m}, \mathrm{m}, \mathrm{m}, ^\circ, ^\circ, ^\circ]$	RTE (m)	0.2811	0.2482	0.2019	0.2178	**0.0169**	0.0191	failed
$[0, 0.5, 0.5, 20, 0, 10]$	RRE ($^\circ$)	0.0517	0.0.0186	0.0285	0.0213	**0.0005**	0.0201	failed
PAVIN	$RMSE$ (m)	0.0836	0.0804	0.0824	0.0860	**0.0347**	0.0678	failed
$[\mathrm{m}, \mathrm{m}, \mathrm{m}, ^\circ, ^\circ, ^\circ]$	RTE (m)	0.0365	0.0253	0.0315	0.0642	**0.0092**	0.021	failed
$[0, 0.5, 0.3, 0, 0, 10]$	RRE ($^\circ$)	0.0058	0.0050	0.0058	0.0090	**0.0034**	0.0058	failed

Table 5.10 presents the results gathered in processing two indoor and outdoor scenes with the state-of-the-art methods. Bold values show the best result. Quantitatively, the $RMSE$ value of the indoor scene reaches 6 cm in the case of point-to-point, 6.2 cm point-to-plane ICP, 6.3 cm for the NDT and the GICP, more than 5 cm for SIFT3D and less than 3 cm for the proposed method. The maximum number of iterations for each test is fixed at 500 beyond which the algorithm is considered as not having converged if it reaches that ceiling, as is the case of the FPFH method.



(a) Comparison of $RMSE$ results of the different registration methods



(b) Comparison of number of iterations achieved at convergence of different registration methods

Figure 5.9: *Convergence comparison between different registration methods.*

Figure 5.9 shows the comparison of convergences between different registration methods. CICP outperforms the state-of-the-art methods on both datasets. In addition, it is shown that CICP is robust against scene variation.

5.6.2.1 Experiment on semi structured environment

The objective of this experimental set is to compare the performance of CICP with state-of-the-art algorithms (ICP, P2Pl, NDT and GICP) using a newly acquired dataset for a semi structured scenario type of environment where planar surfaces are far less pronounced. The results are given in Table 5.11, which presents the ground truth displacement versus the final results found by each algorithm. The tuning parameters using for each one of them is laid out in Table 5.12. Figure 5.10 gives the $RMSE$ error against the number of iterations to convergence. Again, the performance of CICP is very apparent and surpasses state-of-the-art, as shown in Figure 5.11.

Table 5.11: *Performance comparison of each algorithm related to experimental section (§ 5.6.2.1).*

	t_x (m)	t_y (m)	t_z (m)	θ_x (°)	θ_y (°)	θ_z (°)	$RMSE$ (m)	# iter
Actual	1.00	0.50	0.00	0.00	0.00	20.00	–	–
ICP	1.66	−0.28	0.00	−0.01	0.00	17.50	0.3406	500
P2Pl ICP	0.99	−0.52	0.00	−0.01	0.00	20.45	0.2482	361
NDT	1.71	−0.18	0.00	−0.01	−0.01	19.22	0.2876	131
GICP	1.14	0.16	−0.01	−0.06	0.00	15.02	0.5785	188
CICP	**0.99**	**0.50**	**0.00**	**0.02**	**0.05**	**20.24**	**0.0444**	**163**

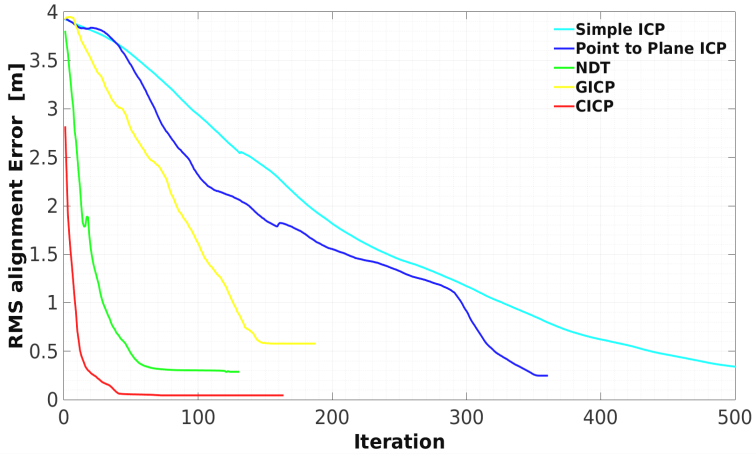
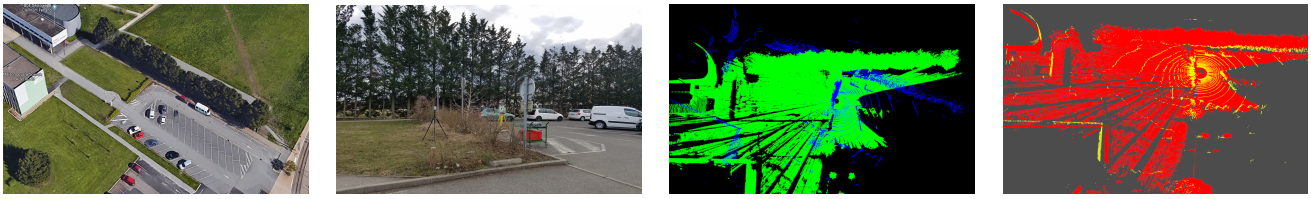


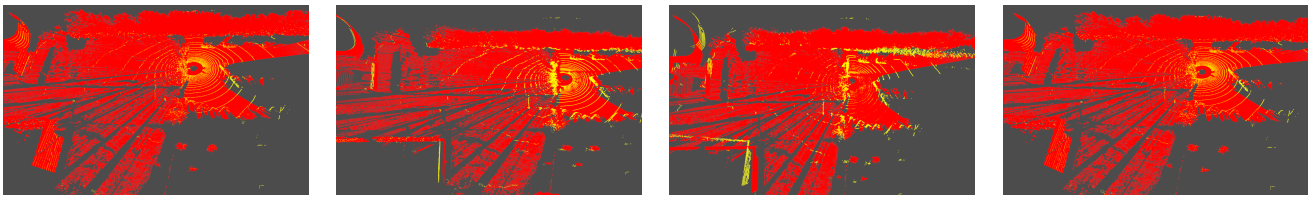
Figure 5.10: *Cost function evolution for the unstructured environment.*

5.6.3 Changes in Density

To investigate the role of density on CICP performance, we conduct two experiments. The first is on two clouds collected with two different sensors, while for the second, the clouds are acquired with the same sensor but by varying the scanning resolution.



(a) Google earth top view (b) Site front view (c) Initial state of two scans (d) ICP



(e) Pt2pl ICP (f) NDT (g) GICP (h) CICP

Figure 5.11: *Results with a semi structured environment. Registration between a Leica P20 scan and the Velodyne HDL32-E is performed using state-of-the-art algorithms and CICP.*

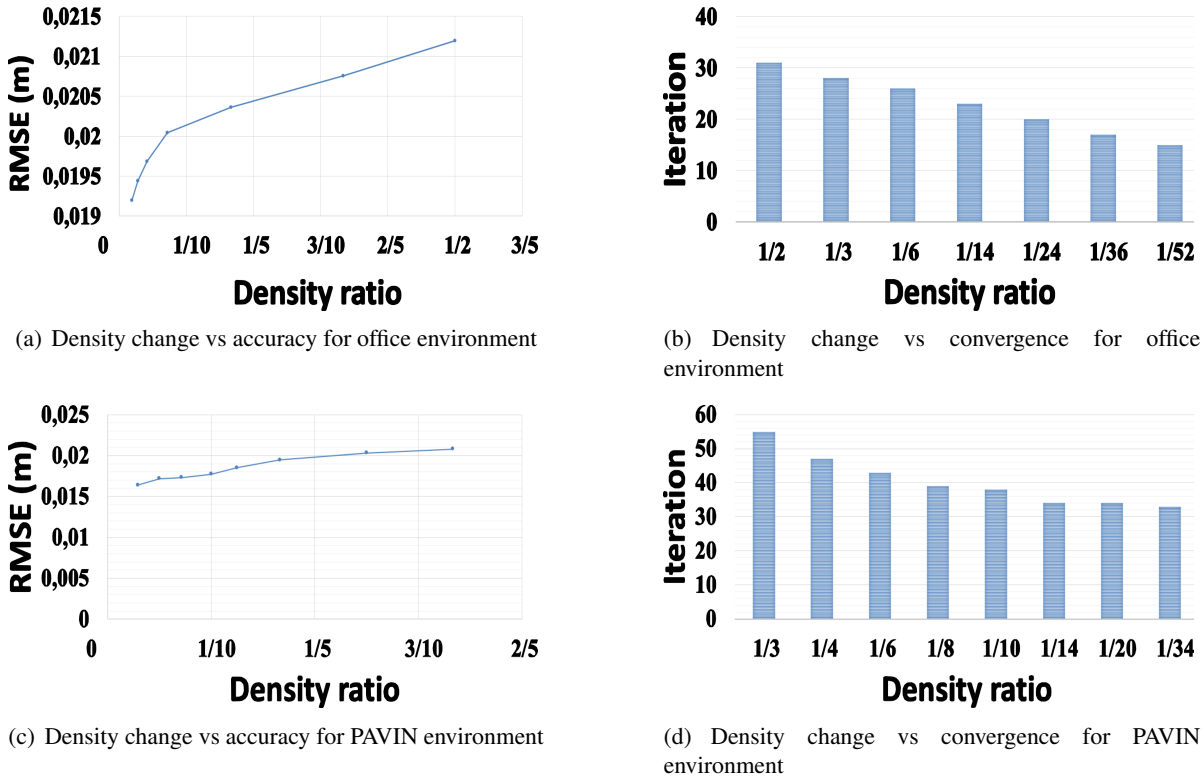
Table 5.12: *Tuning parameters used of each algorithm for experimental section (§ 5.6.2.1).*

Parameter	ICP	ICP P2PL	NDT	GICP	CICP
$\varepsilon(stopcriteria)$(m)	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Rejection distance (m)	0.5	0.5	0.5	0.5	0.5
Maximum iterations	500	500	500	500	500
# neighbors normals estimation	–	10	–	–	10
# neighbors compute covariance	–	–	–	20	–
maximum step size for MT search	–	–	0.1	–	–
Resolution of NDT grid (m)	–	–	1	–	–

5.6.3.1 Data from two different sensors

Our first fold of experiments in this section is performed on clouds from two different sensors. The original dense cloud for the first dataset "office" is acquired with the LiDAR Leica P20. It is of size of 19 615 433 points. We perform different sampling using the method described in [Tazir 2016], which result in seven clouds having a size of between 4 million points and 100 000 points. These clouds are registered with a sparse cloud obtained from the Velodyne HDL-32E sensor of size 69 952 points. The results of this experiment are presented in Figure 5.12.

This experiment shows that despite the substantial difference in density between the two clouds in each test, there is only a slight change in the $RMSE$ (in the order of few millimeters) and in the iterations required to reach the convergence. This is mainly due to the fact that the density does not directly affect the error which is calculated from the pairing set of points. Density acts mainly on the correctness of the clustering. Indeed, the high density gives rise to properly grouped regions, which lead to points exhibiting the surface characteristics that it represents as much as possible, implying good matching and thus high accuracy. This represents the strength of our method.



(a) Density change vs accuracy for office environment

(b) Density change vs convergence for office environment

(c) Density change vs accuracy for PAVIN environment

(d) Density change vs convergence for PAVIN environment

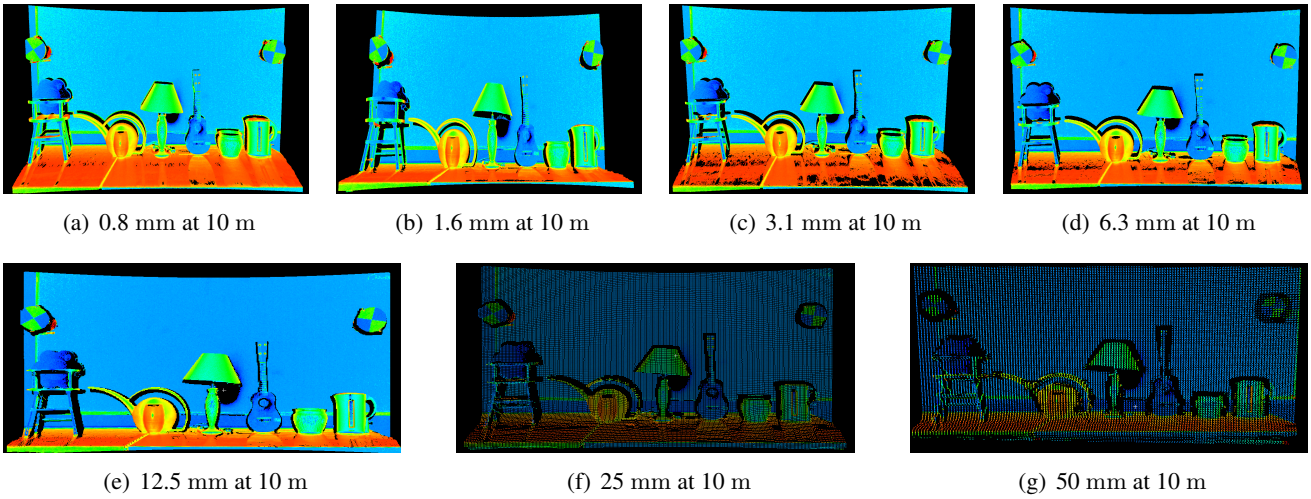
Figure 5.12: *Density change effects on the convergence and accuracy of the CICP method.*

Clustering always ensures the availability of representative points from which the matching is performed, even in the case of low density. The only difference is in the minimal change present in $RMSE$, as illustrated by the graphs in Figures 5.12(a) and 5.12(c). This can be expressed as: high density implies a superior accuracy.

5.6.3.2 Data from the same sensor

The second batch of experimentation in this section consists in aligning different clouds acquired by the same sensor. Figure 5.13 shows seven clouds with different densities. All these clouds are taken with the same sensor by changing the scanning resolution each time. Indeed, the LiDAR Leica P20 allows the change of resolution, by increasing or decreasing the sampling distance (distance that separates two points of the cloud at a given distance from the scanner). In the case of the Leica P20, this distance varies from 0.8 mm per 10 m to 50 mm/ 10 m, giving rise to different density variation as illustrated in Figure 5.13. In this experiment, the dimensions of clouds are fixed within the sensor. This latter is placed in one fixed position, and the only difference between these 7 clouds is the scanning resolution (neither the dimensions, nor the viewpoint). The outcome of this experiment is shown in Table 5.13.

The conclusion that can be drawn from the findings of this evaluation is that the difference of density between the two clouds only affects the convergence of the algorithm. As in this experiment, nothing changes between the two clouds except the density. The final $RMSE$ values are all equal and are close to the accuracy of the sensor



(a) 0.8 mm at 10 m (b) 1.6 mm at 10 m (c) 3.1 mm at 10 m (d) 6.3 mm at 10 m

(e) 12.5 mm at 10 m (f) 25 mm at 10 m (g) 50 mm at 10 m

Figure 5.13: *Density change within the same sensor (the colors from blue to red correspond to the scale of the intensity of the reflectance).*

Table 5.13: *Density change results.*

Resolution cloud 1 (mm at 10 m)	1.6	3.1	6.3	12.5
Resolution could 2 (mm at 10 m)	25	25	25	25
Resolution ratio	1/16	1/8	1/4	1/2
$RMSE$ (m)	0.0027	0.0027	0.0027	0.0027
Iteration	3	6	7	10

(3 mm). The only difference is in the number of iterations needed to reach the convergence. This can be formulated by: a small difference in density between the two clouds extends the convergence process. In fact, this is due to several reasons. First, choosing the sparse cloud as the source cloud (in order to optimize the computing time as mentioned above). This means that the search for nearest neighbors is done in the dense cloud for the points of sparse cloud. Secondly, according to the mathematical definition proposed in this chapter (Section 5.4.1), the voxel size for clustering is fixed according to the number of points in the sparse cloud, in order to ensure points for matching. These two reasons impact the search of the nearest neighbor, which is easier and more accurate in a dense cloud and less precise and longer in a less dense cloud.

5.6.4 Changes in density and viewpoint

Table 5.14 shows the results of the alignment of different clouds acquired by the same sensor and with viewpoint change, which was $[50, 50, 50, 5, 0, 0]$. Our main motivation to conduct this experiment, which differs from the previous one by the addition of the view change, is to confirm the influence of the density change as the scanned pattern is the same.

Table 5.14: *Density & viewpoint change results.*

Resolution cloud 1 (mm at 10 m)	0.8	1.6	3.1	6.3	12.5	25
# points of cloud1	23870726	9800513	2527043	1219973	673537	65626
Resolution cloud 2 (mm at 10 m)	50	50	50	50	50	50
# points of cloud 2	20676	20676	20676	20676	20676	20676
Ratio of points	1/1150	1/470	1/120	1/60	1/32	1/3
$RMSE$ (m)	0.0060	0.0065	0.0073	0.0082	0.0107	0.0166
# iteration	40	44	46	50	58	64

Findings from this second experiment provide further evidence about the conclusions drawn earlier and the role of density. Indeed, as we explained previously, high density gives rise to superior accuracy and improves the convergence speed (this can be seen on the sixth and seventh rows of the Table 5.14).

5.6.5 Comparison with Various Sensors

To investigate the potential of our approach to align point clouds from different sensors, we test it with three kinds of sensors, the Leica P20 LiDAR, Velodyne HDL32-E LiDAR and SR4000 Time-of-Flight camera. Figure 5.14 shows these different sensors and Table 5.15 exhibits their hardware specifications.

5.6.5.1 Leica P20

Many varieties of 3D LiDAR sensors are available on the market, but they all work with the same basic principle [Puttonen 2013]. They emit pulses and detect their reflection in order to explore the object or the environment. Leica P20 is a Time-of-Flight scanner which offers greater range and precision.

5.6.5.2 Velodyne HDL32-E

The Velodyne HDL-32 produces 3D scans by rotating a 32-beam array around its vertical axis at 10 Hz. It produces approximately 700000 points per second or 2200 points per laser beam at a range of 1 through 70 meters. This sensor provides an angular resolution of approximately 0.16° with a field of view (FOV) of 360°. Its vertical field of view is from -30.67° to $+10.67^\circ$ with an angular resolution of 1.33°. Its measurement accuracy is generally less than 2 cm maximum.

5.6.5.3 SR4000 Time of Flight camera

The Time of Flight (ToF) camera is a two-dimensional scanner which captures full depth per frame and with a single light pulse.

Figure 5.14: *Sensors used to test the CICP approach. From left to right: SR4000 Time-of-Flight camera, Velodyne HDL 32-E, and Leica P20 Laser.*

Table 5.15: *Sensors hardware specifications.*

Sensor	Range (m)	Field of view (°) Horizontal	Vertical	Scanning Frequency (Hz)	Accuracy (mm)
Leica P20	$[0.5 \dots 120]$	360	270	50	3
Velodyne HDL-32E	$[1 \dots 70]$	360	$[-30 \pm 10]$	10	20
SR4000	$[0.1 \dots 10]$	43.6	34.6	50	15

The rest of this section discusses the registration of different clouds acquired by these sensors.

5.6.5.4 Leica P20 vs Velodyne

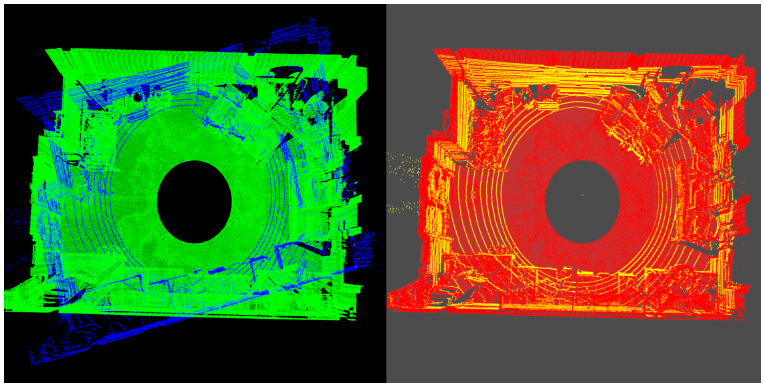


Figure 5.15: *Registration of two clouds captured by Leica P20 LiDAR for the dense cloud and Velodyne HDL-32E for the sparser cloud.*

In this experiment, the point cloud produced by the Velodyne sensor is the sparse cloud and the second cloud, which represents the dense one, is produced by the LiDAR Leica P20 (Figure 5.15). The algorithm converges after 64 iterations, with 0.0199 mm as the $RMSE$ value.

5.6.5.5 Leica P20 vs SR4000

Here, the sparse cloud is produced by the SR4000 Time-of-Flight camera, while the dense cloud is produced by the Leica P20 sensor (Figure 5.16). The latter is less affected by noise than the former.

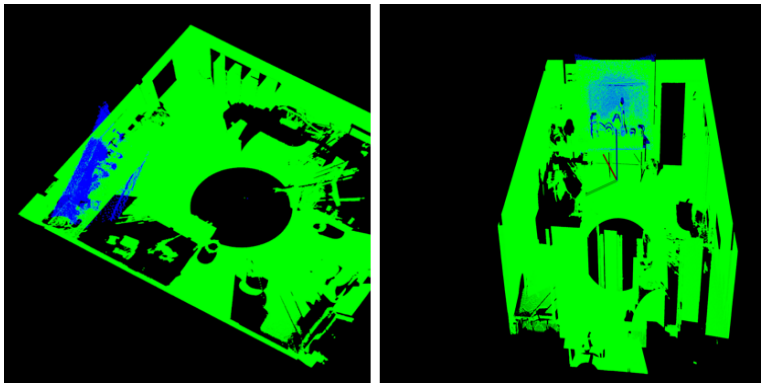


Figure 5.16: *Registration of two clouds captured by Leica P20 LiDAR for the dense cloud and SR4000 ToF camera for the sparser cloud.*

The results of this convergence are 100 and 0.0203 for the number of iterations and the $RMSE$, respectively. The reader can observe that the $RMSE$ of P20 vs SR4000 experiment is higher than the $RMSE$ of the test performed between P20 and Velodyne sensors. This is justified by the large presence of noise in the cloud delivered by the ToF camera.

5.6.5.6 SR4000 vs Velodyne

Here the two sensors are affected by noise. This explains why the registration takes more than 124 iterations to converge. The final $RMSE$ is about 0.0231. In contrast to the previous experiment, the dense cloud is produced by the ToF camera, while the Velodyne cloud represents the sparse one. Even though the total number of points in the Velodyne cloud is almost 3 times higher than the cloud ToF, in the part that interests us (representing approximately $(2 \times 1 \times 1)$ m^3, which enclose the table and objects exposed on it and the table behind), the ToF cloud is denser than the cloud of the Velodyne (Figure 5.17).

5.6.6 Demonstration with Dense-to-Dense Data

At the end of this experimental section, we wish to highlight that the CICP method can be used with clouds of the same nature (dense to dense or sparse to sparse).

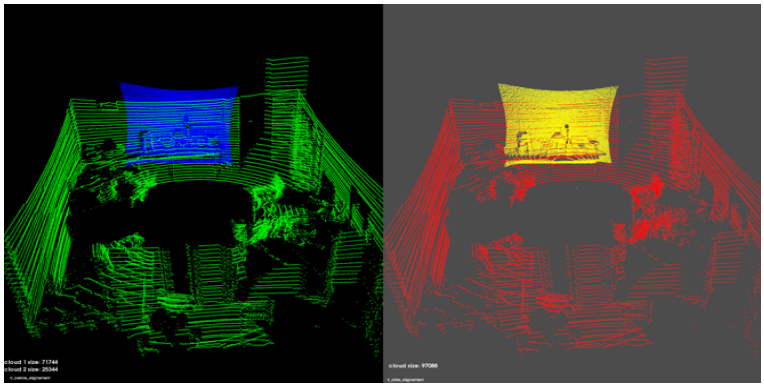


Figure 5.17: *Registration of two clouds captured by a ToF camera for the dense cloud and a Velodyne for the sparse cloud.*

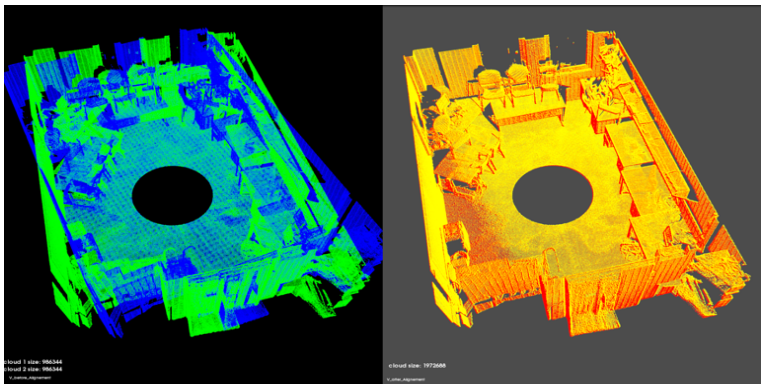


Figure 5.18: *Registration of two dense clouds of an indoor scene captured by a Leica P20 sensor.*

Figure 5.18 shows the state of the two dense clouds before and after the registration. Despite the large number of points, the final result is correctly aligned. Here is another benefit of our approach, the fact of not considering the entire set of points for matching, but only a collected set of points from each local surface, which improves the convergence speed.

5.6.7 Impact of the voxel size

The voxel size plays a very important role in the convergence of the algorithm. Figure 5.19 illustrates how the convergence is impacted by the change of this parameter. When increasing the voxel size, the number of points included in this voxel is increased, which reduces the number of points used for matching. As clustering generates a few representative points relative to the input points, thereby increasing the convergence speed, but decreasing the accuracy. Setting a small voxel size decreases the convergence speed, but increases accuracy, due to the availability of sufficient points for matching. Therefore, a reliable trade off needs to be determined in order to find the optimal discretization of the point cloud.

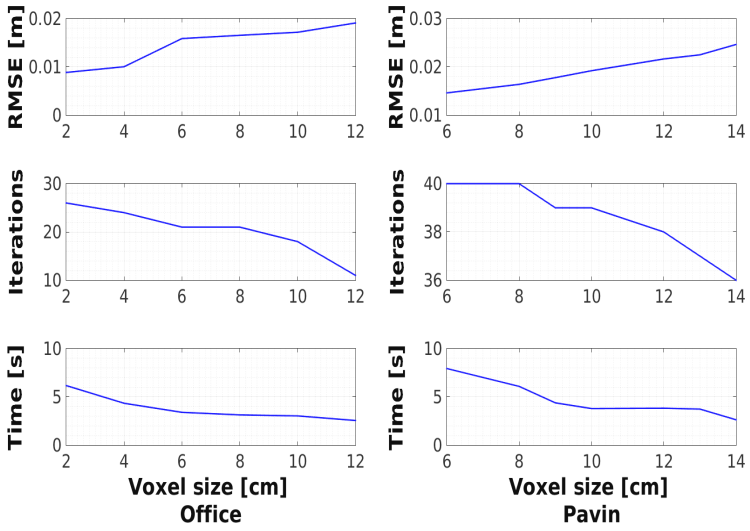


Figure 5.19: *Clustering voxel size effect on registration accuracy and convergence speed.*

5.7 Discussion

1. Two or more point clouds are acquired from the same scene but with different sensors (e.g. vision based system producing sparse cloud and 3D LiDAR producing dense cloud), leading to different clouds with their own local coordinate system, resolution and number of points. Registration methods based on the classical point-to-point ICP metrics fail to provide an accurate pose estimate because of the large discrepancies in density between the two point clouds. The difficulty lies in the fact that there are no direct correspondences between the source and the target point clouds. What is more, methods based on the geometric characteristics are also unsuitable as these (mainly normal and curvature [He 2016]), which are essentially based on their estimations on the neighboring points, are affected by the change in resolution and scanning patterns [Das 2014, Holz 2015].

2. The information carried by a 3D point can contain color, reflectance (intensity), positions, normals and curvatures. Whilst color or reflectance differs from one sensor to another, for the case of a static environment, the global geometric aspect of the scene remains unchanged. This is obviously, why CICP capitalizes on geometric primitives. In addition, this makes it independent of weather and illumination conditions.

3. It complements outperforms the famous the state-of-the-art methods (ICP, NDT, GICP) for this kind of multi sensors applications. These classical algorithms find their limits with dense-sparse registration, because they are all based on point-to-point matching. Whereas our method is based on the concept of surface-to-surface matching, this concept can be generalized to any type of common clustering between the two point clouds. We can imagine the use of segments or dominant direction of variation of the points which corresponds to the highest eigenvalue in the PCA. The matched surfaces are chosen to be very small local surfaces. To do that, a common voxelization between the two clouds with a very small voxel size is performed. The choice of matching small surfaces was made in order to preserve the topological details of the scanned environment and to guarantee a considerable number of matched points between the two clouds.

4. Normals are computed once before starting the process and are used only to distinguish the different local surfaces. They are not used in the alignment process. The use of surface normals in point to plane ICP and its variants is motivated by the fact that the former are robustly estimated in the presence of noisy surfaces. Otherwise, they will cause ICP to diverge. In the case of CICP, we make use of normals to perform surface segmentation. Such an approach improves the surface estimate for noisy measurements. Figure 5.2 shows normal vectors extracted from the same surface scanned by two distinct sensors. It can be clearly observed that though the surfaces are piecewise planar, their estimated normals do not correspond. Therefore, this reinforces the idea of not retaining the normals for the optimization phase.

5. Whether in the case of dense or sparse cloud, there is a certain number of points that are redundant. Redundancy, though useful to make robust the overdetermined system of the normal equation comes at the cost of increased computation. Therefore, the use of representative subsets of points give the same if not better accuracy with less computational time.

6. As shown in this chapter, the use of normals is a double-edged sword and it can guarantee good quality results if accurately exploited. In the same way, normals can amplify noise leading to divergence of the ICP method. The proposed approach makes use of normals for clustering points from the same surface, and we avoid using them to establish correspondences due to various disturbances coming from the sensor. This strategy enables us to overcome the main weakness of dense to sparse/sparse to dense registration. It allows us to surpass the problem of density in search of correspondences. The cascaded effects of an improved surface match correspondence lead to better accuracy in the registration pipeline at reduced computational cost.

5.8 Conclusion

In this chapter, we presented in detail a novel approach for sparse to dense point cloud registration. The traditional ICP pipeline is modified to accommodate a smarter way of surface patch correspondence consisting of three main blocks; voxelization, clustering, representative election. The motivation behind this strategy is to match identical surfaces in the 3D world but scanned using different type of depth sensors. With various sensors come different resolutions and hence different point cloud representations. Throughout our experimental phase, we demonstrate the efficiency of our algorithm in terms of alignment accuracy where other state-of-the-art techniques perform poorly since they do not cater for these above-mentioned differences. Furthermore, we show that the alignment technique works perfectly even by changing the density of the points of the two clouds.

To summarize, our proposed methodology provides the following improvements:

- patch surface segmentation contributing to noise reduction,
- improved selection by a novel surface point representative approach,
- reduced amount of processed data during matching phase,
- dense to sparse registration applicable to the various depth sensors on the market,

- a novel mathematical definition of sparse and dense clouds.

Finally, the scope of this work lies within the generic problem of localization for either hand-held applications for augmented or virtual reality or robotic platforms navigating inside an *a priori* mapped environment. The latter is among potential applications of our approach, since CICP is suitable to localize a vehicle equipped with a sensor that provides sparse data (e.g. Velodyne) in a dense and accurate map. This is what we are going to present in the continuation of this manuscript.

Part III

MAPPING AND LOCALIZATION

It is true that the registration and the scan-matching are two fundamental blocks of our mapping and localization processes respectively, but each one represents only a single brick. The whole of each process emerges only after the successful combination of different pieces of each. The purpose of this part is to present the method for combining registration to produce the reference maps and combining scan-matching to achieve a precise localization.

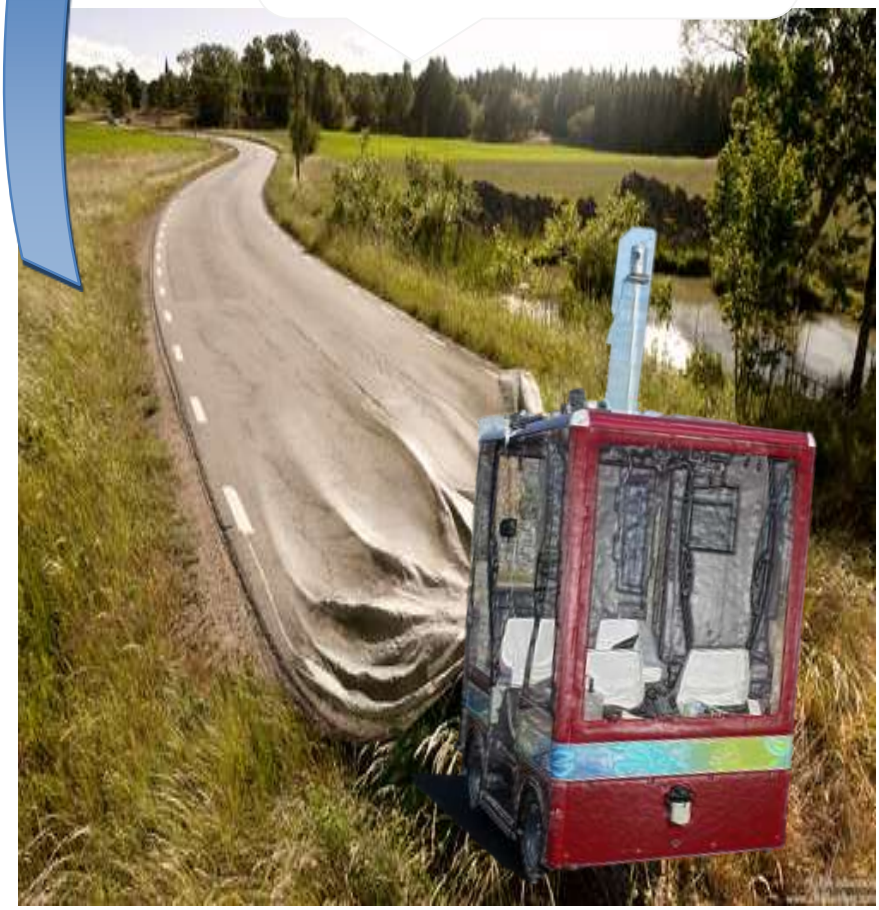
CHAPTER 6 *presents the method used to construct the reference maps using static or dynamic laser surveying technique. Explicit details about the experimental setup and data acquisition campaigns thoroughly carried out during the course of this work are given.*

CHAPTER 7 *introduces a new method allowing points cloud reduction. The goal is to reduce the map volume in order to produce efficient maps that the environment representation contained in the 3D model are compact, robust, and real-time used.*

CHAPTER 8 *discusses the use of the CICP concept to achieve a precise localization using a 3D prior map and the incoming point clouds as the vehicle moves.*

Chapter VI

Creating of the reference map



CHAPTER 6

Creating of the Reference Map

Contents

In this chapter, the construction of reference maps is discussed. These maps can be either built offline with data acquired passively or be on the fly with the motion of the autonomous agent in the space. For the first technique, the data is acquired using a high-resolution Leica P20 platform and processed using its software packages. Whilst, for the second technique, an odometry-free mapping approach based on keyframes representation is adopted. The idea is to construct efficient maps maintaining a coherent model of the explored space. Multiple experiments are performed to evaluate the dynamic mapping technique. Tests are conducted in simulation, local tests in indoor and outdoor environments using our means, and tests on an external dataset (KITTI odometry datasets).

6.1 Introduction

With the development of highly automated driving vehicles, a need for a new type of high-precision map, called HD map, has also appeared. These maps have gotten much higher requirements in terms of details and updates compared to the currently available navigation systems. Because the latter, which are used for vehicle navigation or geographic information systems are not sufficient to meet the new requirements of intelligent vehicle systems such as autonomous driving. There are four main roadmap requirements for intelligent vehicle systems: centimeter accuracy, storage efficiency, real-time feasibility, and map updates. In this chapter, we consider the first constraint; the rest of the constraints will be treated in the following chapters.

Accurate map with centimeter level of precision is a key factor towards enhanced safety for autonomous driving. It allows the vehicle to accurately localize itself within its surroundings in order to avoid obstacles (e.g. road structures, pedestrians and other vehicles). Using a prior map rather than based only on the on-board sensors allows to reduce the navigation problem (perception, mapping, localization and path planning) to only a problem of localization, as with the a priori map, the path planning can be done off-line before the robot starts to move.

However, the acquisition and the construction of dense, precise and usable 3D maps for localization are not yet mastered on a large scale. In this chapter, we will address this topic.

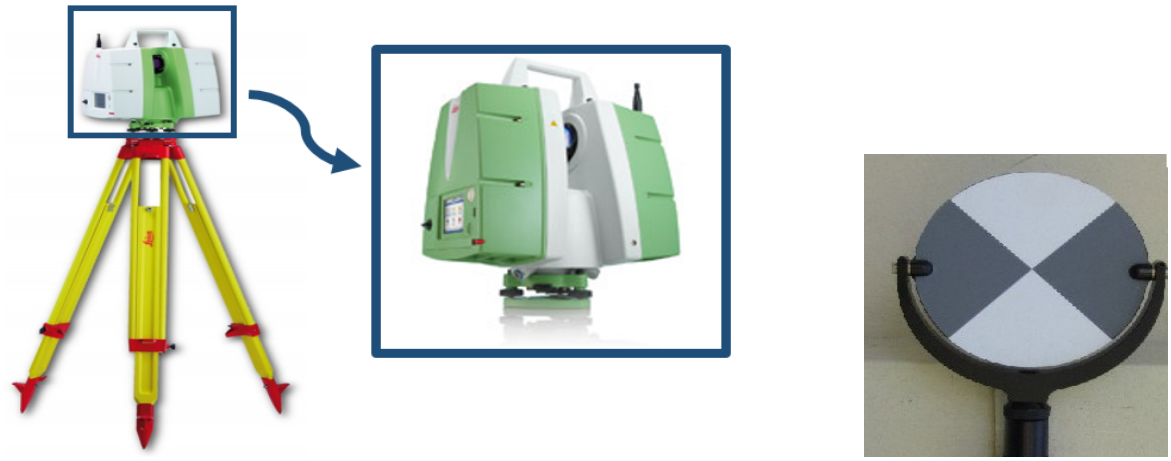
6.2 Map building

As previously introduced, there are two classes of techniques used for mapping the surrounding environment. The static acquisition techniques, used mainly to generate a metrically accurate map for the environment, and the mobile acquisition techniques, which allow mapping a very large environment, but well faster than the static techniques. In our thesis, we used both techniques, and this is what we will explain in the following section of this chapter.

6.2.1 Static acquisition technique

Our goal here is to generate a metrically accurate map of the environment. For that, we use a very powerful terrestrial laser scanner, which is the Leica P20 scanner [1] (Figure 6.1). A recent study published in [Pandžić 2014] had evaluated by means of point cloud fitting algorithm the quality of the data acquired by this sensor and had claimed the satisfying quality of its data. Indeed, this tool allows obtaining a very dense and accurately point clouds. It is based on time-of-flight technology and can deliver one million points per second at a maximum range of 120 m. The Leica P20 has a horizontal field of view of 360° and vertical of 270°, and it is equipped with a camera, allowing to colorize the provided points cloud.

[1] Source: https://w3.leica-geosystems.com/downloads123/hds/hds/scanstation_p20/brochures-datasheet/leica_scanstation_p20_dat_en.pdf



(a) The 3D terrestrial laser scanner Leica P20 (b) The used target

Figure 6.1: *The 3D terrestrial laser scanner Leica P20 and target used for the static map building.*

6.2.1.1 Used methodology: scanning procedure

The TLS can capture data from objects in front of it, so in order to digitize a complete 3D model of the environment, the scanning process is repeated from several locations and diverse angles, each expressed in a local repository. Different point clouds are then aligned together and merged to a common point cloud. However, the scanning process with the TLS is not simply press the starting button, and wait for the results to come out. It is more complicated than that, it is a process that requires advanced knowledge. In addition, this process requires preparatory work to ensure the smooth running of the digitization campaign. The fieldwork consists of:

a. Adjusting the position of the P20 scanner: The location of the scan station should be chosen with maximum precaution to ensure high coverage and good accuracy while minimizing the number of stations to do. The positioning of the scanner or adjusting the levels in some cases can be difficult. The technique is to ensure that the tripod is horizontal to the eye before attaching the scanner. Also, ensure the robustness of the seat (more slippage possible, tightening screw on the extension of the feet blocked). Then place the scanner and lock it. Finally, the fine adjustment is to act on (the) wheel (s) in opposition to the direction of the bubble as Figure 6.2 shown. The direction of the bubble is the peripheral point of the bubble to which the bubble collapses.

b. Targets positioning: Magnetic targets are used to increase the accuracy of the assembly of different point clouds. Targets should be scanned with sufficient density to model their centers and give better results. The size of the target and its distance from the scanner determine exactly how the center can be modeled. Therefore, the correct positioning of the targets is essential to have a very precise assembly. Three common targets between two scan stations is a minimum to assemble two point clouds. Even more, two common targets are sufficient taking into account the verticality constraint of the sensor.

A “smart” or relevant target positioning consists in:

[2]Source: https://w3.leica-geosystems.com

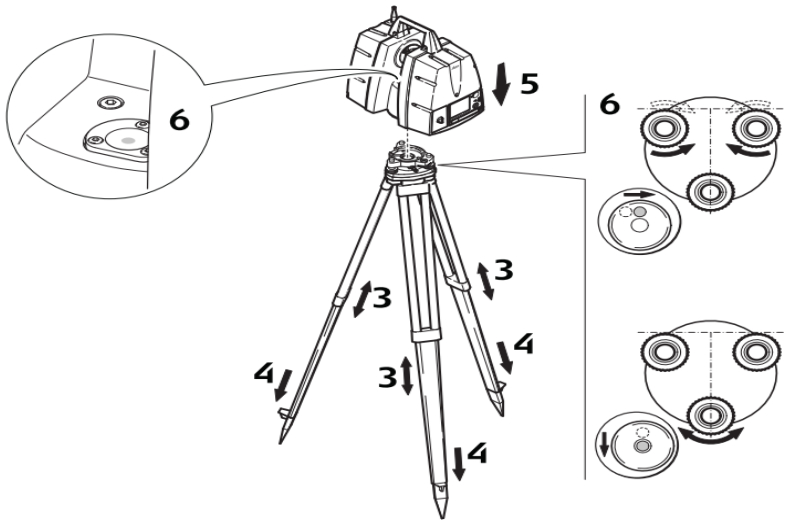


Figure 6.2: *Adjusting scanner position.*[2]

- Use for each scan station the maximum number of targets, even if it may seem superfluous. This redundancy helps to catch up on the data missing.
- A well-positioned target is a target that does not move between a maximum of stations.
- Put at least two targets away from the ground in order to avoid any kind of impediment to register the point clouds (moving object like the infiltration of persons when scanning this target, big distance between target and scanner position, etc.).
- Orient the target's foot towards the scanner in order to obtain all the possibilities for target visibility.
- Set the target bases well. To turn them, hold the base with one hand and turn with the other hand to avoid any error generation.

  Several scans were performed with the Leica P20 LiDAR (Figure 6.3). It takes about 4 hours of work to scan an area containing 10 scan positions with a scan accuracy of 1 to 3 mm every 10 meters and an image resolution of 1200×1200 pixels. This includes the positioning of the targets and scanning process. The post-processing of this amount of data requires three hours. This time includes the time to transfer data from the scanner to the computer, the time of import in the dedicated Cyclone software.

6.2.1.2 Data processing

The software used to align the different scans and assign them a common coordinate system is the manufacturer software **"Leica Cyclone"**. This software is used to manipulate and process the raw data imported from the laser scanner, in order to align them in a single point cloud (Figure 6.4). It also allows observing each scan position as

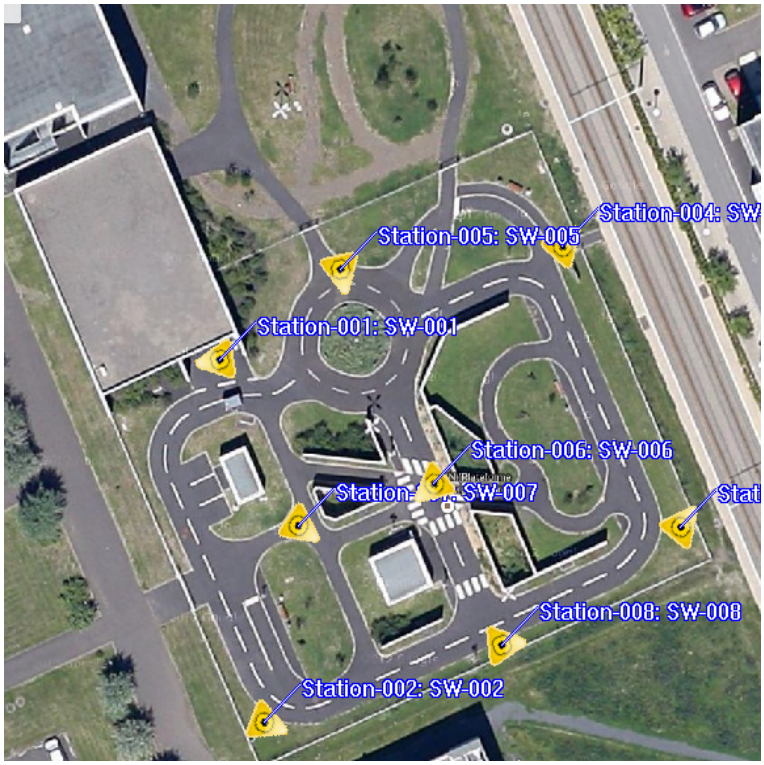


Figure 6.3: *The different position of scanning stations: the yellow triangles indicate the position of the Leica P20 LiDAR. For this experimentation, 8 stations in total are carried out. The stations are not too spaced, and they all overlap (Background image ©2018 Google Earth).*

a panoramic photo of intensity or color, or a 3d view of the scanned area. The latter, however, requires a powerful computer.

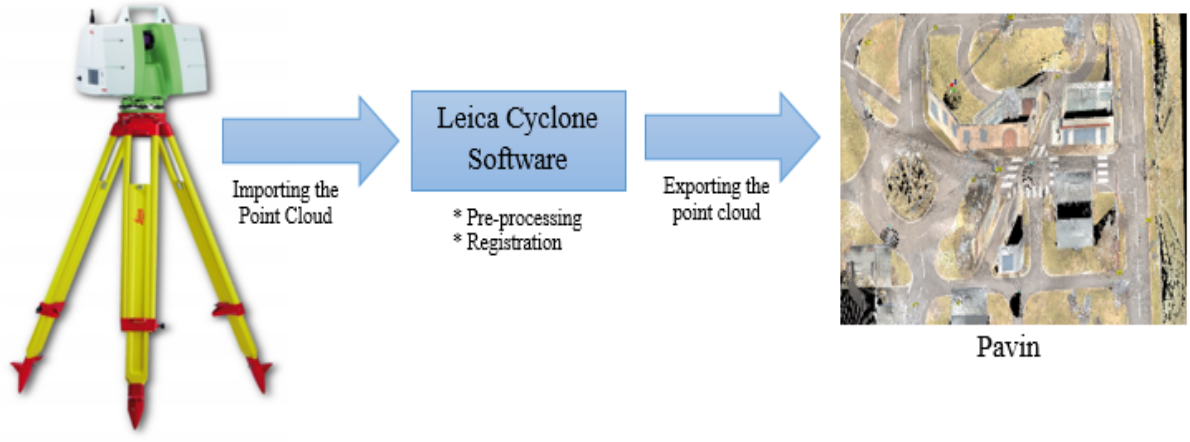


Figure 6.4: *Static map building methodology.*

The registration can be done automatically using the targets that are automatically found by the software. When the different scans are merged and can be considered as a complete single point cloud, this represents the global

map of the environment that is subsequently exported for use in localization purposes. Such a map is illustrated in Figure 6.5.

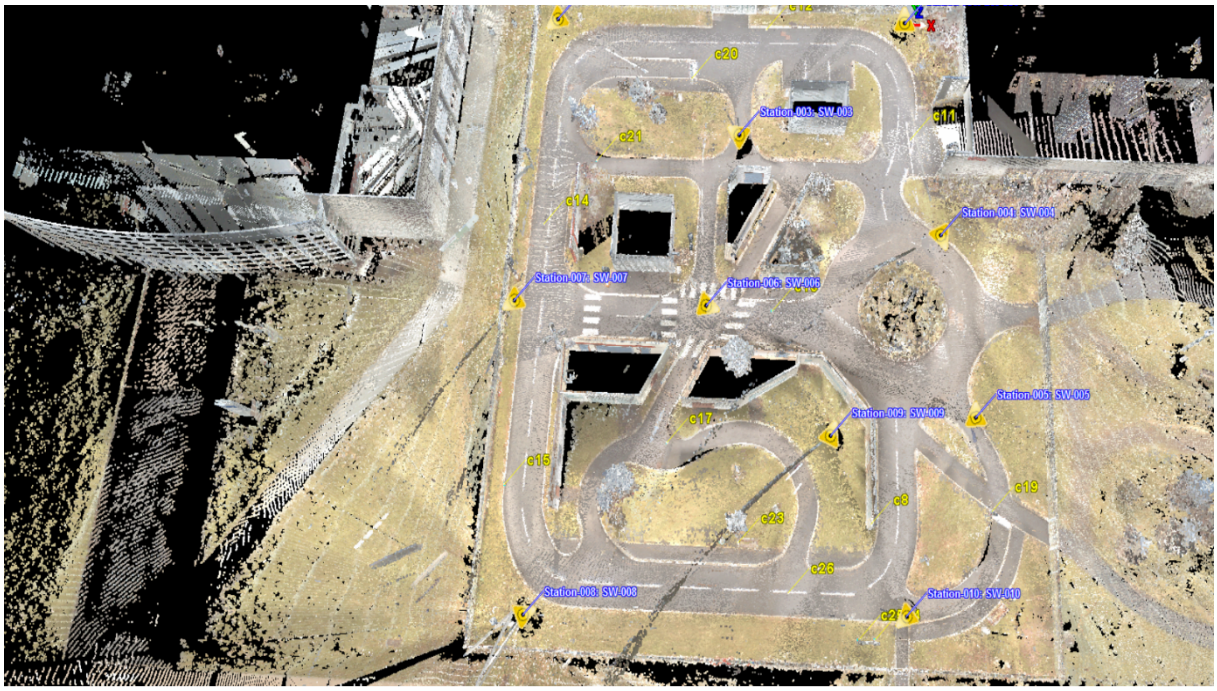


Figure 6.5: *The complete PAVIN datasets comprising 10 scan stations, and over than 500 million points.*

6.2.2 Dynamic acquisition technique

This second technique consists in building the map based on a moving platform. To do this, a variant of SLAM [Zhang 2014] was implemented. In this approach, only LiDAR-generated point clouds are used to build the map and the ICP variant that has been drawn from the previous chapter (chapter 4) was used to estimate the pose. Hence, the vehicle incrementally builds the map as long as it moves and localizes itself within that map.

6.2.2.1 LiDAR odometry

We use an odometry-free mapping and localization technique that is based on LiDAR data. This technique based on "LOAM" [Zhang 2014], which is a state of the art SLAM based scan-matching method using only Velodyne data and without any other information.

The vehicle starts from an unknown position and without any prior knowledge. It begins to receive scans from it on-board Velodyne sensor, one after the other. The vehicle estimates its position by aligning the received point clouds in pairs. The localization is made in relation to an absolute coordinate system corresponds in our case to a geo-referenced position. The map is built from a set of aligned scans. Generally, it is not necessary to add all the incoming scans to constitute the map because there is a risk of overloading it with unnecessary information.

The general principle of the used LiDAR odometry method is schematized in Figure 6.7 and more detailed in Algorithm 5. This is to localize the vehicle over the time on the one hand and build the map incrementally, on the other hand, based on previous reconstructions.

a. Point clouds acquisition

Several sensors can be used to build a 3D map. In our case and for the quest of precision, high refresh rate, a long range and a large field of view, we use a Velodyne HDL 32E sensor, because it represents one of the most suitable sensors for the real-time map building [Choi 2014, Zhang 2015]. The Velodyne HDL-32E produces 3D clouds by rotating a set of 32 beams around its vertical axis at 10 Hz. It generates about 700000 points per second or 2 200 points per laser beam, in a range of 1 to 70 meters. This sensor horizontally provides an angular resolution of approximately 0.16 degrees with a field of view (FOV) of 360 degrees. Its vertical field of view is from $-30,67°$ to $+10.67$ degrees with an angular resolution of 1.33 degree. Its measuring accuracy is usually less than 2 cm. Figure 6.6 shows the Velodyne sensor mounted on the roof of the VIPA test vehicle. Note that the location placement of the Velodyne sensor did not have any specific study. The goal was to only validate the feasibility of the algorithm. For more details on the location and configuration of such a sensor on the vehicle roof, please refer to this study [Mou 2018].

Figure 6.6: *The VIPA vehicle, equipped with the Velodyne HDL 32 E sensor.*

b. Localization

A precise map building cannot be done without a localization process. The vehicle must determine its position in the environment while mapping it as long as it moves. The purpose of the localization method is to determine the position and orientation of the vehicle at each instant t. This localization is achieved through the search for the transformation between a fixed global frame associated to the surrounding environment, and a moving frame

associated to the vehicle, as shown in Figure 3.1. This is equivalent to determine the state vector x_t which expresses the three translations and the three rotations according to the three axes of the coordinate system.

$$x_t = [x\ y\ z\ \rho\ \theta\ \Phi]^T \tag{6.1}$$

From the received scans of the Velodyne sensor, a registration process is performed between every two successive scans. As we have presented in Chapter 4, the registration is based on two main steps: a correspondence between the points of the two scans to obtain a list of matches. Then a pose computing from the obtained matches. This pose represented by its six parameters (three translation and three rotation parameters) represents the local position of the Velodyne sensor. The transformation of this pose into the global coordinate frame gives the absolute position.

More formally, let consider T_i^{i+1} be the transformation between to consequent scans S_i and S_{i+1} acquired at time i and $i+1$ by the Velodyne sensor, and T_c is the transformation between the current scan and the previous keyframe. The displacement between these two scans can be represented by the 6 DoF vector: $[t_x\ t_y\ t_z\ r_r\ r_p\ r_y]^T$ As shown in Figure 6.7, the new scan S_{i+1} is registered against the previous keyframe using the transformation T_c which is calculated from the previous transformations $T_i^{i+1}, T_{i-1}^{i}, \ldots, T_{i-4}^{i-3}$ as:

$$T_c = ({T_{i-4}}^{i-3})^{-1} \times ({T_{i-3}}^{i-2})^{-1} \times ({T_{i-2}}^{i-1})^{-1} \times ({T_{i-1}}^{i})^{-1} \times ({T_i}^{i+1})^{-1} \tag{6.2}$$

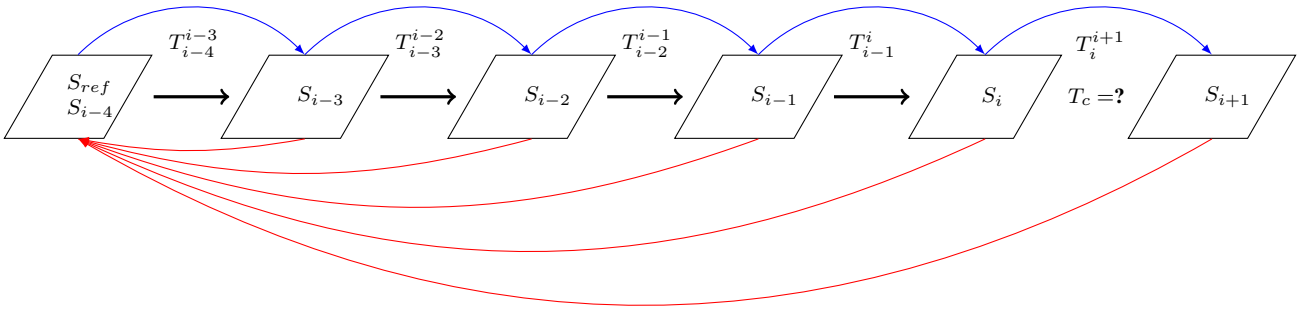


Figure 6.7: *Every incoming scan S_{i+1} is registered against the previous keyframe S_{ref}, and the multiples transformations $T_c^i, T_c^{i+1}, \ldots, T_c^{i+n}$ (red edges) transforming these scans in the coordinate frame of S_{ref}^i, $S_{ref}^{i+1}, \ldots, S_{ref}^{i+n}$ are estimated.*

Because the Velodyne LiDAR is mounted on a moving vehicle, the physical constraints such as the momentum are related to the odometry of the vehicle. Thus, the previously calculated transformation can be used for the prediction and initialization of the next pose estimation (Algorithm 5).

c. Map Construction

To generate the map, a keyframe strategy [Pomerleau 2011] was implemented. First, the keyframes were generated based on a distance threshold, and used to generate the global map. Because for practical reasons and in order to avoid the overloading the map with unnecessary information, it is not necessary to add all the current scans to the constructed map.

The map-building process consists of three main steps:

- selection of keyframes from the stream of current clouds,
- update the map from the selected keyframes,
- local map generation.

We give the details of these three steps afterward.

Keyframe selection To select keyframes, Pomerleau et al [Pomerleau 2011] proposed a solution that consists of holding a keyframe and registering all current clouds against it. For each registration, the algorithm calculates the amount of paired points between the two scans. If this quantity is less than a certain threshold, the algorithm creates a new keyframe from the current cloud. The authors argue that this mechanism reduces the error drift. We adopt this principle in our implementation while choosing as a criterion: a distance-based criterion with a predefined threshold d_{ref}. The change of the keyframe is performed by computing a distance d from the transformation matrix T generated by each registration.

$$d = \sqrt{(T_{14})^2 + (T_{24})^2 + (T_{34})^2} \tag{6.3}$$

The values T_{14}, T_{24} and T_{34} are the first three elements of the last column of the T matrix.

If $d > d_{ref}$, the algorithm instantiates a new keyframe from the current scan. This mechanism is shown by the Figure 6.8.

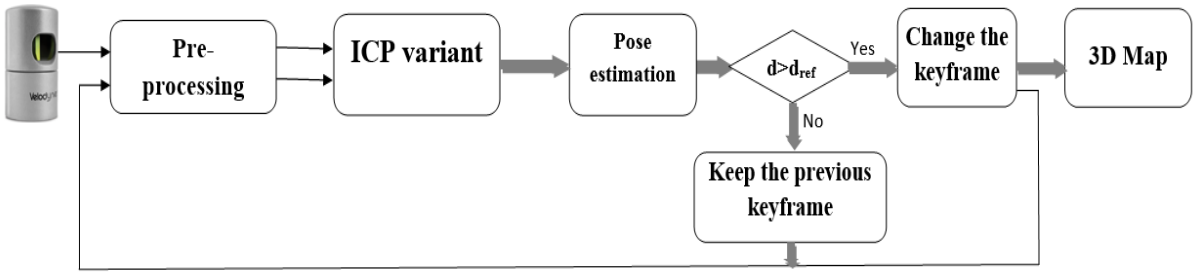


Figure 6.8: *Keyframe selection.*

Update the Map The map consists only of keyframes. This technique allows both to reduce the error drift and to prevent the creation of keyframes (adding new data to the map) if the vehicle remains in the same position. This reduces the information contained in the built-in map by not overloading it with unnecessary information. Therefore, this criterion gives rise to the update of the map with a new scan if and only if it contains new usable information. Due to the reliability of the keyframes update, our technique did not perform loop closure; instead, the keyframe update was used to reset the pose estimated by the LiDAR odometry.

Local map generation In our system, the map consists of a set of raw data. The latter increases with the traveled distance expansion. As a result, the map quickly becomes difficult to manipulate because of this huge amount of data. Therefore, to avoid such a situation, small local maps are generated. The latter is based on two criterions. The first one is based on the traveled distance; we set a predefined distance so that a local map will be generated once this threshold is exceeded. The second is based on the amount of data. Each time a well-defined amount of data is exceeded, a local map is generated. The global map is generated by concatenating all the local maps. This allows limiting the drift, as for each local map generation, the error is reset.

6.3 Experiments

In this section, we describe the map building with the dynamic technique (LiDAR odometry), as the map-creation with the static technique was completely processed using the Leica software packages.

We have based in our implementations on the robotic middleware ROS (Appendix B). Other middlewares can be found such as RTMaps [3] and ADTF [4]. These tools are based on multi-threaded modules, associated with data exchange and synchronization mechanisms. They have several advantages, both in terms of development time and the structuring of the application. They allow an abstraction layer, in order to focus all the energy on the development of the algorithm.

6.3.1 Used methodology

The development of localization techniques requires many experiments to assess their feasibility. In this context, it is important to set up a method that facilitates these tests. Based on the observation that for reasons of complexity and cost of implementation, it is rarely possible to carry out tests directly on the real vehicle. We have structured a methodology that consists of three stages, which are in a successive way: tests in simulation, local test at our scale using our means, general tests on external database taken with meticulous care. We only proceed to the next step once the previous step is validated.

[3] Source: https://intempora.com/products/rtmaps.html
[4] Source: https://www.elektrobit.com/products/eb-assist/adtf/

Algorithm 5: incremental mapping

Input: scan $S_1, \ldots, S_n$
Output: map, currentPosition
Initialize: referenceCloud, currentCloud, transformedCloud, currentPosition, IntialPostion
    $T_{guess} = T_{init} = T_{ref} = T_{curr} = T_{curr\_ref} = T_{curr\_world} = T_{Identity}$
begin
    // Retrieve the intial position and orientation of the robot
        IntialPosition = getIntialPosition ()
    // Create the intial transformation matrix
        $T_{guess} = [R, t]$
    // Transforme the first scan from the velodyne frame to the world frame
        currentCloud= transformePointCloud (s_1, T_{guess})
    // Add Intial point cloud to the map
        map = referenceCloud = currentCloud
    for $i = 2 \quad to \quad N$ **do**
        currentCloud = transformPointCloud (s_i, T_{guess})
        // Registration: Align the current cloud to the reference cloud
            $T_{curr\_ref} = register(currentCloud, referenceCloud, T_{init})$
        // Calculate the sensor pose in the world based frame
            $T_{curr} = T_{ref} \times T_{curr\_ref}^{-1}$
            $T_{curr\_world} = T_{curr} \times T_{guess}$
        // Update the position and the orientation of the robot
            $currentPosition = getXYZ\_RPY(T_{current\_world})$
        // Calculate the traveled distance between the current position and the reference position
            $dist = \sqrt{T_{curr\_ref}(1,4)^2 + T_{curr\_ref}(2,4)^2 + T_{curr\_ref}(3,4)^2}$
        if $(dist >= Kf_{thre})$ **then**
            transformedCloud = transformPointCloud (currentCloud, T_{curr})
            // Insert the Keyframe in the map
                map + = transformedCloud ;
            // Change the reference cloud by the current cloud
                referenceCloud = currentCloud ;
                $T_{ref} = T_{curr}$
                $T_{init} = T_{Identity}$
        else
            // Stay holding the previous reference cloud and match every incoming cloud against it.
                $T_{init} = T_{curr\_ref}$
        end
    end
    return *map, currentPosition*
end

6.3.1.1 Simulation

For every robotic application, the simulation is an indispensable tool [Hossain 2018]. It can make a very prominent difference, by accelerating the development and maintenance processes of the robot application. The contributions of the simulation are no longer to be demonstrated. We can mention among others:

- Test ideas before building: quickly identify whether ideas are feasible or not with almost no expense.
- The perfect control of the environment: an absolute ground truth is available at any time.
- Test in different environment and situation: if errors occur, correct them on simulation and only use the real robot once errors are corrected.
- Ease of implementation of test scenarios and therefore acceleration of development cycles.

Based on the previous points, we have used two 3D simulators: Gazebo and RVIZ (refer to Appendix B for brief detail of these two simulators).

6.3.1.2 Real tests

The simulation remains an approximate tool of reality, it will be necessary to carry out real tests to validate the proposed techniques.

a. Local tests:

To validate our map-building strategy proposed above, experiments took place in two different environments. An external environment represented by The PAVIN platform, and an indoor environments represented by the corridor of the ground floor of building 3 of Institut Pascal. For the indoor environment, we started by the reconstruction of a room and then a whole corridor. For both cases, we have established a protocol of two measurements strategies. For the first one, the measurements were acquired using the "Stop-and-Go" strategy. Ground marks that define a preset location are used as shown in Figure 6.9. These positions are spaced approximately 20 centimetres apart. We perform a single scan in each position. The second was continuously scanning using "On-Drive" strategy; the Velodyne was placed on a trolley and displaced manually.

For the outdoor tests, we use the vehicle VIPA 4 that is designed to serve as a prototype for research and development in the field of autonomous urban transport vehicles in the Institut Pascal. It can carry 4 people and theoretically drive up to 40 km/h. The tests are unfolded in the PAVIN platform. The Velodyne sensor was placed on the roof of the VIPA at a height of 75 cm as shown in Figure 6.6.

The test platform is a laptop equipped with an i7 processor running at 2.7 GHz, and a RAM of 32 GB.

(a) (b)

Figure 6.9: *Left: The scan positions for the "Stop-and-Go" strategy. Right: Acquisition device.*

b. Extern dataset

There are many test datasets in several domains. Particularly, the autonomous driving field has received increasing attention recently, due to the popularity of the self-driving technology. The objective is to understand the challenge of computer vision systems in the context of autonomous driving on the one hand and the development and comparison of algorithms on common data on the other hand. In our work, we are interested in the LiDAR specialized datasets, as our thesis focuses on LiDAR-based localization technique. Among the public datasets, we find Ford campus [Pandey 2011], KITTI dataset [Geiger 2012], or more recently KAIST Urban Dataset [Jeong 2018]. Good comparison tables of the main existing datasets can be found in [Korrapati 2013, Bresson 2017]. In our tests we have used KITTI dataset, because in a few years this latter has established itself as a benchmark for comparing algorithms and has become the most used in the field of autonomous cars. It has many different types of environments: dense urban, rural roads, and highways.

6.3.2 Choice of different optimization parameters

Our approach has been coded in C++11. ROS Indigo and PCL 1.8 were used for perception and 3D geometry processing. The EIGEN [5] library has been used for all matrix and vector operations, and the FLANN library [Holz 2015] has been used to the nearest neighbors search.

Many parameters needed to be fixed in our algorithm:

- We have subsampled point clouds with 1/8 ratio, because of the frequency rate (10 scans per second) and the

[5] Source: http://eigen.tuxfamily.org/index.php?title=Main_Page

density cloud delivered by the Velodyne sensor. We use PCL VoxelGrid [6] for the subsampling. This allows accelerating optimization without losing precision.

- We used a rejection distance between 20 and 50 centimetres depending on the environment. Several values were tested, and this threshold gave the best optimization results.
- For the registration stopping criteria, we have taken a threshold of 10^{-3} meters for the translations and 10^{-4} degrees for the rotations.
- The maximum iteration number for each registration was set at 40.
- The distance threshold for the keyframe change is set between 1 and 2 meters.

6.4 Results

6.4.1 Results on Simulation

We tested the LiDAR odometry approach with the VIPA and PAVIN models in Gazebo. The displacement was on a single axis, as there is no implemented path planning strategy. To move the VIPA, a linear speed is applied on one axis at a time. The initial position is given by the ROS Modelstate [7] topic. The map-building is carried out using the selected variant based on point-to-plane metric and Gauss Newton minimization. The speed of the vehicle can go up to 5 m/s. Figure 6.10 shows the obtained map.

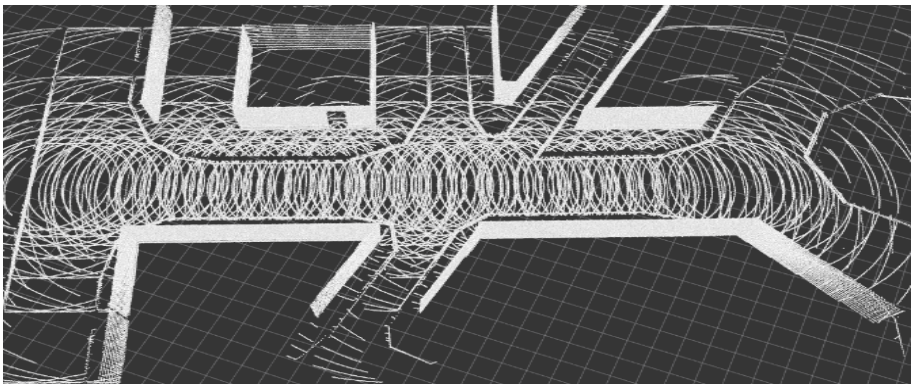


Figure 6.10: *The global map obtained in simulation.*

6.4.2 Results on real data

6.4.2.1 Indoor environment

The corridor of the first floor on the building 3 of the Institut Pascal Laboratory was chosen for the first experiment. The corridor is about 50×2 meters. We describe the reconstructions performed with the two strategies described

[6] Source: http://pointclouds.org/documentation/tutorials/voxel_grid.php
[7] Source: http://docs.ros.org/melodic/api/gazebo_msgs/html/msg/ModelState.html

above:

a. The "Stop-and-Go" strategy

Figure 6.11 illustrates an indoor built map using the Stop and Go strategy. The entire sequence takes place in the same hallway.

The obtained trajectory is indicated by red spheres in Figure 6.11. A sphere represents the estimated pose at each registration.

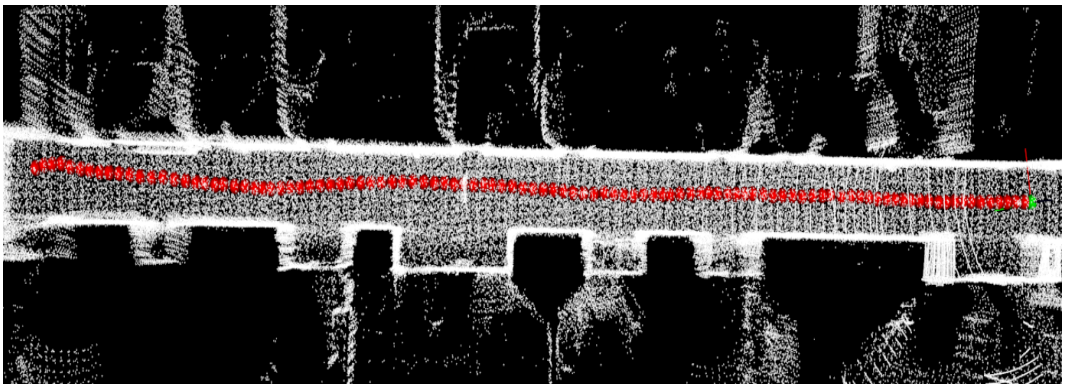


Figure 6.11: *Corridor reconstruction with "Stop-and-Go" strategy.*

The mean square error of each registration is given in Figure 6.12.

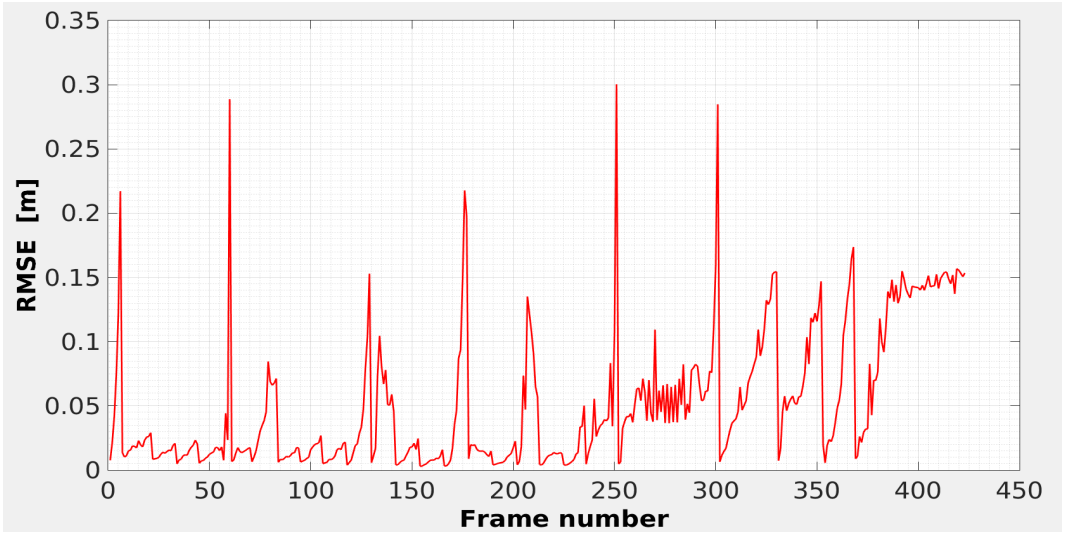
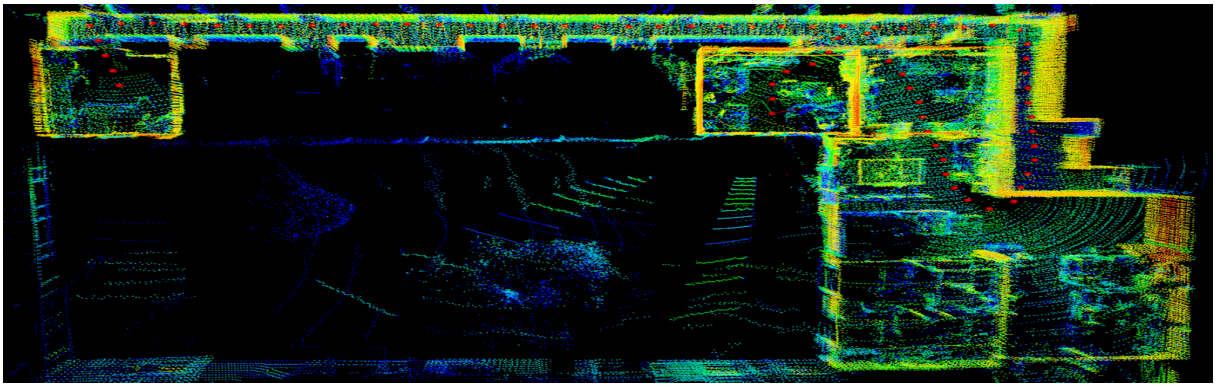


Figure 6.12: *The frame to frame RMSE of the corridor sequence with the Stop and Scan strategy.*

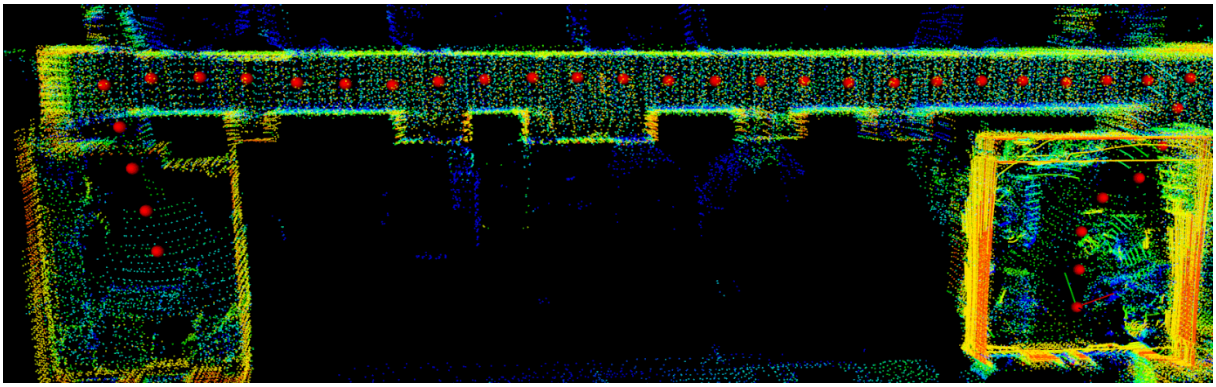
The spikes exhibited in Figure 6.12 correspond to most probably to the passage of the sensor in a region dominated by glass-doors along the hallway. Therefore, as expected the data is corrupted by noise due to multiple reflections.

b. The “On-Drive” strategy

In this strategy, the Velodyne was placed on a cart together with a battery and a laptop computer. One person pushes the cart and walks. This sequence generated over 1300 scans. Reconstruction is given in Figure 6.13. The mean square error of this reconstruction is shown in Figure 6.14.



(a) Top view of the corridor



(b) Zoom on the image

Figure 6.13: *Indoors reconstruction with “On-Drive” strategy.*

In this reconstruction, the distance threshold for the keyframe change is set at 1 meter. The colors of blue to red on the previous figures correspond to the scale of the intensity of the reflectance.

By comparison between Figures 6.12 and 6.14, we find that the mean squared error in the case of the continuous Scan strategy is better than the Stop and Go strategy. This is due to the keyframe change criterion, which plays an important role in reducing drift. The latter experiment "On Drive" has been done in a hallway leading to a cluttered environment.

6.4.2.2 Outdoor environment

In the outdoor environment, we have introduced several datasets with the Velodyne HDL 32 Lidar, mounted on the VIPA 4 like-vehicle. The sensor was placed on the roof of the vehicle at a height of 75 centimetres. The distance

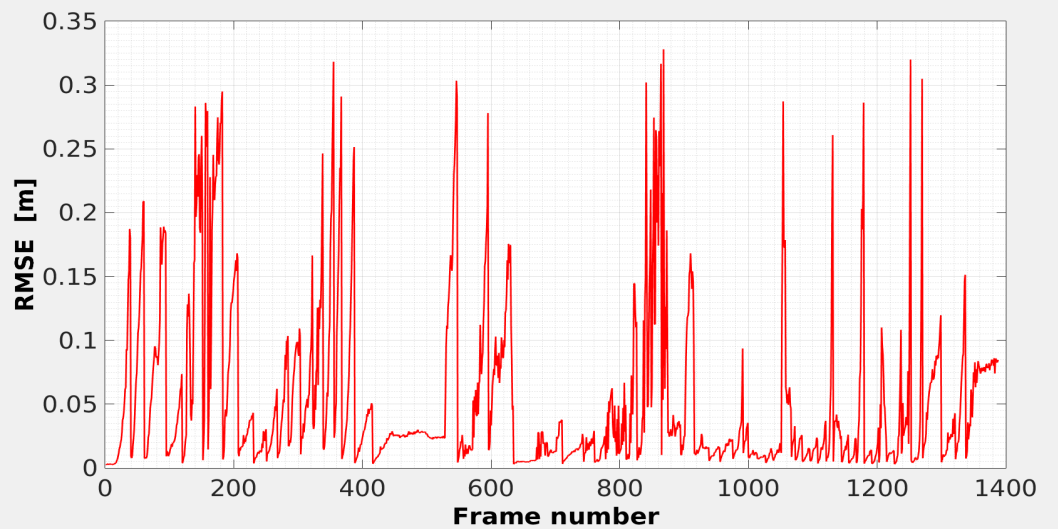


Figure 6.14: *The frame to keyframe RMSE of the corridor sequence with the "On Drive" strategy.*

threshold for the keyframe change is fixed at 0.5 meters. The displacement of the vehicle generates problems that affect the scans geometry. For this reason, low speeds that do not exceed 2 m/s are recommended. This speed gives rise to consecutive scans shifted by 20 cm on average, as the Velodyne refresh rate is of 10 Hz.

PAVIN sequence

The reconstruction of the test performed on the PAVIN platform is illustrated in Figure 6.15. The trajectory measures approximately 200 meters. It consists of 1941 frames of approximately 70 000 points each.

Colors from blue to red in Figure 6.15 correspond to the scale of the reflectance intensity. The use of the keyframe strategy for the map building allowed to compress the 1941 initial frames in 473 frames.

Figure 6.17 shows the appearance of a drift on the final reconstruction. This drift is due to the motion-distortion effect that LiDAR suffers when moving-while-scanning, and to the accumulation of motion estimation errors since our method does not recognize loop closure. In order to fix this, it is necessary to post-process the point-clouds to eliminate motion-distortion and to consider a method of loop closure or/and a fusion with IMU data.

Without ground truth, it is difficult to evaluate the performance of our implementations in the two previous indoor and outdoor sequences. Although visually, the results seem good, the trajectory formed from the estimated poses proves correct and the registration errors between every two successive scans are of the order of centimeters. In the next section, we will evaluate our method in the public KITTI dataset provided with the ground truth, which will enable us to properly evaluate the proposed approach. Unlike to our data, KITTI datasets are already processed, which eliminates the motion-distortion effect and decreases the final drift.

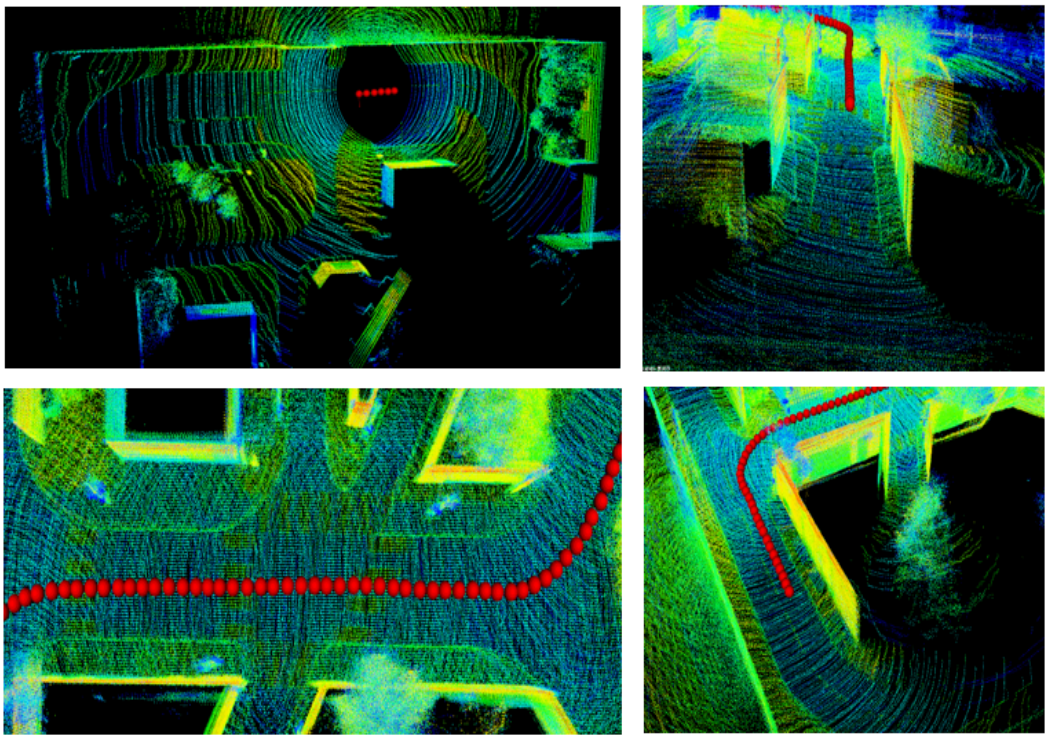


Figure 6.15: *Sample scans from the PAVIN map building (PCL viewer overview).*

6.4.3 Results on external datasets

We used the KITTI odometry dataset to evaluate our LiDAR odometry technique. This dataset consists of 10 sequences of over 21 km captured with Velodyne LiDAR HDL 64, and ground truth data obtained by GPS/OXTS. These sequences are carefully recorded with the LiDAR sensor mounted on the roof of a vehicle driving on structured roads. They are composed of a wide variety of environments such as urban cities (sequences 00, 05, 06, and 07) and highways (sequence 01) with high traffic, rural road with low traffic and a lot of vegetation (sequences 02, 03, 04, 08, and 09). These sequences are provided only on 3D point cloud and not on the raw data. They were handled to eliminate the effect of motion-distortion, therefore all points in a point-cloud are treated as being measured at the same time [Tang 2018]. This allows reaching high speeds while building good quality maps.

6.4.3.1 Evaluation

To evaluate the odometry estimation, we have used three metrics: RMSE, the relative translational error, and the relative rotational error. The former describes the root mean square error of each point cloud registration.

$$RMSE = \frac{1}{n}\sum_{i=1}^{n} \parallel E_i \parallel_2 \tag{6.4}$$

where E_i represents the error of a single point cloud registration, and N is the total number of frames.

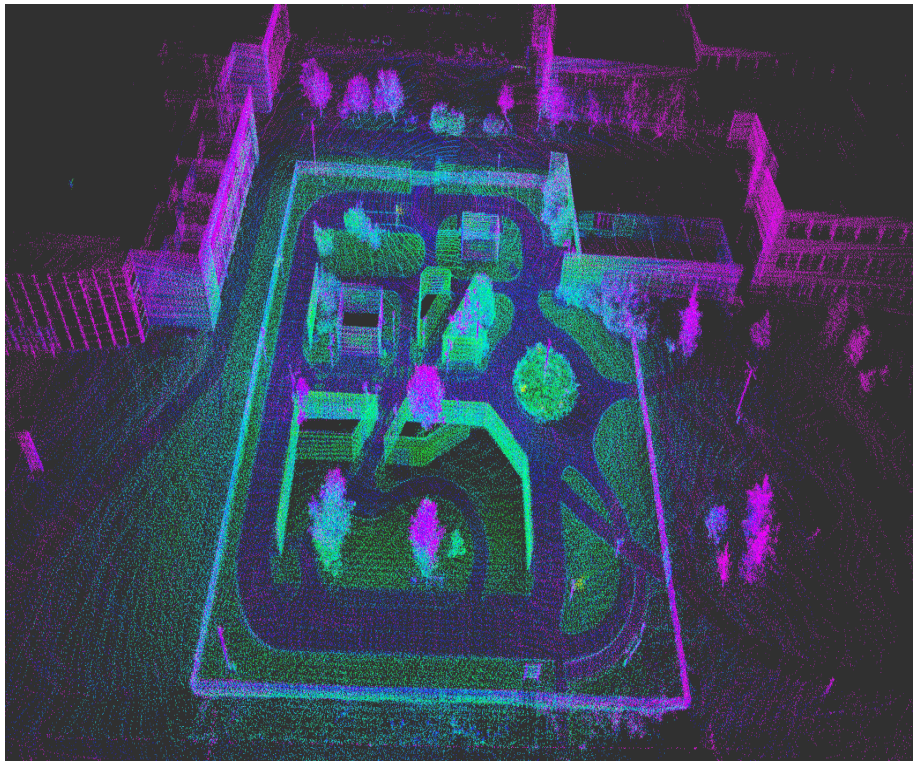
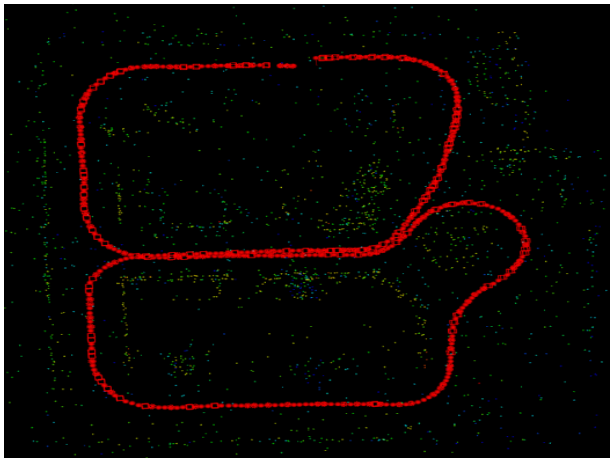


Figure 6.16: *Reconstruction of the PAVIN platform with 473 keyframes and 1941 scans in totality (RVIZ overview).*

(a) View of PAVIN with Google Map



(b) Final trajectory calculated from estimated poses, length: approx. 200 m

Figure 6.17: *Trajectory reconstruction with the estimated poses.*

The second metric is the Relative Translational Error (RTE), which measures the translation gap between the ground truth ($\mathbf{t}_{GT}$) and the estimated ($\mathbf{t}_E$) translation vectors.

$$RTE = \|\mathbf{t}_{GT} - \mathbf{t}_E\|_2 \tag{6.5}$$

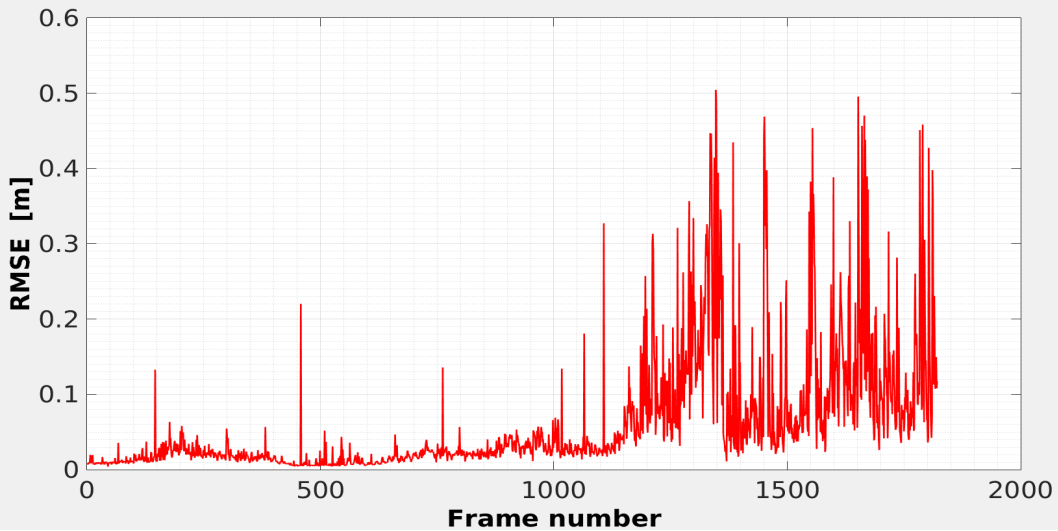


Figure 6.18: *The frame to keyframe RMSE of the PAVIN sequence with the "On Drive" strategy.*

For the Relative Rotational Error (RRE), we use the metric defined on the tangent space of $\mathbb{SO}(3)$:

$$RRE = \| \log\text{m}(\mathbf{R}_E^T \mathbf{R}_{GT}) \|_F \tag{6.6}$$

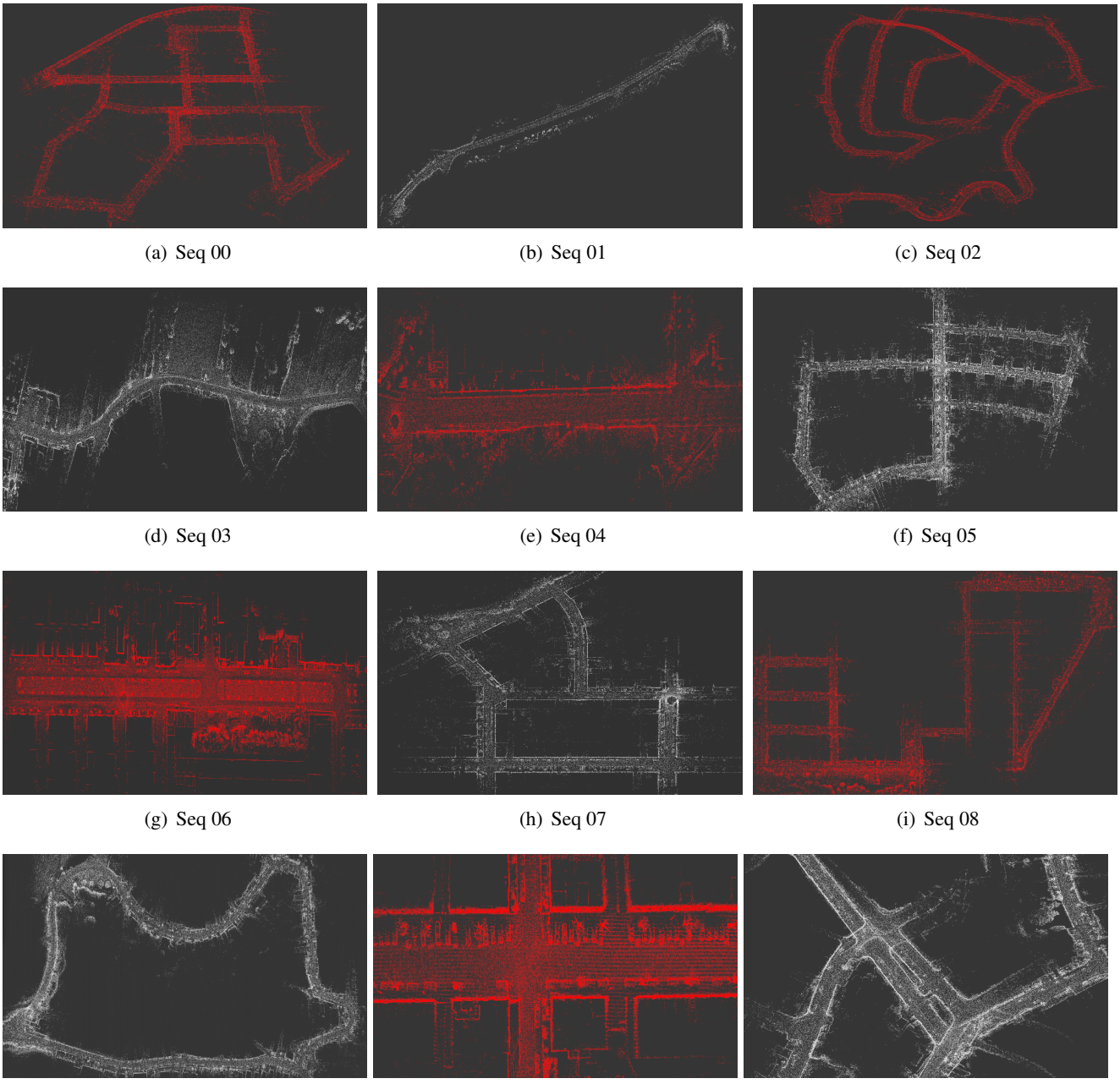
where $\log\text{m}(.)$ is the matrix logarithm, $\mathbf{R}_E$ is the estimated rotation matrix, $\mathbf{R}_{GT}$ is the ground truth rotation matrix and $\|.\|_F$ is the Frobenious norm.

Table 6.1 presents the results found for the 10 sequences. Note that, the fifth and the sixth columns corresponding to the RTE and RRE metrics respectively, are error-affected due to the poor estimation of the vertical component along the Y-axis. Since the ground truth data were obtained by a GPS sensor, which gives a significant imprecision in the estimation of the vertical Position.

Table 6.1: *The evaluation of the odometry estimation for the KITTI data sequences provided with the ground truth.*

Sequence no.	Length (m)	Length (frames)	RMSE (m)	RTE (m)	RRE (°)
00	3718.98	4540	0.2268	9.9118	2.5312
01	2434.54	1100	0.1810	5.1327	1.4600
02	5045.19	4660	0.1391	32.6728	2.3645
03	560.01	800	0.1260	10.9326	2.1662
04	391.38	270	0.1311	4.6222	**0.0291**
05	2806.16	2760	0.1385	8.9344	2.1519
06	1232.56	1100	0.2703	4.2092	1.1840
07	693.44	1100	**0.0567**	**1.1497**	2.2657
08	3214.07	4070	0.1605	11.3617	2.3971
09	1700.06	1590	0.1661	11.1054	2.4818
average	2179.63	2108	0.1595	10.0032	1.9031

Based on the results of Table 6.1, the best RTE result (bold value) is obtained from the sequence 07



(a) Seq 00 (b) Seq 01 (c) Seq 02

(d) Seq 03 (e) Seq 04 (f) Seq 05

(g) Seq 06 (h) Seq 07 (i) Seq 08

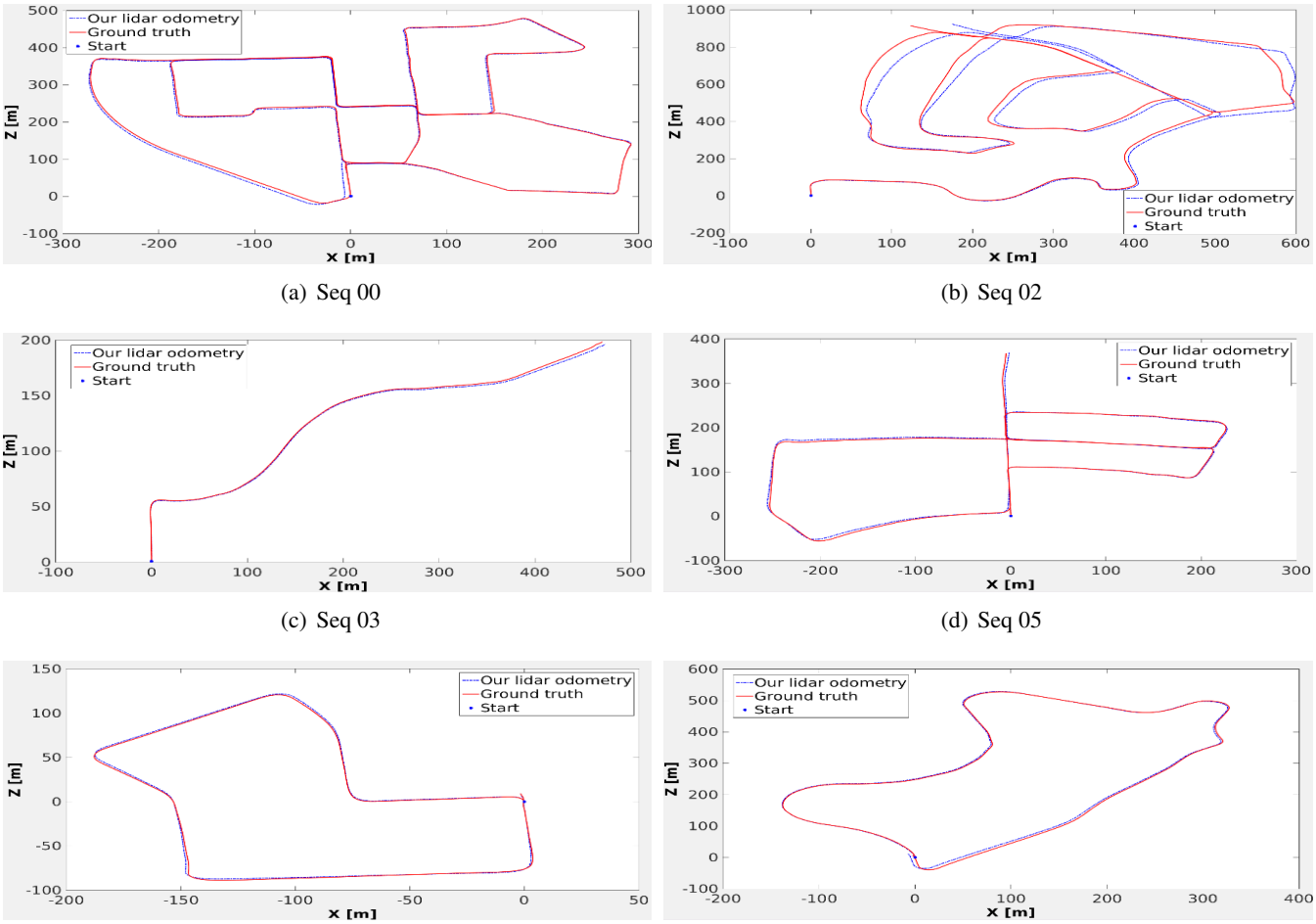
(j) Seq 09 (k) Zoom Seq 00 (l) Zoom Seq 02

Figure 6.19: *The maps produced by our LiDAR odometry in the 10 sequences of the KITTI dataset.*

(Figure 6.19(h)) with an offset of 1.14 meter. The reason behind this is due to the nature of this environment, which is an urban-like environment. Figure 6.20(a) represents a comparison between the pose found by our method (blue dashed line) and the pose given by the ground truth (continuous red line). On the other hand, the worst result (italic value) is obtained from the sequence 02 (Figure 6.20(c)). The latter was captured outside the city center in a rural-like environment. It captures mainly trees, shrubs, and other natural phenomena outside the road. In addition,

its long length (5 km) promotes the drift accumulation. This is clearly visible in Figure 6.20(b) which represents the pose found by our method (blue dashed line) and the pose given by the ground truth (continuous red line). In a qualitative way, Figure 6.19(k) shows a zoom on a part of this environment where the offset on a road is obvious.

For the RRE, the best result is obtained in the environment of the sequence 04, which represents a straight road of almost 400 meters (Figure 6.19(e)). This environment does not contain any bends. In contrast to the environment of the sequence 00 which has the worst RRE Value. The latter shown in Figure 6.19(a) was recorded on an open highway contains several bends. Its RRE can be improved by implementing a loop closure detection technique.



(a) Seq 00 (b) Seq 02

(c) Seq 03 (d) Seq 05

(e) Seq 07 (f) Seq 09

Figure 6.20: *Odometry estimation comparison between the pose obtained from our LiDAR odometry (blue dashed line) and the ground truth obtained by GPS/OXTS (continuous red line).*

6.5 Conclusions

In this chapter, we presented two map-building methods intended for localization of the autonomous driving purpose. The first method is static; it delivers dense and metrically accurate maps. However, it is a relatively

slow method that requires human intervention, at least to the use of the instrument. Given the target areas and their number for the autonomous navigation, it is not conceivable to use this scanning method. It is therefore obvious that it is necessary to use a dynamic method with embedded LiDARs for the map building. In our case, as we do not have a mobile laser scanner (MLS) such as Leica-Pegasus [8] or RIEGL VUX-1HA [9], etc., we validated the proposed approach with a Velodyne HDL-32. The adopted approach uses an odometry-free mapping and localization technique that is based only on LiDAR data. To generate the map, a keyframe strategy was implemented. First, the keyframes were generated based on a distance threshold, and used to generate the global map. This technique allows both to reduce the error drift and to prevent the creation of keyframes (adding new data to the map) if the vehicle remains in the same position. This reduces the information contained in the built-in map by not overloading it with unnecessary information. Therefore, this criterion gives rise to the update of the map with a new scan if and only if it contains new usable information. Due to the reliability of the keyframes update, our technique did not perform loop closure. As a result, it accumulates errors during the mapping. If the distance of the map is short, the errors are minimal. Nevertheless, the larger the size of the map, the more the error in the 3D map can also be important, and consequently cannot be ignored. In this case, a loop closure method is needed to solve this problem. Finally, we have performed multiple experiments to evaluate the proposed LiDAR odometry method. The tests have been conducted in both indoor and outdoor environments as well as on the KITTI odometry datasets. Built maps prove to be of a good quality and are ready to be used for localization. It is in this context that we will now address the next chapter presenting a precise localization in 3D prior map for autonomous driving.

[8] Source: https://leica-geosystems.com/fr-FR/products/mobile-sensor-platforms/capture-platforms/leica-pegasus_two

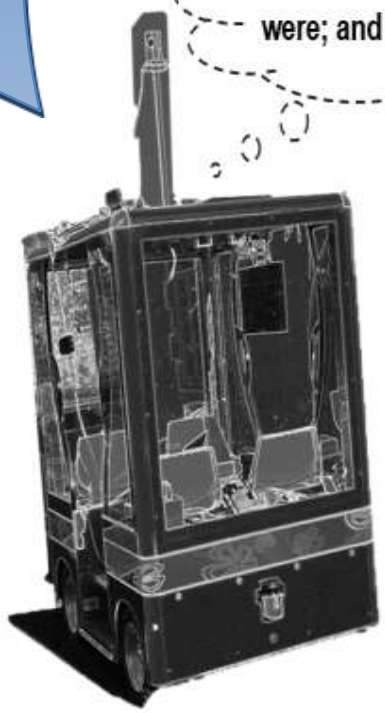
[9] Source: http://www.riegl.com/products/newriegl-vux-1-series/newriegl-vux-1ha/

Chapter VII

Point cloud reduction

"You see things; you say, 'Why?' But I dream things that never were; and I say 'Why not?"

George Bernard Shaw



CHAPTER 7

Point Cloud Reduction

Contents

In our quest for precision, we have chosen very powerful sensors, such as the Leica P20 LiDAR. These tools allow obtaining very dense point clouds. However, the interest of such a quantity of information is not always justified. Thus, the process of sampling (simplification) of point clouds becomes an essential step in the process of treatment. It determines the relevance and accuracy of the following steps. In this chapter, we propose an original sampling approach. This approach is based on the use of both color information and the geometry of the scene. First, a voxelisation is performed to maintain the topological details of the scene and then, for each voxel, a classification according to the colors of the points is performed. Then a point of each color class is preserved and the other points are deleted. This method was developed as part of our team's participation in the project FUI ROMAPE.

7.1 Introduction

From past decades, 3D scanners have emerged and evolved as powerful tools which have been applied in various fields. This growth was driven by the need for accurate, efficient, safe and fast modeling enabling a comprehensive analysis of the scene [Puttonen 2013]. Professionals from all sectors who usually worked with two-dimensional plans and schematic representations (robotics, aerospace, automotive, architecture, naval, etc.) now have the opportunity to give a new dimension to their tools. Several objectives can be at the origin of a three-dimensional modeling. It may be, as in this study, the 3D map used by the vehicle to localize itself.

The use of color in the fields of computer vision and robotics perception provides a rich and quality information. Undoubtedly, this quality of information has stimulated the performance of variety of applications in these areas.

For the purpose of simplifying the amount of 3D data, the traditional digital chain can be synthesized in three successive stages [Lee 2008]: 3D acquisition, sampling and mesh generation.

With the high-speed evolution of RGB-D sensors, it is now possible to obtain very dense point clouds [Song 2009]. A sensor like Kinect produces a cloud with more than $300\,000$ (640×480) points, which represents a lot of information. Furthermore, 3D laser scanners provide higher resolution (around two orders of magnitude) [Wiemann 2014].

As point cloud processing is directly proportional to number of available points, in terms of processing time and data storage, it is rather disadvantageous to have more points than required [Li 2012]. Because these additional points slow down the processing time and need data storage space, without giving any more additional information [Puttonen 2013]. To the best of our knowledge, to perform a simple operation on every point of a cloud, it would be of $\mathcal{O}(n)$, n being the number of points. To compare every point with its k nearest neighbors, it would be $\mathcal{O}(nk)$. It is clear now why some algorithms processing the 3D point clouds take several seconds to run. Current scanner devices tend to provide a lot of redundant information (the region of interest represents a small percentage of acquired cloud). Thus, expensive pre-processing steps are required to reduce the number of points by selecting only the relevant ones. The successive steps leading to the mesh model tend naturally to decrease in number of points, because mesh operations are quite complex and computationally expensive [Pauly 2002].

The objective of our approach is to reduce the number of 3D points while preserving the amount of information carried by these points.

7.2 Related Work

Today's scanners are able to achieve millimeter accuracy with a high resolution [Li 2012], supplying the user with a huge amount of data e.g., a scan of an object of few centimeters can produce hundreds of thousands of points. Therefore, it is essential to simplify this redundant information to make it tractable for a standard computer [Song 2009]. Reducing the number of points may be performed during the measurement process by setting a suitable scanning resolution or by limiting the scanning angles [Puttonen 2013]. In this research, we are rather interested in reducing the point cloud after the measurement instead of limiting it during the acquisition time, and so acquiring as much information as possible, in order to get the more precisely map. Given the amount

of work on this topic, we do not present an exhaustive survey. However, we will cover an overview of the subject where the main approaches are described.

A large number of pre-processing approaches aim to reduce the density of the cloud, by removing all points that are unnecessary. Within these approaches, there are several ways to achieve this, such as trim all outliers from the cloud, then perform segmentation to extract only the parts of interest. A common operation is also sampling, which returns an equivalent cloud to the original one, with fewer points. This equivalency was defined in [Puttonen 2013] such that the data in the sampled cloud have similar properties (e.g., color, intensity, position, etc.) than those in the original one, while the sampling ratio is reduced as long as possible.

It is conventionally accepted that the point cloud reduction process is based on three main groups as shown in Fig. 7.1. The simplification by partitioning is done via voxelization [Li 2012]. First, the cloud is divided in multiple cube-shaped regions with the desired resolution. Then, all points within every voxel are processed such that one remains. The simplest way would be to randomly select one of them or the voxel center, but a more accurate approach would be to compute the centroid.

As a variation of this technique, Lee et al. [Lee 2001] propose a point reduction method using 3D grids. The method compartmentalizes the points by a spatial decomposition process based on the principle of octrees. This spatial decomposition is performed using the standard deviation of the normal vectors estimated from points within each cell. Therefore, the cell size corresponds to the curvature generated by the points of each cell. The reduction is then obtained by selecting a representative point in each cell.

The authors of [Lee 2006] present a point simplification algorithm using local coplanar analysis on the basis of an octree data structure. This algorithm performs a recursive and hierarchical partition of the point cloud. This subdivision is stopped when the points of each division form a single plane. This ensures a dynamic subdivision based on variations of cloud surfaces. The nearest point of the centroid of each voxel is chosen as a representative point in the simplification step. However, this method deforms considerably the shape of the model which contains many flat surfaces, where a large number of points are reduced to a single point.

In the work presented in [Se-Ho 2012], the point cloud reduction is not the principal purpose of this research, but it is an important step of the process which concerns the development of mobile system for 3D environmental data representation. The authors adopt the voxel-based simplification, with a centroid representative of original 3D point within each voxel.

Reduction points via voxelization method is simple, fast and memory-efficient [Li 2012]. Nevertheless, these methods still have some drawbacks which can be summarized by their difficulty in controlling the partitioning surface resolution, and by the large topological errors that appear when the size of the voxel increases. The second

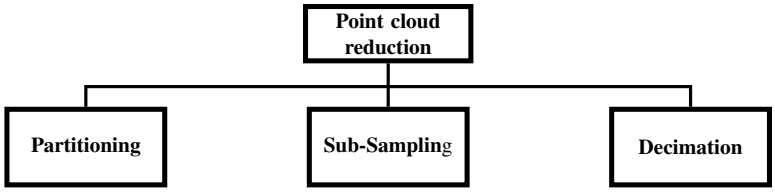


Figure 7.1: *Classification of 3D point cloud reduction methods.*

group of point cloud reduction techniques is subsampling which may be implemented as a pre-calculated binary

mask applied to the point cloud. As a general rule, the subsampling can be applied in a random way by just fixing the reduction rate [Masuda 1995] or with a uniform selection strategy [Zhang 1992, Blais 1995]. Boissonnat and Cazals [Boissonnat 2001] present a data reduction algorithm that randomly calculates a subset of points and builds 3D Delaunay triangulation. This algorithm adopts a costly pre-processing to preserve the original surface data before simplifying the point set. Alexa et al. [Alexa 2001] propose tools to increase or decrease the density of the points, thus, allowing an adjustment of the spacing among the points to control the fidelity of the representation.

The work introduced in [Puttonen 2013] focuses on subsampling methods and how they impact the information carried by the points before the reduction. It provides two methods to reduce the point cloud data while maintaining the maximum content of the original cloud. These methods are called level histogram sampling and inversely weighted distance sampling. They use the information of the distance distribution in the cloud to carry out the reduction. These sampling methods are tested in three separate cases in which data are collected through terrestrial or mobile laser scanning system. Two reference methods (uniform and linear point picking sampling) are used to compare the results. The performance of the method is evaluated using two criteria. The first criterion is to preserve information carried by the original cloud. The second criterion is the change in the total processing time including the time spent for sampling.

The principle of decimation algorithms [Song 2009, Li 2012] lies in the definition of an importance function for each point of the cloud. For this, several metrics are used such as those based on neighborhood, position or normal. Once this function is defined, points are sorted according to their importance in a priority queue. Then, the points that have minimal metric function are removed. For the points in the vicinity of deleted points, the importance function must be recalculated and the queue is updated. This process is repeated with every point of the cloud until the satisfaction of the stop condition that can be a number of target point or a maximum allowable error. Algorithm 6 summarizes the principle of decimation.

The first algorithm which uses decimation was proposed by the authors of [Dey 2001]. It exploits the Voronoi diagram to detect the redundancy in the input point cloud and therefore the points to be deleted. An input parameter ρ is also used to determine the level of decimation. However, this method requires many computations [Lee 2006].

Another paper that addresses the point cloud reduction by decimation is proposed by the authors of [Lee 2008]. It is based on the Discrete Shape Operator (DSO) to classify all the points in a propriety queue. The DSO attributes high values for the feature points and a small values for the other points which present small variation. The simplifying process carries on the points with a small value in the priority queue. This procedure is achieved in two steps. The first one consists in detecting the neighboring points of each point: the distance to the nearest neighbor is multiplied by a constant, and the sphere with this value as radius is considered. Thus, all points within this area are taken as neighbors of the given point. The second step is the computation of normal at each point. For that, the best-fit plane of neighboring points is estimated using the least square method. Thereafter, the method uses the normal vector and neighbors points to estimate a discrete shape operator DSO of each point. This latter is considered as the importance function of the decimation process. The DSO is calculated by combining two quantified functions, one to estimate the curvature and the second for the torsion. The curvature denotes the variation of the tangent vectors, and the torsion denotes the variation in the binormal vectors. The root of the sum of squares of the two functions denotes the degree of local surface variation.

Song and Feng [Song 2009] propose a similar algorithm for the decimation of mechanical parts with edges and corners. The basic idea is to keep the points pertaining to the edges and corners in the final reduction as

Algorithm 6: Reduction by decimation.

1: Sort the points according to their **importance** in a priority queue **F**
2: Remove the head point **F** (minimum error)
3: **Optimisation**: recalculate the importance of the adjacent points of the deleted point
4: Update **F**
5: Goto 1 until the stop condition
- Number of target points
- Maximum allowable error

they help to define geometrical shapes, while points pertaining to smooth surfaces are largely removed. In the first step, the algorithm detects points on or close to sharp edges and corners using the deviation of normal vectors of the neighboring 3D points. The second step consists in the simplification process. It is achieved by progressively removing points in smooth regions. The quantification of a point's importance is based on points in its neighborhood and corresponds to the point's contribution to the representation of local surface geometry. The algorithm calculates the average of the Euclidian distances from the considered point to the estimated tangent plane represented by its neighbors. The idea is to evaluate whether or not the local surface geometry of this point can be reliably implied from its neighboring points. A big value indicates that the position of this point cannot be reliably represented by its neighboring points. Therefore, it plays an important role in the definition of the local geometry. A small value indicates that even without this point the local form derived from its neighbors would not change much; therefore, it is not so important and is likely to be removed. The problems that arise here are how to select neighboring point, and how many points should be selected. Both steps of this algorithm require establishing neighborhoods for each data point and estimate its normal vector. This is very costly in computing time. Other flaw is that at each iteration, one point is deleted. Furthermore, for each point removed, the algorithm recalculates the importance value of all points in the vicinity. Finally, due to the exigency of maintaining all the edges and corner points, the number of points in the simplified point cloud cannot be less than the total number of these points, this may cause problems for objects that have many textures.

In 2012, Li et al. [Li 2012] introduced a decimation algorithm with feature preservation. This algorithm extracts the feature points because they have a high importance since they describe the basic shape of the object. These points are positioned at convex or concave boundaries and sharp regions. In order to keep these points, the authors establish a curvature threshold defined by the sum of the distances from the k nearest neighbors of each point to the tangent plane at each point. For the rest of the points, the algorithm covers them by spheres with radius adapted to the density of the neighborhood. For each sphere, an optimized point is determined. It replaces all the points of the sphere.

As the new 3D measurement tools provide highly accurate data and can capture any surface variation, any reduction can lose textural details. Despite all the research presented above, it remains an interesting question about what could be an appropriate RGB-D reduction (based position and color) and how to use these information to reduce the size of raw clouds. Li and his colleagues in their paper [Li 2012] report that to ensure the quality of simplification the geometrical characteristics must be taken into account. We propose here to add the color feature in the reduction criterion so that the cloud retains the full of its original information. Our second contribution lies in the ability of our algorithm to process point clouds that can go up to several tens of millions of points. All these previously presented research test their algorithms on small clouds of points that do not exceed one million points.

Major reduction rates can be achieved with our method, which can exceed the rate of 90 % of reduction while preserving the amount of information carried by the original cloud, which constitutes our latest contribution.

7.3 The proposed approach

The information carried in the 3D point contains the position information, color and reflectance. In this approach, we show that reducing the huge amount of raw data points can be done without losing the essential information and its initial accuracy. The proposed approach is based on two different RGB-D cloud features: color and 3D location. Figure 7.2 illustrates the work flow of the proposed RGB-D down-sampling method that consists of three sub-tasks: (1) voxelization, (2) clustering and (3) subsampling. The first task performs a spatial grouping which attempts to preserve the topological information based on the 3D position of the points. A set of 3D cubic regions (voxels) is generated where all points within the voxel have very close spatial positions. The second task bundles all points of each voxel based on color. Once this grouping is done, we perform the last task which keeps one point for each color (cluster) in the voxel. This point is selected in function of the cluster centroid. This allows to have a much smaller representation while preserving spatial and color information of the original cloud. A detailed description of these three tasks is presented next.

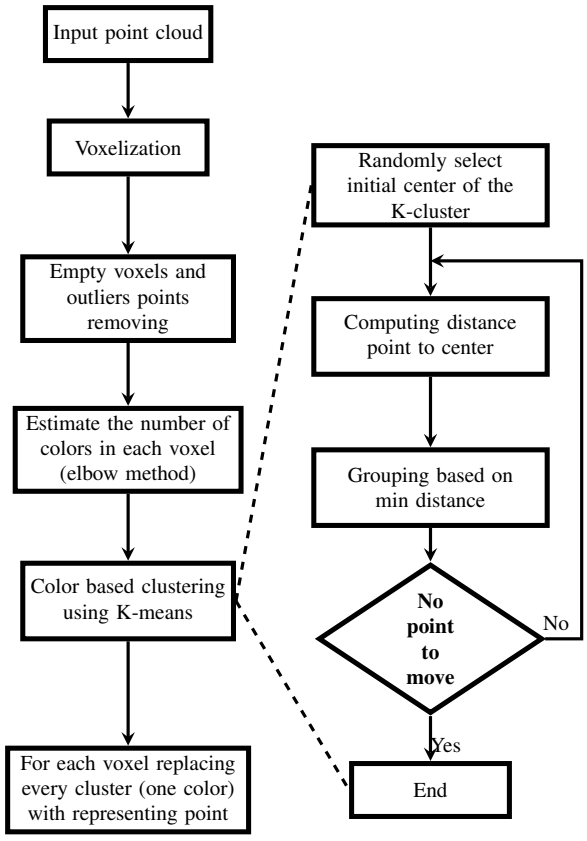


Figure 7.2: *The flowchart of the approach.*

7.3.1 Voxelization

At the beginning, the procedure applies a bounding box to the entire point cloud by finding the minimum and maximum positions of points along the three axes X, Y and Z. The number of voxels for this bounding box is determined by the voxel size. Figure 7.3 shows the output of this processing.

7.3.1.1 Voxel assignment

Each voxel is identified by a unique linear index. If i, j, k represent the voxel indices in the X, Y, Z dimensions respectively, numDivX, numDivY are the number of voxels along x and y axis, the formula to encode the linear index taken from [Igelbrink 2015] is:

$$idx = i + j * numDivX + k * numDivX * numDivY \tag{7.1}$$

According to this equation we assign an index idx to each point. This relationship allows direct access to the desired voxel, avoiding to browse the set of all voxels [Wiemann 2014].

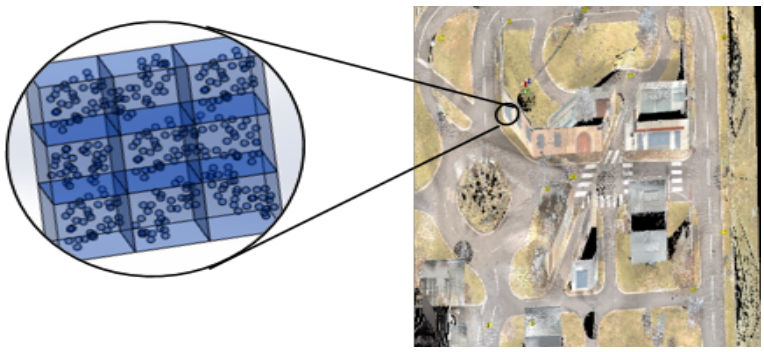


Figure 7.3: *Voxelisation of the Pavin map. The topological features are retained by chosing small voxel size.*

7.3.1.2 Voxels suppression

As the shape of the point cloud is arbitrary (see Fig. 7.3), the step of delimiting points by bounding box creates many empty voxels. All empty voxels will be removed and only voxels which contain points will remain. This step will also allow removing noise and outliers which are easily detected at this stage: all voxels containing a number of points below a threshold and having empty adjacent voxels are considered as voxels containing noise or outliers, and therefore, are deleted.

7.3.2 Clustering

Clustering can be defined as the task of automatically identifying groups of similar characteristics from a given set of data points [Kaufman 1990]. In our case, this process is performed on points of each voxel in order to group those that have the same colors. This step is carried out using one of the most popular clustering techniques,

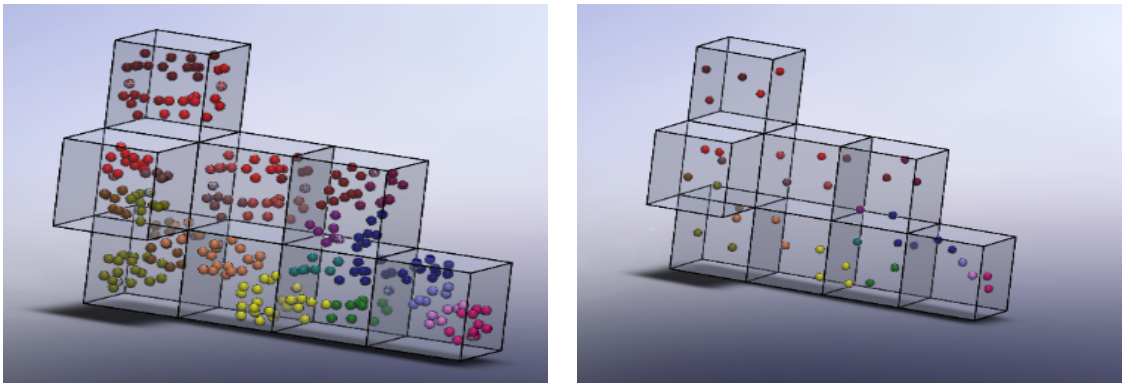
which is the k-means algorithm [Kaufman 1990, Arthur 2007]. We have detailed the fundamental principle of this method in Chapter 3.5.1.

7.3.3 Subsampling

When the clustering method is applied on points within each voxel, it generates several groups of color, according to the color variation in this voxel. The points in each group are considered unnecessary points insofar as they belong to the same voxel and have the same color. Therefore, keeping a single representative point for each class is sufficient. In our case, we keep the centroid of each color class. Figure 7.4 illustrates this process. This process of removing unnecessary points is called subsampling.

7.4 Results

In this section, we evaluate our method with a series of experiments. The results presented were obtained using a C++ implementation of the method under Linux Ubuntu 14.04 with an Intel Core i7–4800MQ, 2.7 GHz with 32 GB of RAM. To the best of our knowledge, no other state-of-the-art method considers the color information in the reduction process. This is why it is practically impossible to evaluate and compare in some ways, fairly, our method with existing works. However, we provide a comparison with a known sampling method (PCL voxel grid) [Rusu 2011] which proceed only on 3D point clouds.

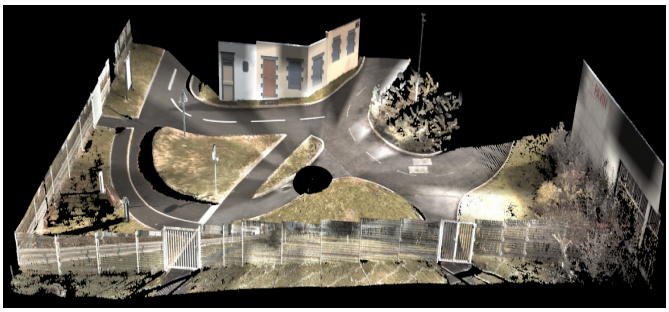


(a) Point cloud before reduction (b) Point cloud after reduction

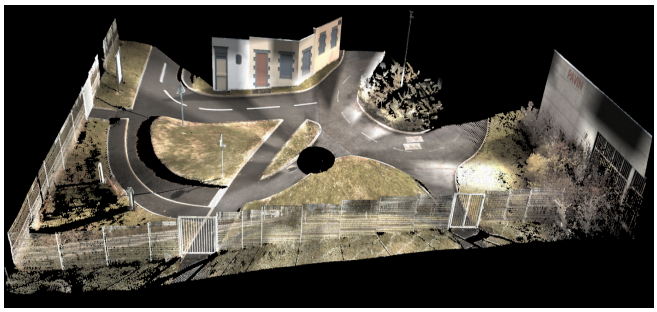
Figure 7.4: *Color-based point cloud reduction. The point cloud is voxelized and in each voxel clustering of points is performed based on their colors. Each color's clusters' centriod is then chosen as the representative point, removing the remaining points.*

During the implementation of the method, we tried to be as simple as possible while maintaining a high efficiency. This is the reason for the existence of a single parameter that adjusts all the algorithm. This parameter represents the voxel size. When increasing the size of the voxels, we increase the number of points enclosed within this voxel, which leads to a larger reduction rate. The efficiency is highlighted by a large rate of point reduction while preserving the maximum of original information. Figure 7.5 shows four point clouds used for the evaluation of the method. Because of the huge amount of points collected over the entire map (almost 1 trillion points),

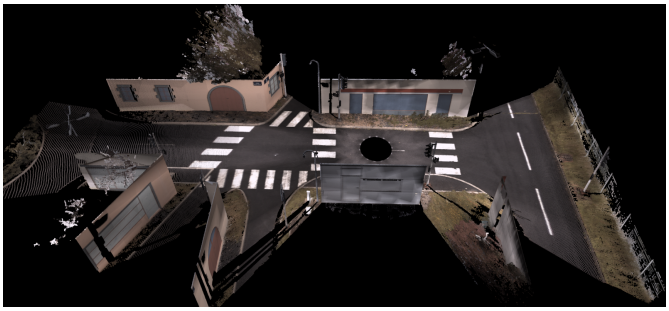
Original clouds Simplified clouds (after reduction)



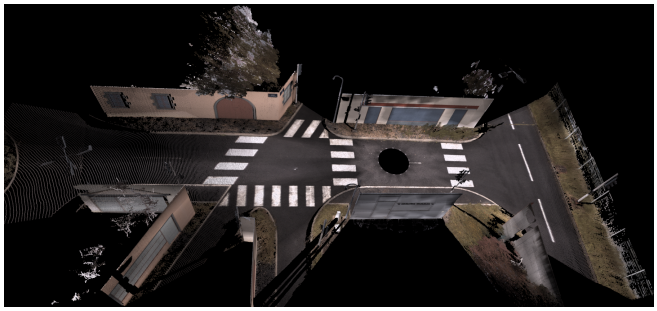
(a) Pos 1: 12 443 799 pts



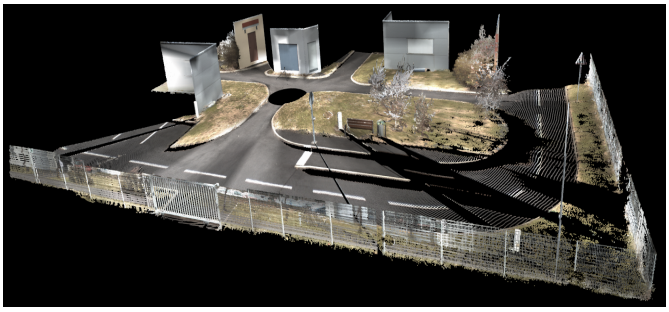
(b) remaining points: 1 239 171 pts – reduction rate: 90 %



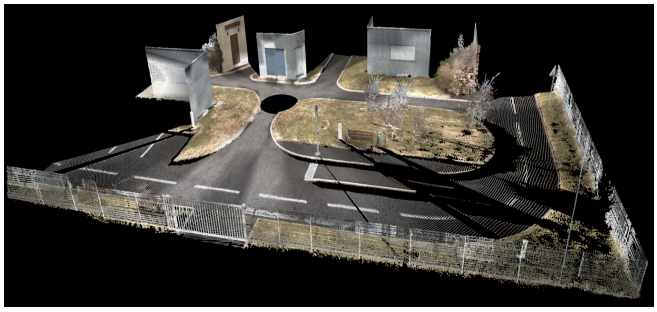
(c) Pos 2: 19 828 910 pts



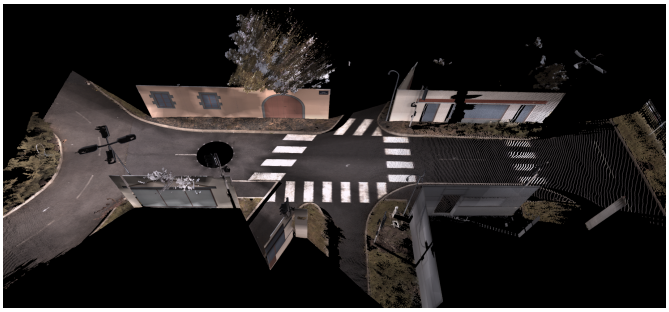
(d) remaining points: 1 450 549 pts – reduction rate: 93 %



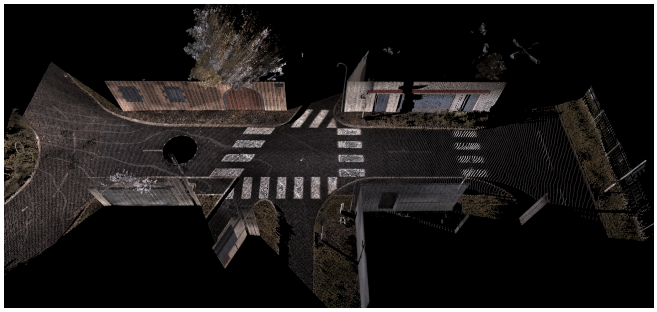
(e) Pos 3: 13 714 237 pts



(f) remaining points: 662 343 pts – reduction rate: 95 %



(g) Pos 4: 19 615 433 pts



(h) remaining points: 392 174 pts – reduction rate: 98 %

Figure 7.5: *Four examples of colored point clouds before and after reduction.*

making it very complicated to process, we limited our evaluation to clouds having between 10 and 80 million points.

Table 7.1: *Simplification results on Pavin map datasets.*

Samples	Original clouds	Simplified clouds				Processing time			
	Number of points	Number of points	Simplification rate	Number of voxels	Average of points/voxels	1 (s)	2 (s)	3 (s)	Total (s)
Pos 1	12 443 799 pts	1 239 171 pts	90 %	617 683	20 pts	0.769	90.998	0.013	91.78
Pos 2	19 882 910 pts	1 450 549 pts	93 %	724 441	27 pts	1.223	92.425	0.006	93.654
Pos 3	13 714 237 pts	662 343 pts	95 %	291 095	47 pts	0.837	144.502	0.021	145.36
Pos 4	19 615 433 pts	392 174 pts	98 %	152 979	128 pts	1.206	198.268	0.303	199.777

Table 7.1 presents the results of the reduction step. For each cloud, we indicate the original number of points and the total number of voxels. The reduction rate (ratio between the size of the original cloud and the size of the cloud after reduction) is then presented. Computation time of reduction process is also provided. The columns refer to the time taken by the steps that follow: (1: voxelization) (2: determining the number of colors in each voxel) (3: grouping with k-means using the right number of clusters). As expected, the step for detecting the number of colors in each voxel is the most costly part. This step, which is directly proportional to the number of points in each voxel, will benefit from parallelized implementation in coming versions of the algorithm. The computation time of this step may seem high, but it should be noted that for each n points of each voxel, the algorithm runs $n/2$ times the clustering process with k-means, and each time it calculates the cost function (distance between each point and its center).

Figure 7.6 illustrates how the voxel size impacts the computation time to detect the required cluster number for the same point cloud of 72 947 044 points. The corresponding clouds to this transition are given in Fig. 7.7.

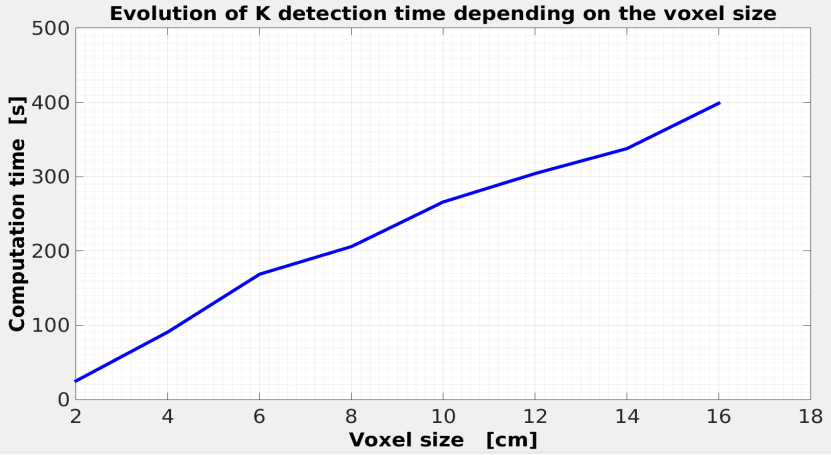
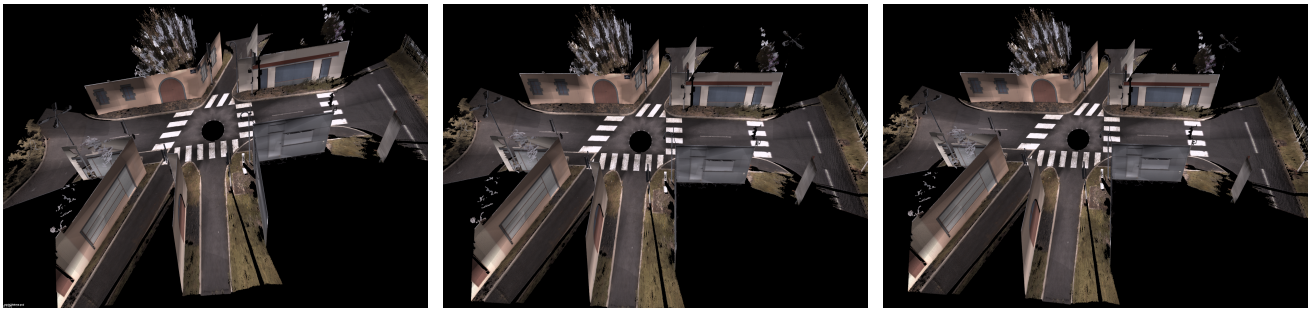
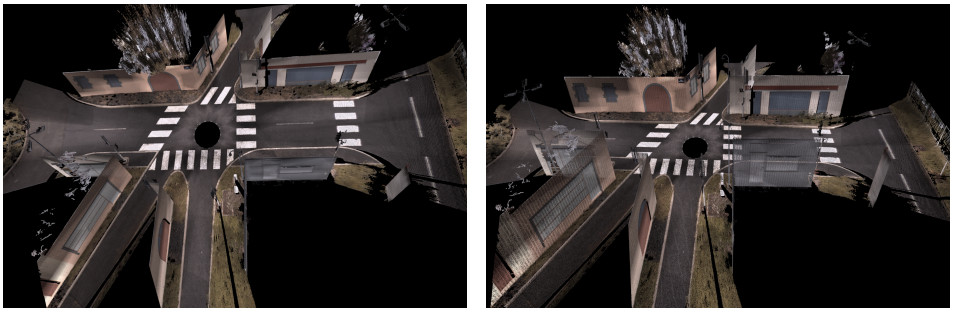


Figure 7.6: *Evolution of the processing time according the voxel size.*

Evaluation of the results of a reduction process is considered as a challenge for the scientific community, as Zhang et al. assert in their paper [Zhang 2012] published in 2012. Indeed, almost the majority of research articles does not provide metrics to evaluate their results. They are based on the visual aspect to assess it. In order to make a comparison with the PCL method, we propose a quantification of the number of colors in the two resulting clouds after reduction. These two results are compared with the number of colors in the original cloud. For this, three voxelizations are done: one for each cloud with the same voxel size.



(a) Original:72 947 044 (b) 6 847 434 pts 90 % (c) 5 025 457 pts 93 %



(d) 2 732 783 pts 96 % (e) 1 712 950 pts 98 %

Figure 7.7: *Evolution of reduction process according to the change of voxel size for a single point cloud [Original size:* 72 947 044 *points].*

A color clustering process in each voxel is made. The result of this comparison, of 100 voxels randomly taken, is shown in Fig. 7.8. This experiment was taken from [Tazir 2016]. It can be seen that our method (green) performs far better than PCL method (red), while the original one being represented as blue. The main difference

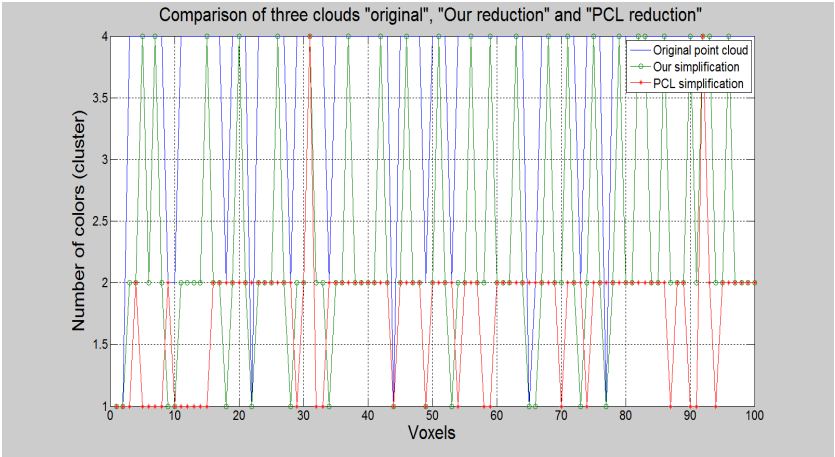


Figure 7.8: *Number of detected colors according the methods used. The original cloud is included for comparison.*

between our method and the PCL method lies in calculating the color of the representative point. The PCL method replaces all points within a voxel by their centroid. The color of the resulting point will be calculated from the average color of all points in that voxel. Therefore, PCL creates new points that were not in the original cloud.

In our case, however, the exact features of the original point cloud are maintained. This is done by keeping a single representative point of each color for each small voxel. The choice of smaller voxels is intended to limit the reduction rate and to preserve the topological details of the cloud.

7.5 Conclusion

In our localization approach, a map obtained with a dense and precise digitization is used. However, this map quickly becomes difficult to manipulate because of its huge amount of data, which can reach billions of points. A phase of data reduction is inevitable in order to exploit the richness of all the information carried by this map. To the best of our knowledge, the literature review has showed that no reliable method existed yet for the reduction of colored point clouds. Most of the methods only consider the geometric constraint in the reduction process, or create some new vertices which replace the huge entry points, dismissing by that a large proportion of information carried by the input samples. Our goal was to introduce color features into the sub-sampling process. The approach presented in this chapter aims to find sets of 3D points with the same color in very small region and replacing each set with one point. The volume of the map will be significantly reduced, while the properties of this map such as the shape and color of scanned objects remain preserved.

With this sampled map, the vehicle starts to move while receiving on-line data. The localization is then ensured by the matching between these data and the prior map.

Chapter VIII

Localization within a Prior Map

CHAPTER 8

Localization within a Prior Map

Contents

Now that we have laid the basics of localization techniques, detailed different used methods and means, create the reference map, we will address in this last chapter the main contribution of my thesis: a 6 DoFs LiDAR-based localization in a priori map. This localization is mainly performed by matching a 3D prior map with incoming point cloud structures as the vehicle moves. Several sensors can be used for this purpose. In our case, we make use of a high precision panoramic LIDAR together with Velodyne technologies to meet our ends of precision. This result in two different data structures; the former output very dense data at high precision, high resolution whilst the latter gives sparse data at low resolution at the expense of lower computational effort. We use an approach based on the extension of the CICP concept to deal with that, which proves to be an efficient viable solution to this map matching problem.

8.1 Introduction

Just as a human being, who is able to use a map, to explore it, and to combine it with visual inputs to locate itself efficiently, the vehicle uses prior made maps that combines with local data of its onboard sensors to find its position in the global map. The key to ending at this stage is the matching between the online data and the map.

Many companies had started to get interested in mapping services. Ranging from giants of technology, such as Google, Apple, and Intel to major automakers such as Volkswagen, Ford, BMW, and Mercedes. This recent attention has given rise to various precise 3D maps allowing the robot to navigate safely. These maps, which covers many places in the world, are carried out with different sensors. However, in order to use these maps, the robot must contain exteroceptive sensors compatible with those used in the development of the prior map. Indeed, The localization using maps is achieved by place recognition process, which is defined by the ability to automatically detect the robot position within a map in the absence of specialized infrastructure including GPS [Bosse 2013]. This is done by matching the collected sensor data (3D points) and the part of the map corresponding to the same location. Up to now, we find in the literature only matching (alignment) algorithms, which deal with point clouds coming from the same sensor. Nevertheless, in the exemption of the embedded sensors compatible with those used in the elaboration of the prior map, it is not possible to benefit from all the produced maps with the maximum of the desired precision.

In this chapter, we will be interested in this issue, and we will apply our method of sparse-to-dense registration proposed in Chapter 5 to perform the matching between the online sparse data and the dense map-data. This method allows to surpass the notion of density and consequently used maps of different origins with data from different sensors. Moreover, it increases the precision as well as the computation time since matching is performed on a reduced set of points.

8.2 Similar works based on LiDAR

Map-matching is the concept referring to the fusion of sensor measurements with map information in order to improve localization [Quddus 2007, Romero 2018]. Apart from the classical methods [White 2000, Taylor 2001, Ochieng 2003, Wolfson 2004] that seek to improve the precision of the GPS or dead reckoning position by comparing them to a digital map, or visual localization that use maps acquired with cameras, we focused on LiDAR techniques to directly benefit from 3D data and overcome the problem of photometric appearance. Consequently, localization can be done at a different timescale compared to the mapping, which requires that the process of localization should be robust to the environment change (such as the lighting change).

The first state-of-the-art methods [Levinson 2007, Levinson 2010a] relie on the reflectance, which is the proportion of light reflected by the surface of a material. This allows capturing lane markings, pavement variation, tar strips, etc. In these works, others extract the ground points from the laser data and build 2D maps of ground-points intensities. The localization is performed using the online 3D LiDAR scans and IMU. Baldwin and Newman [Baldwin 2012] develop a similar approach by using a 2D LIDAR scanner to create 3D swathes which can be matched statistically within the 3D survey. However, reflectivity-based methods alone may fail when the road appearance degrades over time or is obstructed by harsh weather conditions, such as rain puddles and snow that can accumulate and obstruct the view of Informative lane marking.

More modern efforts include probabilistic localization using multiresolution Gaussian mixture maps. Wolcott and his colleagues [Wolcott 2015, Wolcott 2017] propose a method to localize a self-driving vehicle in an urban environment. They model the world as a mixture of several Gaussians characterizing the distribution over the z-height and the reflectivity of points. The localization is achieved by registering the online point cloud data against these maps by formulating the scan matching problem as a branch-and-bound search. This approach performs matching in 2D space and consequently, it provides only 3-DoF poses. The same authors propose in [Wolcott 2014] a camera localization in 3D LiDAR map using a geometric matching in order to avoid the problems of environmental conditions and photometric appearance change. Although this approach also provides only 3-DoF poses, the method presented by [Caselitz 2016] estimates the complete 6 DoF poses. The latter localizes and tracks a monocular camera in a 3D LiDAR map. It reconstructs a sparse set of 3D points from its input images via bundle adjustment. Then, the reconstructed points are aligned with the 3D LiDAR map points by using ORB-SLAM [Mur-Artal 2015]. However, both approaches required expensive image rendering supported by GPU hardware.

Other researchers [Maddern 2015] have imagined another concept based on the fact that each vehicle independently creates and manages its own map. When the vehicle travels through a route several times, it creates the map of this environment, this map is improved as the vehicle pass by this route. Once enough experiments are accumulated, the vehicle can drive in autonomous mode on certain sections of this frequently-traveled route. For their tests, they use a vehicle equipped only with 2D LiDAR.

In this work, we propose an approach that relies on geometric information only. By directly matching the 3D geometry of sparse online data and dense precise map in order to achieve accuracy. This approach enables us to overcome two major shortcomings: the problem of correspondences in different point cloud densities, noise inherent in sensors leading to noisy normals. In doing so, an improvement on the convergence domain between the frames of the online received data and the reference map tethered to two dissimilar depth sensors is considerably improved leading to robust localization. Moreover, our approach increases the precision as well as the robustness of the alignment since matching is performed on a reduced set of points.

8.3 Localization process

The localization task is performed by matching between a 3D prior map and point clouds received as the vehicle moves, as Figure 8.1 shows. To do so, two distinct phases are deployed. The first allows the construction of the map, with centimeter accuracy using static or dynamic laser survey technique, as we have seen in the previous chapter. The second corresponds to the capacity to localize the vehicle within a 3D map in the absence of dedicated infrastructure including GPS, Inertial Measurement Unit (IMU) or beacons. The proposed localization can use data from any sensor providing cloud points, namely LiDAR, RGB-D/ToF cameras, and stereo vision systems. The following subsections explain in detail the process used, this latter is schematized in Figure 8.2.

8.3.1 Prior map

The reference map can be loaded dynamically via a ROS topic as 6 DoF $sensor\_msgs :: PointCloud2$. While the online data are logged at a refresh rate of 10 Hz. Each point cloud contains approximately 70,000 points.

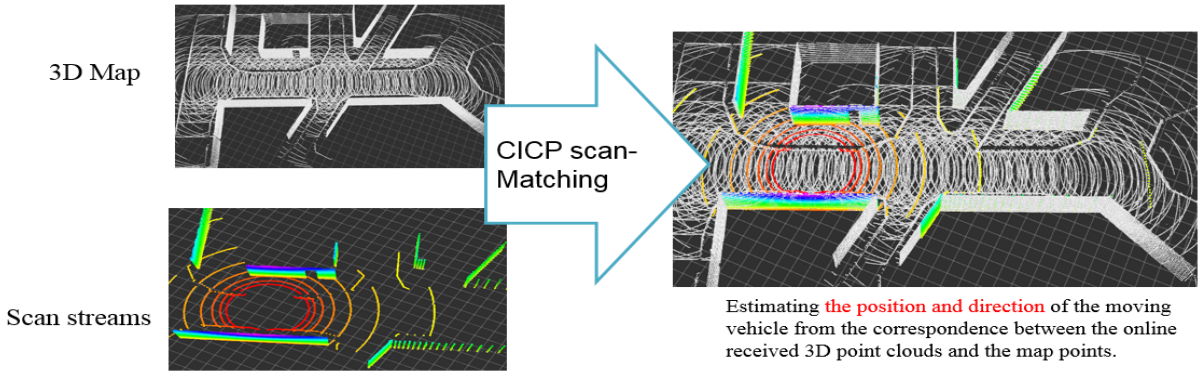


Figure 8.1: *3D prior map based localization.*

8.3.2 Downsampling

The amount of time required to match the online scans and the reference map increases dramatically when the number of points of both becomes larger. For this reason, down-sampling is performed for both the prior map and the online scan stream. For the RGBD map, an offline sampling process is carried out using our down-sampling method [Tazir 2016] detailed in the previous chapter. This is in order to reduce the number of points to the order of few millions without losing useful information. Regarding the online scans stream, they are also sampled using the voxel-grid method [1]. The sampling level is set according to the desired localization accuracy and the available computing resources.

8.3.3 Filtering

The online-received data are filtered based on a distance criterion, in order to use data only within the recommended distance interval of LiDAR measurements. Moreover, since the location placement of the Velodyne sensor did not have any specific study, it may be useful to remove the points too close to the sensor, because they can belong to the vehicle itself and not to the environment. This can be achieved by applying a minimum and maximum threshold to the distances returned by the LiDAR sensor.

8.3.4 CICP based map-matching

The map-matching process intends to localize the vehicle by corresponding the online scan data and the map. This is done using our CICP method presented in Chapter 5, which is a sparse-to-dense registration technique. CICP takes as input point clouds of different resolution, gathered by different sensors, or with the same sensor. It is based only on the geometric information of points, which makes it independent of weather and illumination conditions. The proposed approach aims to cluster points of the same surface as one topological pattern and replace all the points held by this model by one representing point for the matching step. Technically, CICP starts with the estimation of normals of the map and current point clouds using Principal Component Analysis (PCA) [Jolliffe 1986]. Thereafter, the map cloud is subdivided into small voxels. Points belonging to each voxel are subjected to a classification process based on their normals, giving rise to different groups of points, according

[1] Source: http://pointclouds.org/documentation/tutorials/voxel_grid.php

to the geometric variation of each voxel. Each group of points represents a local surface since they share the same normal vector. A single point is chosen from a local surface extract to be used for the matching process. The closest point to the centroid of each local surface is elected a winner. Similarly, the current scan point is first transformed into the reference frame of the map cloud using a pose estimate before undergoing a similar process; voxelization, normals-based classification, designation of point's representatives. At the end of these steps, the method results into two improved sets of points from the corresponding clouds. Each set contains the most probable points to match with the points of the second set (this is more particularly in the overlapping area of the two clouds, as it reflects the same geometry seen from two different viewpoints).

The main highlight of this method is the way 3D surfaces are segmented in a point cloud representation using voxelization and clustering approaches. The advantages are multifold: both sparse and dense point clouds are subsampled by maintaining the geometry of the surface. Moreover, better obtained point estimates are used for later stages of matching and registration. In doing so, the convergence domain of the cost function is greatly improved leading to a faster convergence of the algorithm where, classical techniques (ICP, GICP, NDT) fails. Additionally, the method of normal based segmentation not only improves on the weakness of the heterogeneous problem of sparse to dense registration but also deals with sensor noise leading to noisy normal extraction. For further details, please refer to Chapter 5.5.1.1.

8.3.5 Localization validation

Once the online scan is matched to the map cloud by the CICP technique, two metrics are computed in order to evaluate whether this pose is valid or not. The first one is the root mean squared error (RMSE) and the second one is the percentage of matching points. The typical thresholds of these two metrics according to [Costa 2016] are less than 3 cm for the RMSE metric and at least 35 $\%$ of matching points. If these two metrics are within these two thresholds, the map-matching task is considered successful, and the vehicle pose can be computed.

If the first metric exceeds the desired threshold, the system will ignore this pose estimation and attempt to estimate it in the next sensor data updates. After a certain number of matching attempts between the online scans and the map, the system activates a localization by LiDAR odometry (6.2.2) from the scans stream of the LiDAR sensor.

8.3.6 Map Update

The map update is linked to the second metric, by adding live sensor data to the map only when the percentage of matching points between the received point cloud and the equivalent part of the map drops below the predefined threshold. The built-in point cloud is transformed using the found pose in the map reference frame before being integrated into the map cloud.

8.3.7 Algorithm

The initial position of the vehicle is initialized using a GPS position. When the first online scan is received, the algorithm performs a correspondence search for the best possible matching around the initial starting position. This best matching then becomes the new position for the next matching search. The vehicle is then tracked by taking

the last estimated pose and performing a local search around it. The complete algorithm is shown graphically in Figure 8.2.

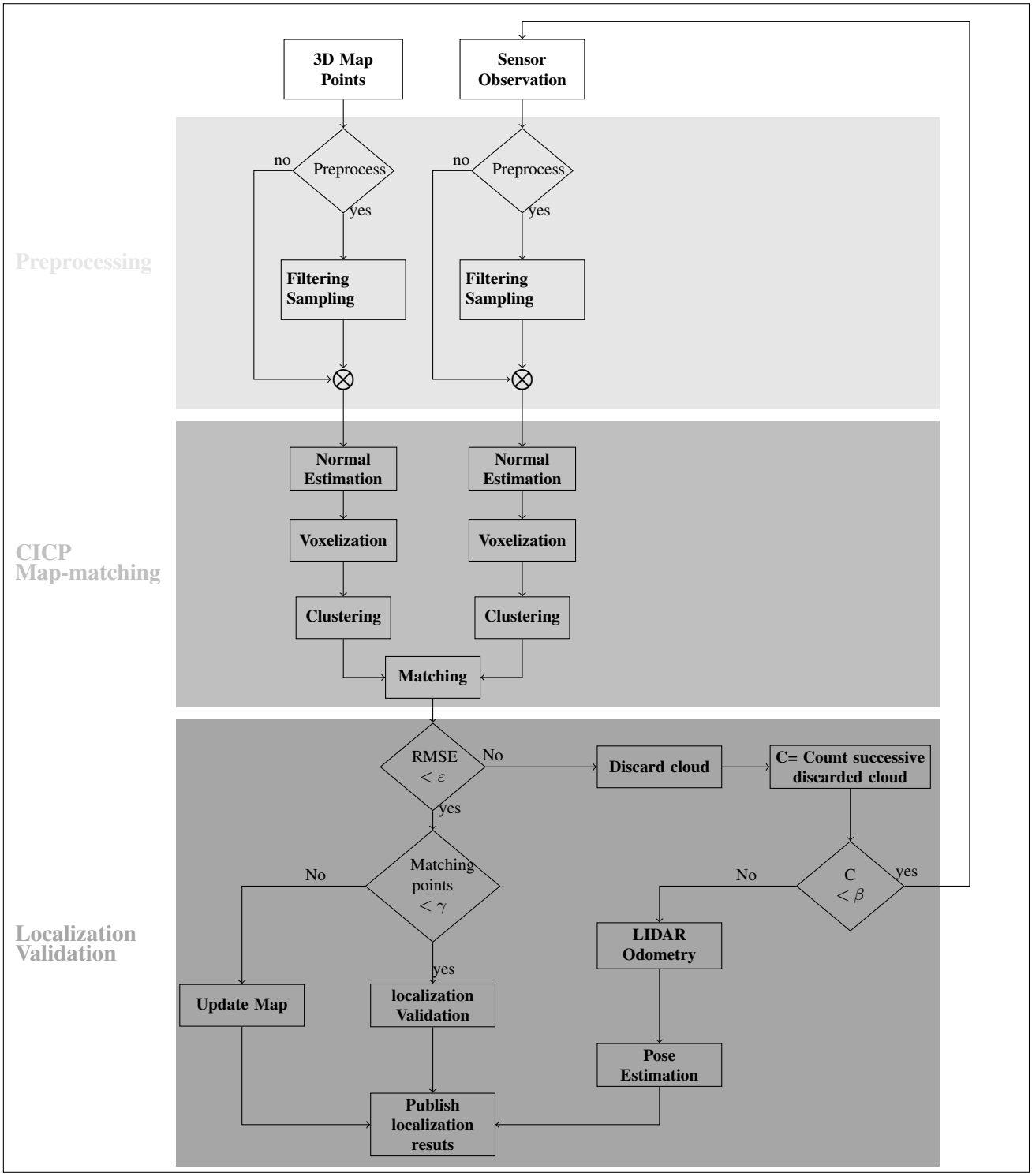


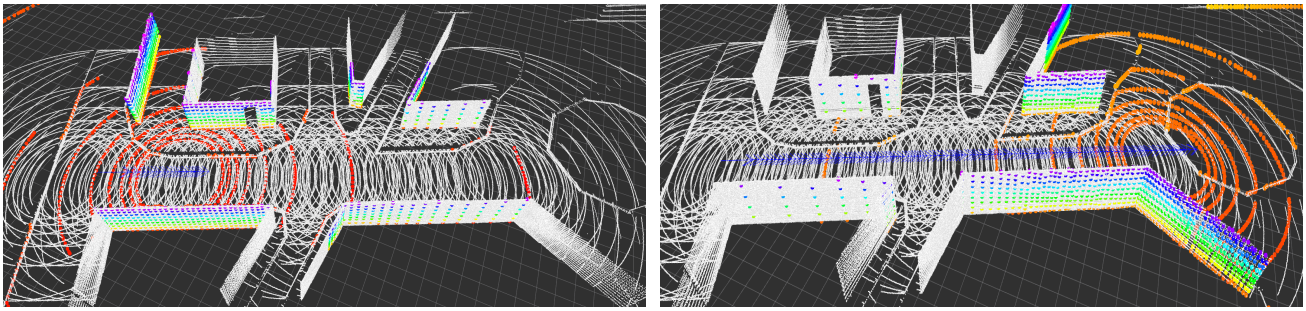
Figure 8.2: *The flow chart of the overall map-based localization process.*

8.4 Experimental results

In this section, we will examine the localization performance in simulation and real-world environment. For the simulation, we use the tools presented in the previous chapter (model of PAVIN and Vipa in Gazebo, RVIZ). Regarding the real-world environment, the experimental tests consist of 3 data acquisitions companies on PAVIN between January and May 2018. All the tests were performed using a laptop equipped with a Core i7 processor running at 2.7 GHz, and a RAM of 32 GB. It has to be pointed out over here that the initial mapping phase has been conducted using LiDAR odometry (Velodyne) and the subsequent localization phase is conducted using CICP. The main motivation for that is that CICP is very computation intensive and at the time of writing this manuscript, the optimization of the code has been left out.

8.4.1 Results on simulation

To test the localization process by map-matching, we generated with the vehicle a trajectory consisting of almost 200 LiDAR poses (bleu arrows in Figure 8.3) on the simulated model of the Pavin platform in gazebo. Its length is of 30 meters. The map used in this experimentation is obtained by a first pass of the robot through the environment. Then, the subsequent passes would be used to test the localization with respect to this built map. The obtained results are shown in Figure 8.5.



(a) (b)

Figure 8.3: *Simulation results.*

8.4.1.1 Localization accuracy

A precise localization is essential for any autonomous system to operate successfully in a fully autonomous manner. Figure 8.5 shows the comparison between the pose estimation of the LiDAR odometry without using map (continuous green line) and the pose estimation by map-matching (blue dashed line), the ground truth is given in red line. This ground truth is extracted via the ROS topic gazebo_$msgs$ :: $ModelState$ as 6 DoF poses. This figure presents the localization accuracy in terms of x, y, and z position relative to the ground truth, as well as the RMS error and the relative translational error (RTE) and the relative rotational error (RRE). Despite a little drift of 5 cm on the z position, we have a good horizontal alignment (x and y positions). The results shown over here is made up of 200 frames with an accumulated distance of 30 meters. According to our data analysis, we notice that

the velodyne data present certain associated issues. For example, the ground points do not superimpose even if the two scans are close, which trumps the convergence of typical ICP approach to accurate registration. Moreover, the sparsity of the data (gaps between the rings) causes also a lack of spatial correspondences between scans. This is shown in Figure 8.4. These issues have been raised by [Velas 2016].

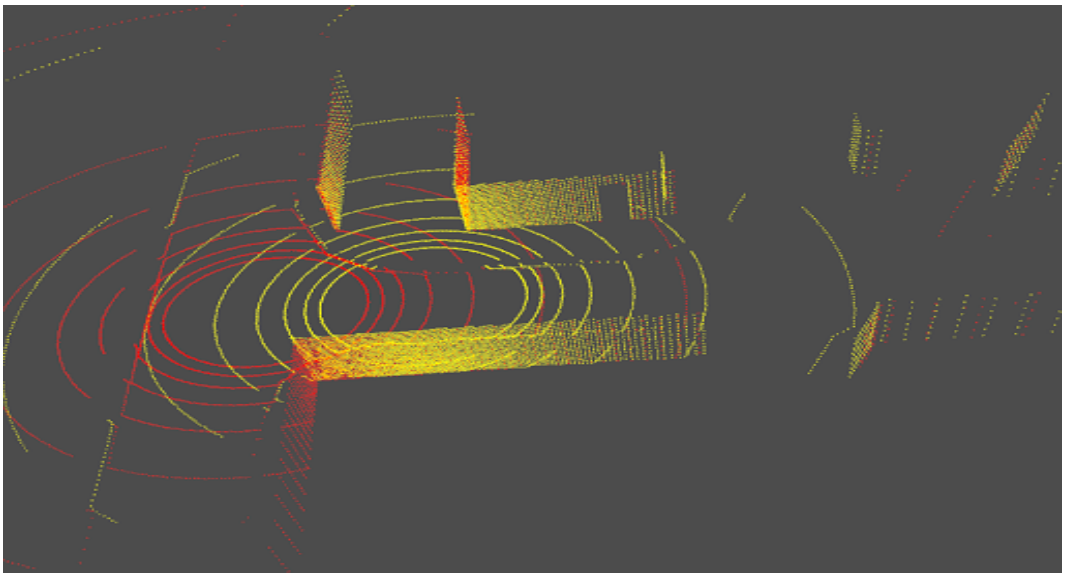


Figure 8.4: *The problems associated with the point clouds captured by the Velodyne LiDAR scanner. The ground points do not superimpose even if the two scans are close, which trumps the convergence of the algorithm.*

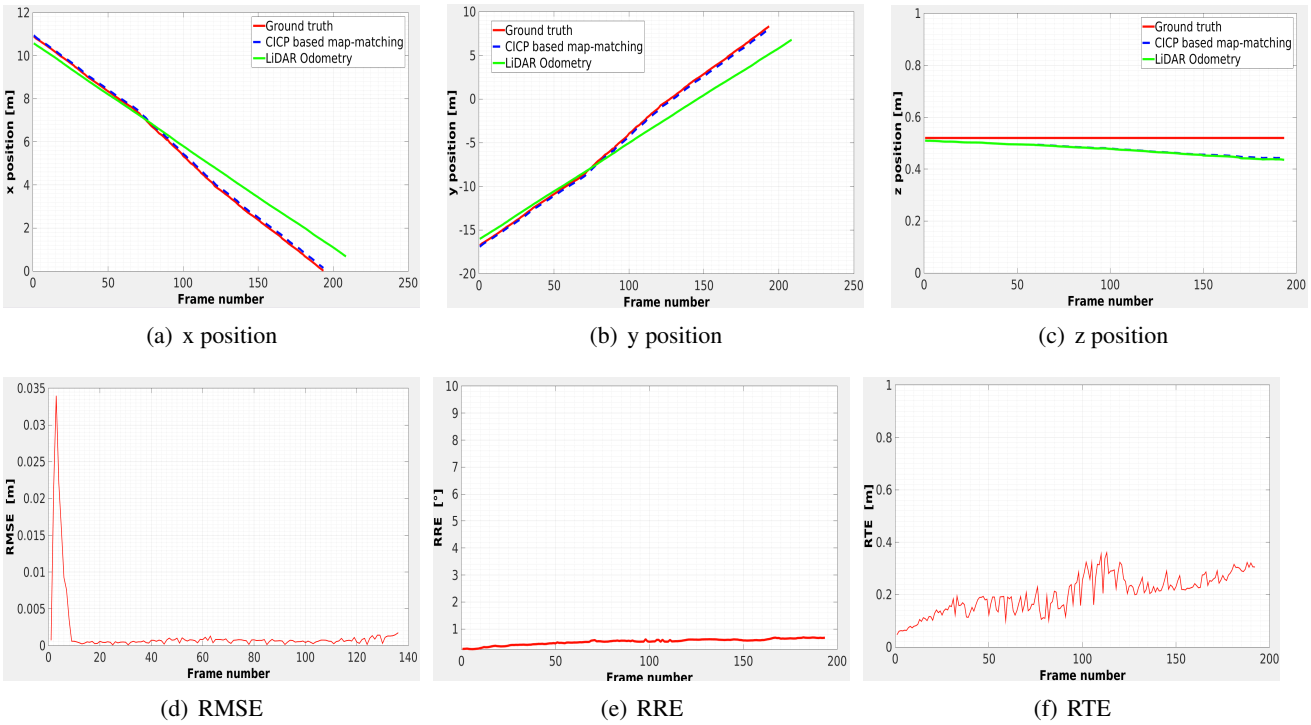
Therefore, we suspect that the drift obtained from the Z-axis may be coming from there. For the RMS error, RTE, RRE, we are able to achieve an average of 0.0024 meters, 0.1822 meters, and 0.4947 degrees. The spike exhibited in figure 8.5(d) corresponds to an initialization phase in order to localize the agent within the map. This may take several frames before an accurate position is obtained within the map, causing the estimated pose to stabilize thereafter along the trajectory. These results show that the performance achieved by the map-matching based localization is greater than that without using a map for both horizontal and vertical positions.

8.4.2 Results on real data

Regarding the real tests, we build the maps of the environment statically using the Leica P20 LiDAR, or dynamically by making a pass through the test environment (PAVIN site). We then, perform passages in the same test environment (on different days) while performing the localization against the built maps.

8.4.2.1 With statically constructed maps

These maps are built in a static way using the Leica P20 TLS, as we have detailed in Chapter 6. In order to be able to use the map for the localization, it should be expressed in a common global frame of reference, which will also be inferred by the vehicle. We have two methods to do this: the first is to geo-referenced during the time of



(a) x position (b) y position (c) z position

(d) RMSE (e) RRE (f) RTE

Figure 8.5: *Simulation localization study in the Pavin platform in Gazebo.*

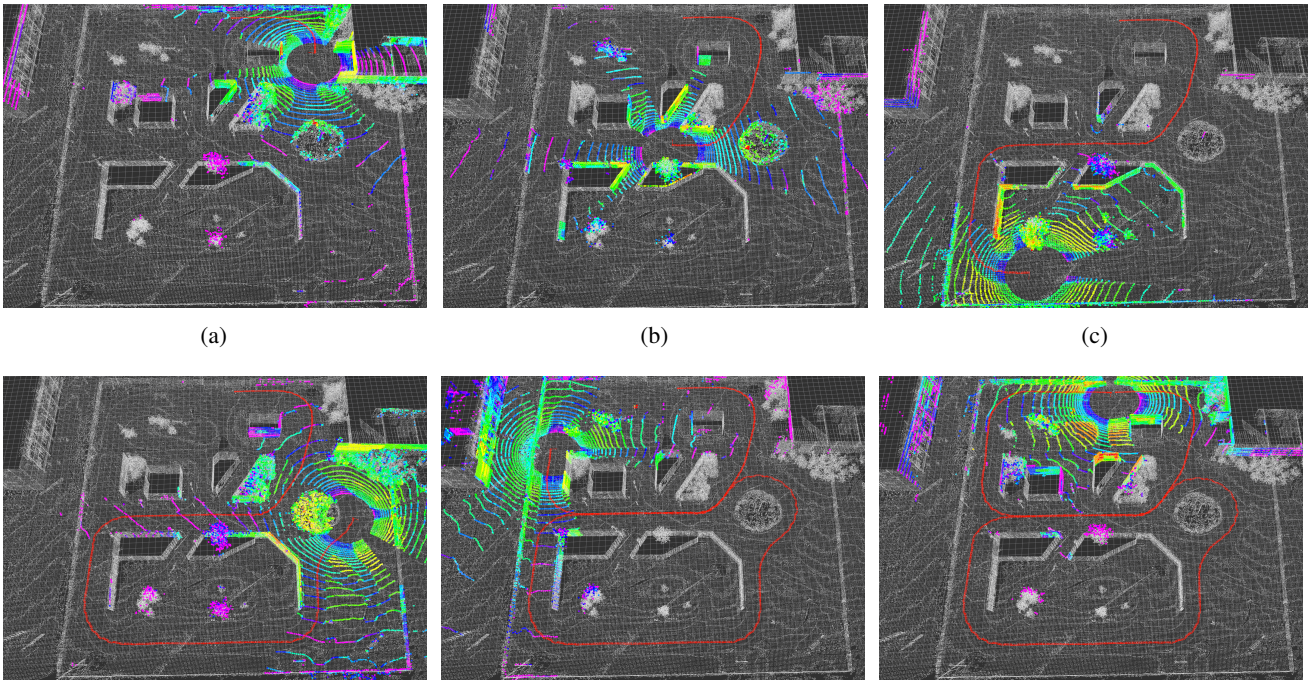
acquisition process the scans acquired in a GPS coordinates frame. The standard process consists of tying scan data to the appropriate coordinate systems. The second is to transform the map cloud (s) in a frame of a map already expressed in the global coordinate system to which the robot is attached. For this latter method, a simple registration technique is necessary to find the transformation between the two maps.

Qualitative evaluations

Figure 8.6 shows the qualitative evaluation results of our CICP map-matching algorithm using the Leica P20 built map. This figure shows some samples of the localization process using this kind of map. The vehicle loads this map dynamically via a ROS topic as $sensor\_msgs :: PointCloud2$. It starts to move while receiving the online data in real time. These data are presented in color according to the intensity of the reflectance. The localization is then ensured by the map-matching of the online data and the map. The trajectory of the vehicle is shown in red, which consists of almost 2000 poses for this experiment.

Localization accuracy

Figure 8.7 shows the comparison between a frame to keyframe RMSE error obtained from classical incremental LiDAR odometry and the RMSE error obtained from a frame to map localization algorithm using our CICP. The drift is evident when no map is being made. This is obvious on the RMS error curve in Figure 8.7(a), where it



(a) (b) (c)

(d) (e) (f)

Figure 8.6: *Sample results of our CICP-based map-matching technique using Leica P20 map.*

is shown that at the end of the sequence, drift accumulation becomes important. On the other hand, when an a priori map is used the drift is greatly reduced and the error remains constant throughout the whole sequence. On this experiment, the average error is about 5 millimeters. The error draws to this value after a few frames at the beginning, which is necessary to initialize the localization phase.

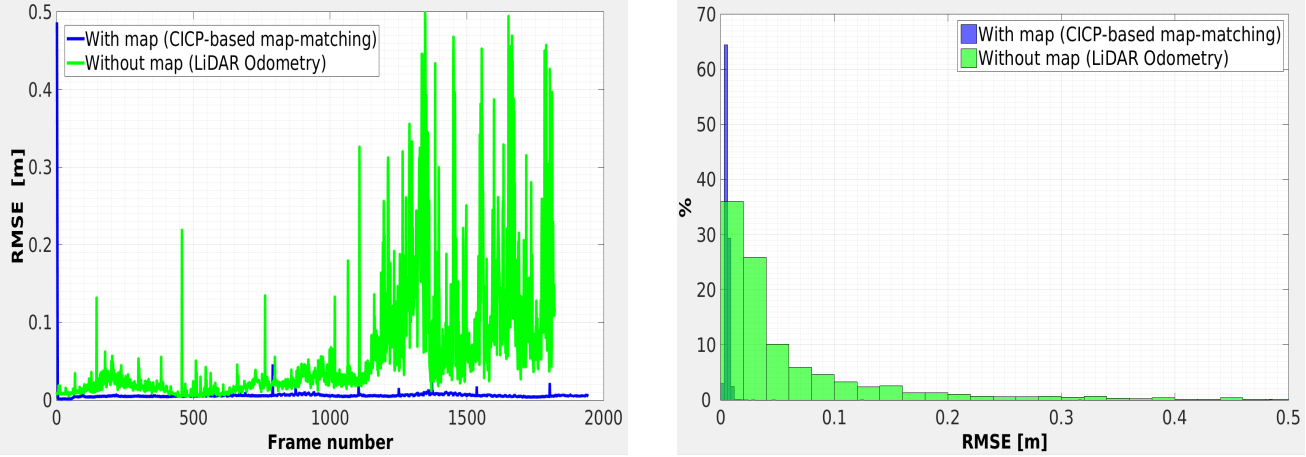
Another criterion that testifies the precision of the CICP map-matching localization is the histogram of the errors, shown in Figure 8.7(b). According to this histogram, 98 % of errors are less than 5 millimeters.

In addition, the trajectory formed from the estimated poses is illustrated in Figure 8.8. This trajectory is able to close the loop correctly, contrary to the trajectory estimated by the LiDAR Odometry (i.e. without a map) shown in the previous chapter Figure 6.17(b) where there is a slight deviation between the final and the initial position.

However, without ground truth, it is difficult to make any quantitative conclusions about the performance achieved. But qualitatively, we notice the optimal performance of the algorithm with negligible trajectory drift, and that too without loop closing. The matching error is of the order of millimeters.

8.4.2.2 With dynamically constructed map

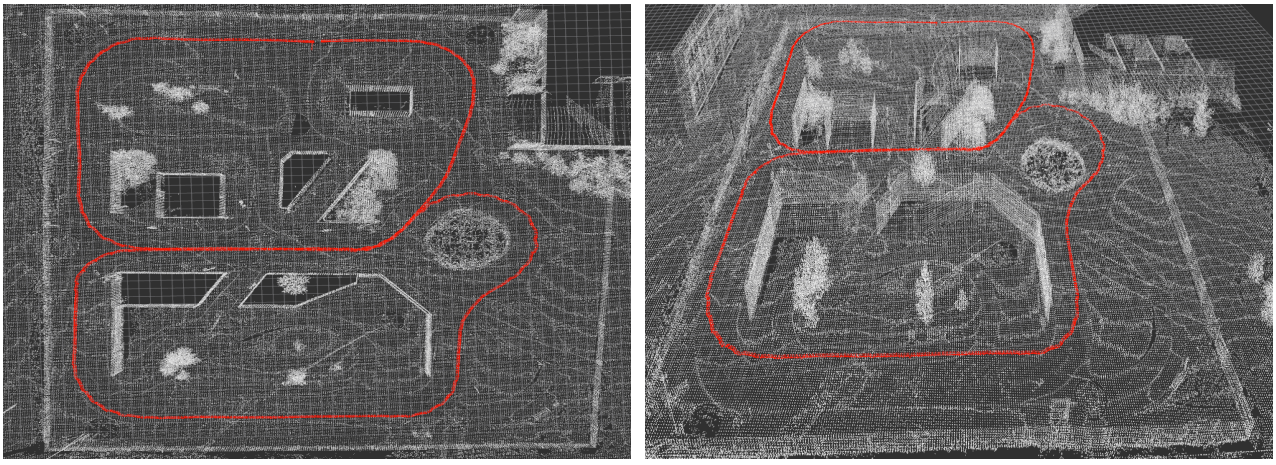
In reality, these kinds of maps are constructed dynamically with specialized vehicles equipped with specific sensors. The goal is always to have the most possibly accurate map, since all the errors contained in the map will be reflected in the localization process. The advantage of these types of maps, they are built in a practical and fast way. It consists only of a tour through the environment and the map is built simultaneously. In our thesis,



(a) RMS error

(b) Histogram error: 98 % of RMS error were under 5 mm.

Figure 8.7: *Comparison of the RMS Error of Pavin sequence using Leica P20 map, blue: our method based on CICP map-matching, green: LiDAR odometry.*



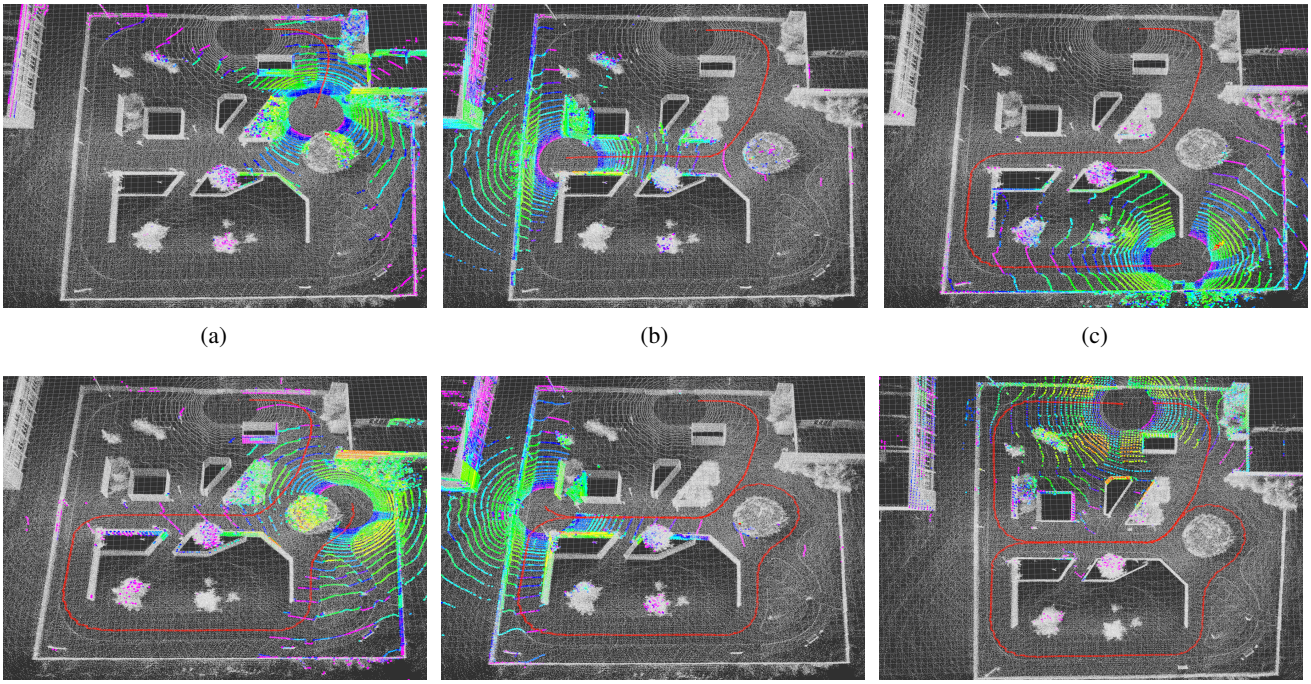
(a) (b)

Figure 8.8: *Trajectory computed from CICP map-matching poses. Left: top view, right: horizontal view.*

given the unavailability of such precise sensor (MMS), our primary objective was to validate the method using a dynamically built map from the Velodyne HDL 32 data, even if the built maps are less accurate.

Qualitative evaluations

As the same as the previous experiment, Figure 8.9 shows some samples of the localization process using the velodyne map. The online data are presented in color according to the intensity of the reflectance. The obtained trajectory of the vehicle is shown in red, which is stored dynamically in a $geometry\_msgs :: PoseArray$ ROS topic.



(a) (b) (c)

(d) (e) (f)

Figure 8.9: *Sample results of our CICP-based map-matching technique using LiDAR odometry built map.*

Localization accuracy

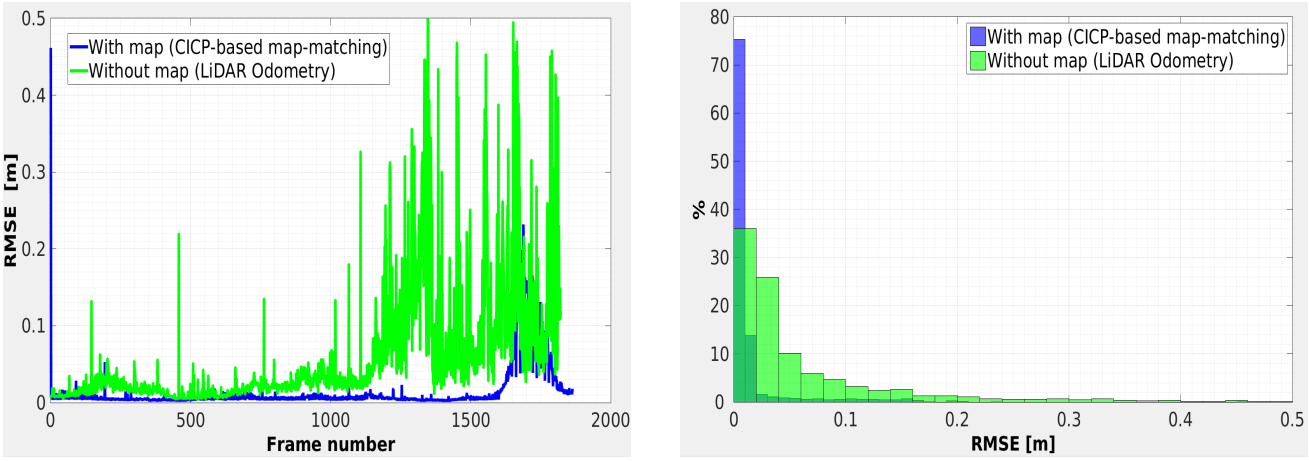
Figure 8.10 shows the comparison between a frame to keyframe RMSE error obtained from classical incremental LiDAR odometry and the RMSE error obtained from a frame to map localization algorithm using our CICP.

Similar to the previous experience, a comparison between a frame to keyframe RMSE error obtained from classical incremental LiDAR odometry the RMSE error obtained from a frame to map localization algorithm using our CICP is performed. The results are presented in Figure 8.10(a). The average error is in the order of 16 cm. This value is larger than the previous experience. This is mainly due to the nature of the used map. The maps constructed with the LiDAR Odometry present an average error of 20 cm, whereas statically constructed maps have an error of about 1 mm (value indicated by the Cyclone software). The RMSE curve shows significant errors towards the end of the sequence. This is due to the inaccuracy of the map at this location (the drift becomes important in such a way that it influences the localization process).

The histogram of the errors shown in Figure 8.10(b) shows that 90 % of the errors are less than 2 cm.

8.4.3 Towards lifelong mapping

In this section, we give an insight of the importance of having a compact, efficient and re-usable mapping system for autonomous navigation. We cannot only rely on a single turn built environment representation as the latter changes, with time, over seasons and years due to ongoing natural and man-made occurrences. Therefore, updating the map



(a) RMS error

(b) Histogram error: 90 % of RMS error were under 2 cm.

Figure 8.10: *Comparison of the RMS Error of Pavin sequence using LiDAR odometry built map, blue: our method based on CICP map-matching, green: LiDAR odometry.*

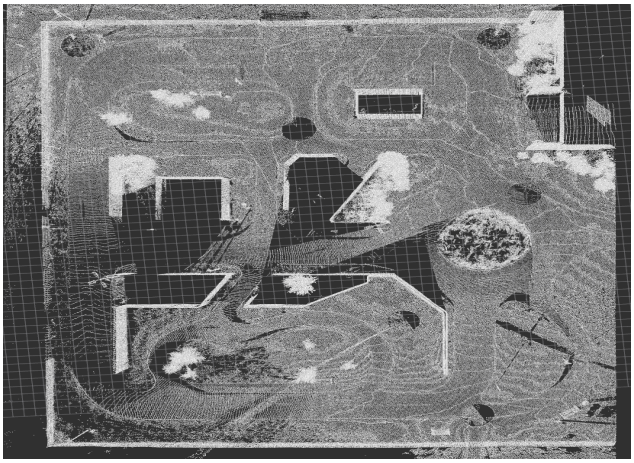
so that it serves its true purpose over the longevity is of utmost importance. Here two scenarios are envisaged:

8.4.3.1 Scenario 1: lack of information in a map portion

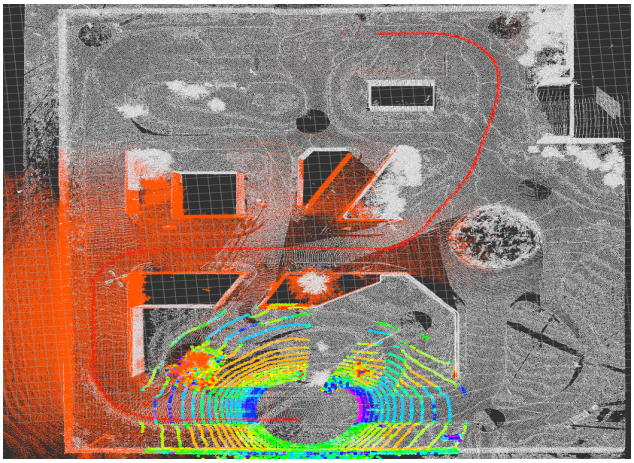
This can be useful in the case where a portion of the map has a failure or lack of information. In this case, the vehicle uses its live point clouds to localize and correct or/and update the map. Figure 8.11 shows this, where it shown the use of a map containing non-mapped passages. The vehicle is able to locate correctly in these places and update the map (the orange parts).

8.4.3.2 Scenario 2: completion of the mapped area

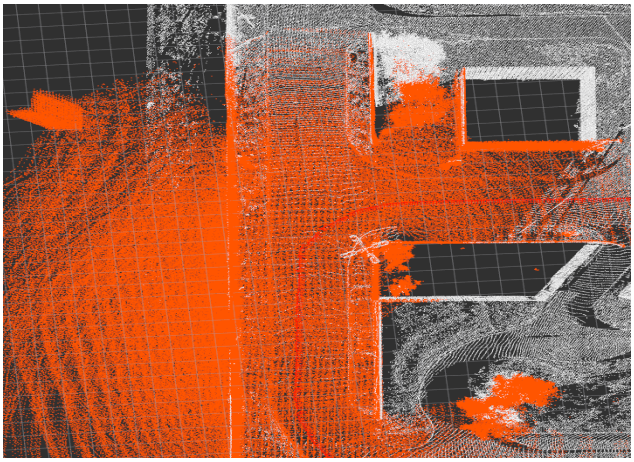
The navigable surface by the autonomous cars is immense, and until today, the mapping part is minimal. The question that arises here is: "does a self-driving car that is based on the use of the prior information to localize must be limited only to the mapping area?". The answer of course is "No". In the case of an autonomous car uses a priori map for localization and arriving at the end of this mapping part, the car continues to localize itself using the stream of scans received from its Velodyne sensor. This amounts to using an odometry localization process that we have detailed in Chapter 6. In parallel with this localization process, the vehicle updates the map.



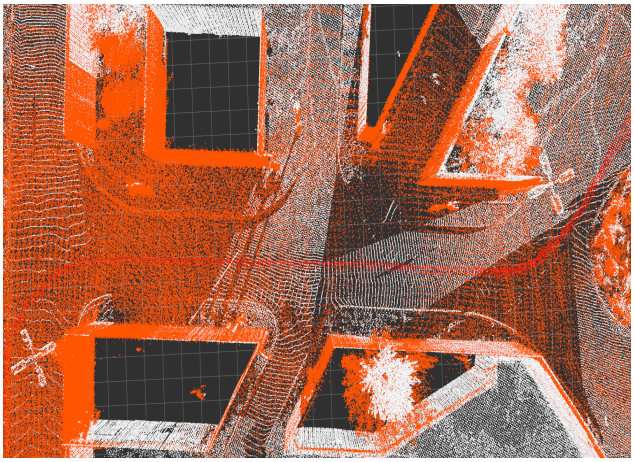
(a) Original map



(b) Update map

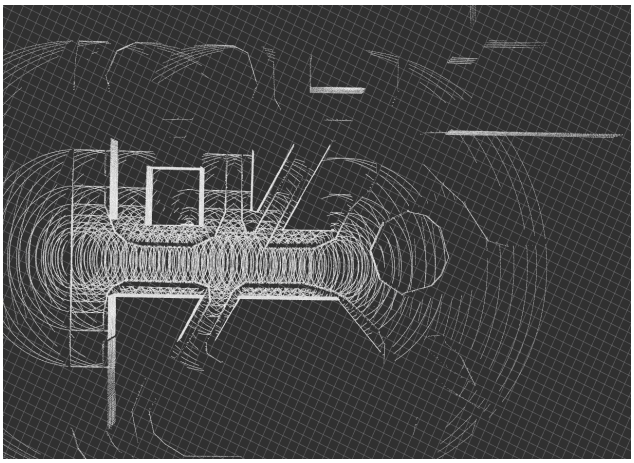


(c) Exploded view of the map updated portion

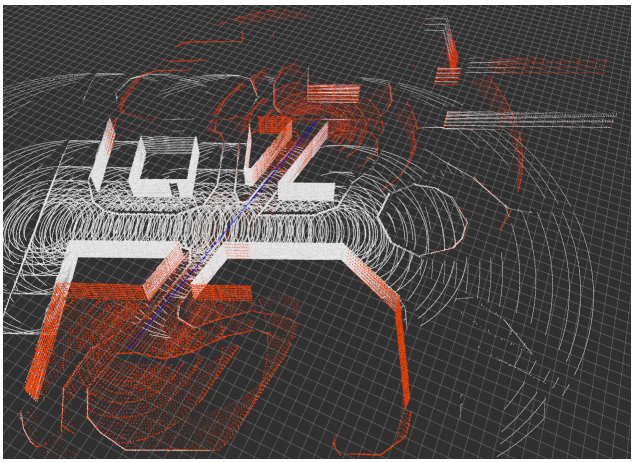


(d) Exploded view of the map updated portion

Figure 8.11: *Map update: lack of information in map.*



(a)



(b)

Figure 8.12: *Map update: completion of the mapped area.*

8.5 Conclusion

In this chapter, we propose a 6 DoFs LiDAR-based localization whose approach extends of the concept of CICP sparse-to-dense alignment detailed in chapter 5. The method proposed over here belongs to map-matching techniques. It handles localization using maps of different densities and online data, tethered to two dissimilar depth sensors.

The results obtained confirm that localization using a prior accurate map (Leica P20) can significantly improve the accuracy and robustness of localization process. Furthermore, it can reduce /complements several steps in the SLAM backend pipeline such as loop closure and global optimization. Here we do not mean that it replaces these steps, but the fact of using a priori map gives, as we saw, better results than the overall SLAM process including the outlined steps.

The lack of ground truth however, has been a minor shortcoming, which would have otherwise given us even more dimensions to explore in our comparative study.

CHAPTER 9

Conclusion & perspectives

9.1 Conclusion

At the heart of the scientific and technological actuality, the autonomous driving is for many industrialists a key technology that will undoubtedly determine their future competitiveness. Some authors go so far as to qualify this period of the golden age of the autonomous car by talking about a new technological revolution. The scientific community has also taking up this theme very seriously. Hence, many projects and scientific works in different disciplines englobing the field of autonomous vehicles.

This thesis focuses on tools to determine the precise localization of an autonomous vehicle in an *a priori* known environment. This localization is determined by map-matching between a 3D prior map and the data acquired as the vehicle moves.

However, a fully autonomous car requires a planning of its route according to its location and the surrounding environment. Planning can be seen as a process that starts with memorizing the vehicle environment. This memorization is followed by getting informed about its current position and then defining the strategy to reach its destination. In other words, three phases are needed in order to carry the whole process: mapping, localization and path-planning. **Mapping** is the phase that aims to represent the spatial structure of the environment, to update a pre-built map and/or to correct an incomplete or erroneous map. **Localization** is intended to determine the position of the vehicle on this map corresponding to its exact position in the real world. Once the location of the vehicle in the map is established, the **path-planning** phase allows the vehicle to anticipate the movements to be carried out to reach its goal. Our research work is particularly interested in the intermediate phase, which is localization, with a significant attention to mapping techniques in the context of mobile robotics, and more especially in the specific area of the autonomous vehicles.

In particular, the thesis subject includes several research components that are: the map construction, its treatment in order to make it usable for the localization phase, and finally the stricto sensu localization of the vehicle on the built map.

We have presented this thesis work in three major parts: the first deals with the theoretical and bibliographical aspects of the mapping and the localization fields. The second part is devoted to the point cloud registration techniques, and the last one proposes to work on the map construction techniques and the use of these maps for localization. These three parts are preceded by an introduction that exposes both the usefulness and the current character of this research subject, while justifying our choice on the use of 3D prior maps for the localization of the self-driving car.

The theoretical part (Chapter 2) first gives a brief review of principal sensors used in autonomous driving localization. Then a detailed review of the mapping strategies is presented, followed by the localization approaches. This chapter allows us to propose an original classification of the localization methods, by grouping them into four categories, according to the surrounding environment and the robot's knowledge of this environment. The underlying objective of this classification is to better understand the localization problem as a whole. In the third chapter, basic concepts such as the registration techniques, matching, and clustering are introduced to the reader in order to provide a strong prerequisite in the understanding of this work.

The second part, consists of two Chapters (4 and 5), is devoted to the registration technique. Chapter 4 presents a bibliographical and experimental study of the Iterative Closest Point (ICP) method. This chapter aims to understand ICP that is considered to be one of the key components of mapping and robot localization. Our primary concern was the adaptation of this method to our thesis work, which is based on 3D laser localization. This resulted in the implementation of more than 200 variants of ICP. These variants are tested with several dozen of point clouds from different environments. In the following chapter, we conducted a thorough study of the techniques of scan-matching, and more specifically on the "dense to sparse" or "sparse to dense" matching, since our localization strategy is done by matching between sparse data and a dense map. This resulted in a proposal for a new method of sparse-dense registration, exploiting normals differently, by clustering points of the same surface into one topological pattern and replacing all the points held by this model by one representative point. These particular points are then used for the association step of registration instead of directly injecting all the points with their extracted normals. In our work, normals are only used to distinguish different local surfaces and are ignored for later stages of point cloud alignment. This approach enables us to overcome two major shortcomings; the problem of correspondences in different point cloud densities, noise inherent in sensors leading to noisy normals. In so doing, improvement on the convergence domain between two reference frames tethered to two dissimilar depth sensors is considerably improved leading to robust localization. Moreover, our approach increases the precision as well as the computation time of the alignment since matching is performed on a reduced set of points. Through a series of experiments, we show the effectiveness of the proposed approach in terms of alignment accuracy. A comparison with the techniques presented in the literature shows a real added value of our approach for the type of sparse-dense (dense-sparse) registration.

The last part focuses on both the mapping and the localization used technique. We started with the maps building (Chapter 6). The latter are elaborated by two different techniques: a static technique that provides dense and metrically precise maps. To do this, we use a dense and precise LiDAR sensor, the Leica P20 and its software packages. However, this first method is relatively slow and requires human intervention, at least for the use of the instrument. Given the target areas and their number for autonomous navigation, it is not conceivable to use this scanning method. It is therefore obvious that it is necessary to use a dynamic method with incorporated LiDARs for the map establishment. The latter is our second method of maps construction. However, in the absence of a mobile laser scanner (MLS) such as Leica-Pegasus 10 or RIEGL VUX-1HA 11, etc., which deliver accurate data, we have validated the used approach with a Velodyne HDL-32. The adopted approach uses an odometry-free mapping and localization technique that is based only on LiDAR data. To generate the map, a keyframe strategy was adopted. First, the keyframes were generated based on a distance threshold, and used to generate the global map. This technique allows both to reduce the error drift and to prevent the creation of keyframes (adding new data to the map) if the vehicle remains in the same position. This reduces the information contained in the built-in map by not overloading it with unnecessary information. Therefore, this criterion gives rise to the update of the

map with a new scan if and only if it contains new usable information. Due to the reliability of the keyframes update, our technique did not perform loop closure. As a result, it accumulates errors during the mapping. If the distance of the map is short, the errors are minimal. Nevertheless, the larger the size of the map, the more the error in the 3D map can also be important, and consequently cannot be ignored. Finally, we have performed multiple experiments to evaluate the proposed LiDAR odometry method. The tests have been conducted in both indoor and outdoor environments as well as on the KITTI odometry datasets.

In the seventh chapter, we propose a method of point clouds sampling. Indeed, the used maps quickly becomes difficult to handle because of its huge amount of data, which can reach billions of points in the case of maps created by the Leica P20. However, the interest of such a quantity of information is not always justified. Thus, a phase of points cloud reduction is inevitable in order to exploit the richness of all the information carried by these maps. The proposed approach is based on the use of color information and the geometry of the scene. First, a voxelisation is performed to preserve the topological details of the scene, and then, for each voxel, a color-based classification of the points is performed. Then one point of each color class is preserved and the other points are deleted. The volume of the maps will be greatly reduced, while the properties of these maps such as the shape and color of scanned objects remain preserved.

Now that we have laid the basics of localization techniques, detailed different used methods and means, create the reference map, we will address in this last chapter the main contribution of my thesis: a 6 DoFs LiDAR-based localization in a priori map. The method used is based on the extension of the sparse-dense registration technique proposed in Chapter 5. It is able to localize the vehicle precisely using maps of different densities and online data, tethered to two dissimilar depth sensors. Different tests are used to demonstrate the reliability of the method and its high accuracy.

9.2 Implementation and results

9.2.1 Results of perception and map-based localization

The results presented show that the use of prior maps for localization of autonomous vehicles improves the accuracy of the localization, which determines the relevance and accuracy of the following steps (path planning, obstacle avoidance, etc.). These results also prove the usefulness of analyzing the nature of the data of different sensors and take into account in order to conceive techniques usable with a variety of sensors and scenarios. These results have been demonstrated in simulation and using real data. We have performed several experimentations on our test vehicle.

9.2.2 Implementation improvements

An important part of this research work was the implementation of different tested approaches on C++. The computational efficiency of the different implemented algorithms was beyond the scope in this thesis. We are rather focused on the methodology. An important work which remains at hand is the parallelization of different implementations and more importantly the CICP on several CPU cores or using a GPU to make it more efficient,

depending on the needs and objectives laid out so that the algorithm may run for real-time applications on the fleet of vehicles available at Pascal Institute. This indeed requires a whole set of pure software skills and management, which needs to be undertaken but still envisaged by involving the technical team at Pascal Institute.

9.3 Possible ways of improvement

9.3.1 Method validation

9.3.1.1 Validation with data from different sensors

The use of 3D maps and online data acquired from different sensors to further test the robustness of the algorithm, such as using a set of 3D points extracted from images features and matching them against the Leica P20 map. In this context, it would have been interesting to integrate the works of Eric Royer [Royer 2007] which constitute of building maps with monocular vision only. The output is highly optimized sparse feature based point clouds.

9.3.1.2 Expansion of testing campaign

Expansion of test campaigns on dynamic environments and other types of environment (urban, rural, semi-rural, etc.). These tests offer the opportunity to validate the approach in real conditions and to improve its efficiency by learning from errors and encountered situations.

9.3.1.3 Comparison with other methods

In this thesis, we have only compared our work only with readily available integrated algorithms either in PCL or in relying on our own implementation. The major reason behind is the stability of the PCL library throughout the years, which has itself, become a benchmark for comparison. On the other hand, mostly other minor available implementations online come with risks of compilation and run time bugs, which might take us in a spiral loop of endless, debugging, bearing in mind the poor recognition of software hassle towards our final objectives. Having said that, it would have been interesting to compare CICP with other recent trustworthy libraries such as NICP [Serafin 2015] or Open3D [Zhou 2018] which have also recently improved the state of the art on 3D registration.

9.3.2 Map-based localization

9.3.2.1 Reference maps

- In this work, we have not been able to experiment with classical optimization libraries such as CERES and G2O. It would be interesting to apply a pose graph optimization library to further rectify the poses. However, any noticeable improvement remains to be seen given that the trajectories computed in this work is very close to optimal even without any loop closure as shown by the results.

- The methodology proposed in this thesis is purely metric based localization, which is normally located at the base or lowest layer of any mapping technology. Basically, mapping can be segmented into various levels, in order of increasing information abstraction; metric, topological and semantic. Therefore, the approach proposed can very objectively fit in a wider mapping system to provide complementary information about the level of localization required (whether metric, topological and semantic).

9.3.2.2 Map-matching

- Most vehicles adapted for autonomous navigation are equipped with an extensive range of sensors such as IMUs, GPS, cameras, encoders or Velodynes. In this work, we have made use of Velodyne only localization. The fleet of vehicles available at Institute Pascal, such as the Vipa models or even the EZ-10 present possibilities to explore the data fusion techniques in order to augment the online data with other dimensions, since the maps produced with the Leica P20 have other dimensions such as color, or intensity. For example, information available from the above-mentioned sensors, though they present various sources of uncertainties of all kinds can be integrated into a single framework for accurate and precise map-matching estimation. Over here, a paradigm of fusion techniques emerges out with deals exclusively with filtering techniques (Kalman and variants, Particle, etc...) which is again complimentary to the optimization framework used in this thesis.

- Incorporating the concept of measurement uncertainty into the alignment model serves both to improve our CICP map-matching and to probabilistically apprehend a fusion step in our map creation pipeline. We believe that the fusion of points taking into account the different sources of uncertainties (the uncertainty of the points, the uncertainty of normals, and the uncertainty of the poses) improves the accuracy of the points and therefore the localization of the vehicle in the map. In this context, the Joint compatibility Branch and Bound (JCBB) fits our pipeline pretty much by including a probabilistic formulation to data correspondences. These techniques will definitely improve the positions of the 3D points of the same map by fusing theme together in order to better align the incoming scans during the map-building process. This will help to reduce the noise at the point's level and to have a more precise map for the localization.

- With the recent development of the deep learning methods, it seems that the use of this new paradigm in the field of perception and localization is promising, as long as there are sufficient training databases. In this area, the force of the major technological companies such as Google, Apple, and Facebook is undeniable. This has already led to the emergence of innovative solutions such as LocNet [Yin 2018], VoxelNet [Zhou 2017], PointNet [Qi 2016], etc.

9.3.2.3 Dynamic Map update

An important stage would be to consider object-level description in order to detect the mobile and dynamic object (parked car for example) to remove them in the update stage of the map.

9.3.3 Public dataset for map-matching localization

During the course of this work, we were led to manipulate a whole range of software (ROS, Cyclone, Cmake)/hardware tools, test vehicles and sensors. In particular, the manipulation of the two major "protagonistic" sensors: Velodyne HDL–32 and the Leica P20 were very much handful which required tens of acquisition campaigns. At the end of the day, we believe that the maturity and the expertise gained can be worthwhile by putting our efforts towards building a full-fledged benchmark data for 3D map-matching localization and make it available to the research community for evaluation. This again opens up endless possibilities where people experimenting with the dataset will come up with even more robust localization techniques at the benefit of the community thereby pushing forward the state of the art.

APPENDIX A

Publications

A.1 Journal Publication

Mohamed Lamine Tazir, Tawsif Gokhool, Paul Checchin,Laurent Malaterre,Laurent Trassoudaine, "CICP: Cluster Iterative Closest Point for Sparse-Dense Point Cloud Registration.", **Robotics and Autonomous Systems Journal (RAS), 2018**.

A.2 Peer-Reviewed Conference Papers

1. **Mohamed Lamine Tazir**, Tawsif Gokhool, Paul Checchin, Laurent Malaterre, Laurent Trassoudaine, "Cluster ICP: Towards Sparse to Dense Registration.", **15th International Conference on Intelligent Autonomous Systems IAS-15, (IAS 2018)**.

2. **Mohamed Lamine Tazir**, Paul Checchin, Laurent Trassoudaine,"Color-based 3D Point Cloud Reduction", **The 14th International Conference on Control,Automation, Robotics and Vision, (ICARCV 2016)**.

3. A.K. Aijazi, Laurent Malaterre, **Mohamed Lamine Tazir**, Paul Checchin, Laurent Trassoudaine, "Detecting And Analyzing Corrosion Spots On The Hull Of Large Marine Vessels Using Colored 3d Lidar Point Clouds", **ISPRS Annals of the Photogrammetry, Remote**

APPENDIX B

Implementation & simulation

In this chapter, we will briefly detail the used middlewares ROS and PCL, as well as, the 3D simulators used, Gazebo and RVIZ.

B.1 ROS

ROS for Robot Operating System [Quigley 2009] is an environment that facilitates the development of robotics applications. It includes libraries and tools to provide hardware abstractions, device drivers, message passing between processes, etc. Not only but also high-level features (asynchronous and synchronous communications, centralized database, robot configuration system, etc.). It has been developed by Willow Garage [1] since 2007 around its PR2 robot. Since then, many robotic platforms are compatible. One of the advantages of ROS is that it is an open source system with a strong community. The programs are built as ROS "nodes", which connect to a single "master", the *"ROSCORE"*. The basic principle of ROS is to run in parallel a large number of executables that must be able to exchange information in a synchronous or asynchronous manner. For example, the Master must interrogate at a defined frequency the robot sensors (LiDAR, cameras, gyroscope, etc.), retrieve this information, process it (data fusion, faltering, etc.), switch it to processing algorithms (mapping and localization, path planning, obstacles avoidance,...) and finally controlling the engines. This whole process is carried out continuously and in parallel.

ROS responds to this entire problem thanks to simple basic notions. The first notion is the "Node" concept. In ROS, a node is an instance of an executable. A node can correspond to a sensor, an engine, a processing algorithm, etc. The second notion is the "Master", a master is a service of declaration and registration of the nodes, which allows nodes to know each other and exchange information. The exchange of information is done either asynchronously via a "topic" or synchronously via a "service". A topic is an information transport system based on the subscription/publishing system. One or more nodes can publish information on a topic and one or more nodes can read the information on this topic. The topic is somehow an asynchronous information bus that publishes information (message) that is always structured in the same way. An example that summarizes all these notions is given by the Figure B.1.

Another interesting contribution from ROS to robotics is the Unified Robot Description Format (URDF). It is an XML format for describing a complete robot in the form of a standardized file. The described robot can be static or dynamic, and physical and collision properties can be added. In addition to the standard, ROS offers several tools for generating, parsing, or validating this format. The URDF is used in our case to represent the VIPA

[1] Source: http://www.willowgarage.com/

vehicle and its environment. Like any complete and complex tool, ROS requires a certain time to master it. All the developments of this project were carried out under *Ubuntu 14.04*, using the robotic operating system *ROS Indigo*.

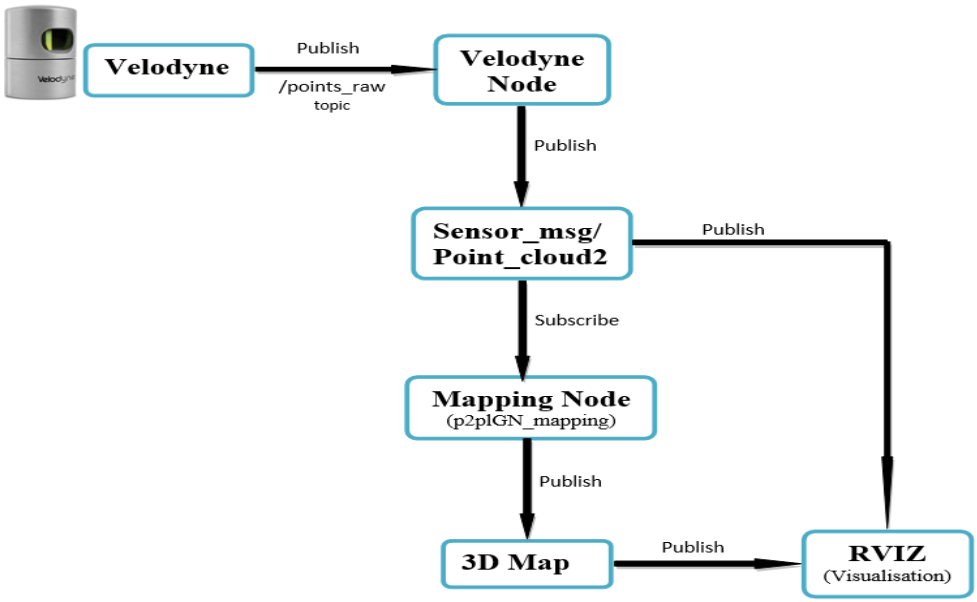


Figure B.1: *Nodes and topic in ROS.*

B.2 PCL

The Point Cloud Library (PCL) [Rusu 2011, Holz 2015] is an open source framework for point cloud and 3D geometry processing. It contains numerous state-of-the art algorithms including feature estimation, surface reconstruction, 3D registration, filtering, model fitting, and segmentation.

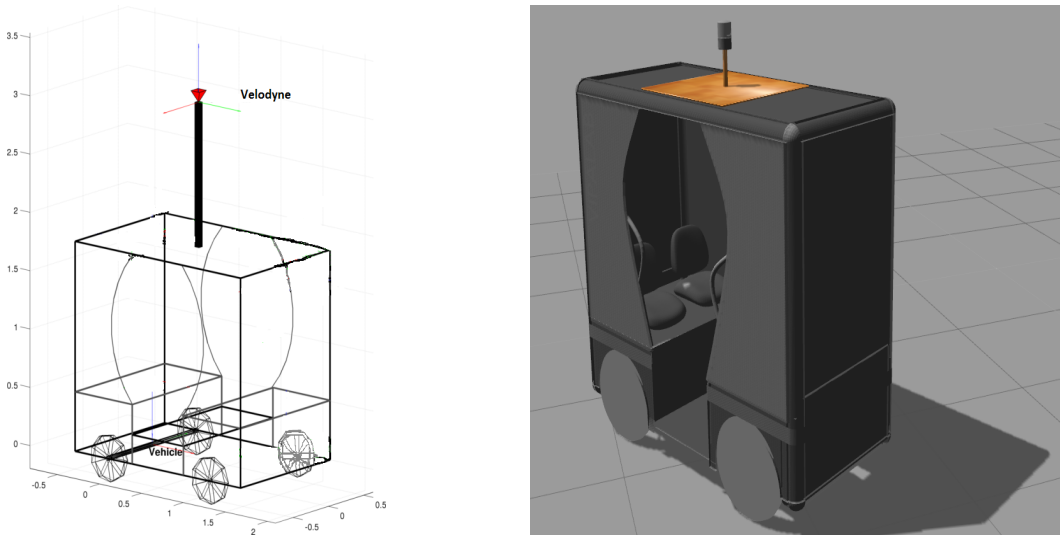
B.3 Simulation

B.3.1 Gazebo

Developed by the Open Source Robotics Foundation (OSRF) now Open Robotics [2], Gazebo is an interface to a virtual robotic world that can control a set of robots in indoor or outdoor environment. It provides a library of virtual worlds components. It is able to simulate a set of robots, sensors, and objects, but does so in a three-dimensional world with high-quality graphics. It generates realistic information for sensors and simulates physical interactions between objects (by integrating a robust physics engine).

[2]Source: https://www.openrobotics.org/

Vipa simulation in Gazebo The VIPA model is shown in Figure B.2 (left). We simulated its chassis and the wheels. We also modeled the wheels joints. Finally, we use only one sensor which is the Velodyne HDL 32E. This sensor is attached to the roof of the Vipalab at more than 2 meters from the ground, as shown by the Figure 6.6. It should be noted that, the sensor modeling is difficult because it strongly depends on the processing developed by the manufacturer in the internal electronics of the sensor. Simulation is, therefore, an approximate tool of reality.



(a) (b)

Figure B.2: *VIPA model (left) and its simulation in Gazebo (right)*

Pavin PAVIN is an experimental site for the development of automated vehicles in a realistic urban environment. Its model simulated in Gazebo is given by Figure B.3. Note that, this model is not a copy of reality (millimeter precision). It is designed from rough dimensions, but it allows having a fairly accurate representation of the reality. The purpose of this simulation is to validate the designed algorithms.

B.3.2 RVIZ

RVIZ is different from gazebo in the fact that it shows only what the robot knows about its world. In other words, it shows the sensor output. The transition between the different benchmarks (world, vehicle, sensor, etc.) is ensured by the transformation matrices defined in the ROS TF. This is shown in Figure B.4.

A very effective advantage of RVIZ that allows gaining enormously in terms of development time is the fact of recording and re-playing real data. For example, it is possible to perform a LiDAR data recording during a real test with the VIPA vehicle. Then replayed and viewed this data with RVIZ. This allows reproducing this test for several times, which enable the acceleration of the development process, as we have already mentioned.

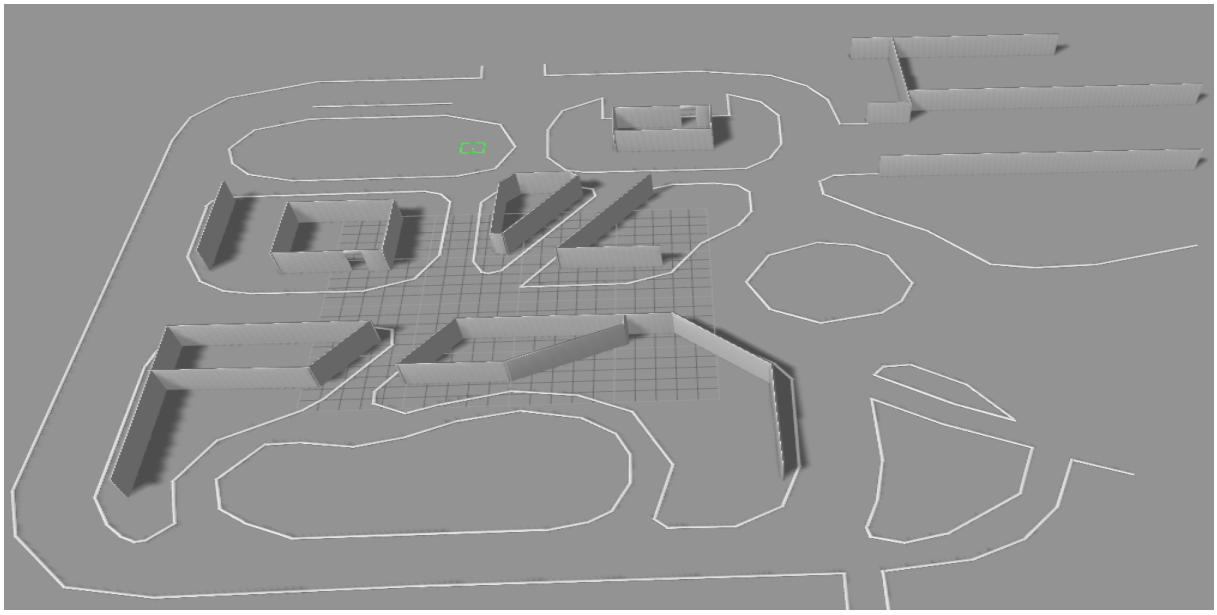


Figure B.3: *Pavin's model in Gazebo.*

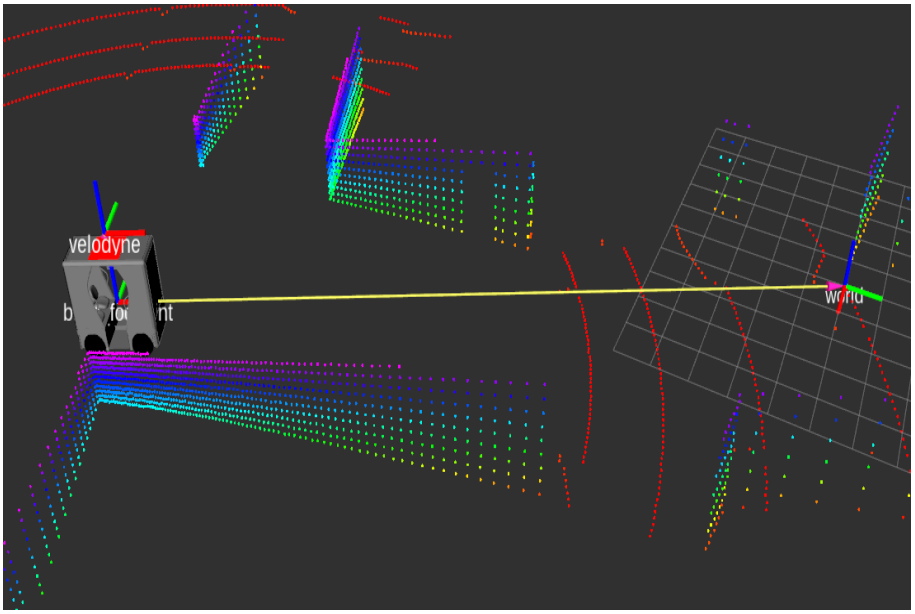


Figure B.4: *The VIPA model in RVIZ with benchmarks transformation.*

Bibliography

[A. Donoso 2016] F. A. Donoso, K.J. Austin and P.R. McAree. *How do ICP variants perform when used for scan matching terrain point clouds?* Robotics and Autonomous Systems, 2016. (Cited on pages 87 and 95.)

[A.C. Schultz 1998] A.C. Schultz and W. Adams. *Continuous Localization using evidence Grids*. In IEEE International Conference on Robotics and Automation, pages 2833–2839, 1998. (Cited on page 47.)

[Agamennoni 2016] Gabriel Agamennoni, Simone Fontana, Roland Y Siegwart and Domenico G Sorrenti. *Point Clouds Registration with Probabilistic Data Association*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4092–4098, 2016. (Cited on pages 83, 84 and 106.)

[Agarwal] Sameer Agarwal, Keir Mierle and Others. *Ceres Solver*. `http://ceres-solver.org`. (Not cited.)

[Aghaboni 1981] Fataneh Aghaboni and J. M. A. Tanchoco. *A LISP-based controller for free-ranging automated guided vehicle systems*. International Journal of Production Research, vol. 26, no. 02, pages 173–188, 1981. (Cited on page 51.)

[Akca 2005] D Akca. *Registration of point clouds using range and intensity information*. In The International Workshop on Recording, Modeling . . . , pages 115–126, 2005. (Cited on page 87.)

[Al-nuaimi 2017] Anas Al-nuaimi. *Methods of Point Cloud Alignment with Applications to 3D Indoor Mapping and Localization*. PhD thesis, 2017. (Cited on page 90.)

[Aldoma 2011] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu and Gary Bradski. *CAD-model recognition and 6DOF pose estimation using 3D cues*. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 585–592. IEEE, 2011. (Cited on pages 83 and 108.)

[Alexa 2001] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C.T. Silva. *Point set surfaces*. In 12th IEEE Visualization Proceedings,VIS '01., pages 21–28, 2001. (Cited on page 167.)

[Anderson 2018] Zackary Martin Anderson and Marco Giovanardi. *Self-driving vehicle with integrated active suspension*, 2018. (Cited on pages 24 and 58.)

[Arthur 2007] David Arthur and Sergei Vassilvitskii. *K-means++: The Advantages of Careful Seeding*. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. (Cited on pages 74, 113 and 171.)

[Arun 1987] K. S. Arun, T. S. Huang and S. D. Blostein. *Least-Squares Fitting of two 3-D Point Sets*. In IEEEE Transactions on pattern analysis and machine intelligence, pages 698 – 700, 1987. (Cited on page 69.)

[Bailey 2006] Tim Bailey and Hugh Durrant-whyte. *Simultaneous Localization and Mapping (SLAM)*. In Robotics & Automation Magazine, volume 13, pages 108–117, 2006. (Cited on pages 50 and 55.)

[Baldwin 2012] Ian Baldwin and Paul Newman. *Laser-only road-vehicle localization with dual 2D push-broom LIDARS and 3D priors*. IEEE International Conference on Intelligent Robots and Systems, pages 2490–2497, 2012. (Cited on page 178.)

[Bay 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. *Speeded-up robust features (SURF)*. Computer vision and image understanding, vol. 110, no. 3, pages 346–359, 2008. (Cited on page 82.)

[Bellekens 2014] Ben Bellekens, Vincent Spruyt and Rafael Berkvens Maarten Weyn. *A survey of rigid 3D pointcloud registration algorithms*. In Fourth International Conference on Ambient Computing, Applications, Services and Technologies, Proceedings, pages 8–13. IARA, 2014. (Cited on page 106.)

[Ben-Moshe 2014] Boaz Ben-Moshe, Paz Carmi and Eran Friedman. *Modeling GNSS Signals in Urban Canyons using Visibility Graphs and 3D Building Models*. pages 1–14, 2014. (Cited on page 48.)

[Benenson 2008] Rodrigo Benenson. *Perception for driverless vehicles : design and implementation*. PhD thesis, Ecole des Mines de Paris, 2008. (Cited on page 7.)

[Besl 1992] Paul J. Besl and Neil D. McKay. *A Method for Registration of 3-D Shapes*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 2, pages 239–256, 1992. (Cited on pages 48, 68, 81, 82, 86, 87, 89, 106 and 108.)

[bib 2015] Toward Object-based Place Recognition in Dense RGB-D Maps, 05/2015 2015. (Cited on page 108.)

[Birek 2018] Lech Birek, Adam Grzywaczewski, Rahat Iqbal, Faiyaz Doctor and Victor Chang. *A novel Big Data analytics and intelligent technique to predict driver's intent*. Computers in Industry, vol. 99, no. February, pages 226–240, 2018. (Cited on page 14.)

[Birem 2015] Merwan Birem. *Localisation et détection de fermeture de boucle basées saillance visuelle: algorithmes et architectures matérielles*. PhD thesis, 2015. (Cited on page 47.)

[Biskup 2007] K. Biskup, P. Arias, H. Lorenzo and J. Armesto. *Application of terrestrial laser scanning for shipbuilding*. In IAPRS Workshop on Laser Scanning, pages 56–61, 2007. (Cited on page 38.)

[Bjorck 1996] Ake Bjorck. Numerical methods for least squares problems. Siam, 1996. (Cited on page 115.)

[Blais 1995] Gerard Blais and Martin D. Levine. *Registering multiview range data to create 3D computer objects*. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 17, no. 8, pages 820–824, 1995. (Cited on page 167.)

[Blanco 2010] José-Luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Technical report, University of Malaga, 2010. (Cited on page 116.)

[Boissonnat 2001] J.-D. Boissonnat and F. Cazals. *Coarse-to-fine surface simplification with geometric guarantees*. Computer Graphics Forum, vol. 20, no. 3, pages 490–499, 2001. (Cited on page 167.)

[Bosse 2013] Michael Bosse and Robert Zlot. *Place recognition using keypoint voting in large 3D lidar datasets*. In IEEE International Conference on Robotics and Automation, pages 2677–2684, 2013. (Cited on page 178.)

[Bresson 2017] Guillaume Bresson, Zayed Alsayed, Li Yu and Sebastien Glaser. *Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving*. IEEE Transactions on Intelligent Vehicles, pages 1–1, 2017. (Cited on page 152.)

[Brubaker 2016] Marcus A. Brubaker, Andreas Geiger and Raquel Urtasun. *Map-based probabilistic visual self-localization*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 4, pages 652–665, 2016. (Cited on page 32.)

[Cadena 2016] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J.J. Leonard. *Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age*. IEEE Transactions on Robotics, vol. 32, no. 6, page 1309–1332, 2016. (Cited on pages 50, 55, 83 and 108.)

[Carvalho 2015] Ashwin Carvalho, Stéphanie Lefévre, Georg Schildbach, Jason Kong and Francesco Borrelli. *Automated driving: The role of forecasts and uncertainty - A control perspective*. European Journal of Control, vol. 24, pages 14–32, 2015. (Cited on page 11.)

[Caselitz 2016] Tim Caselitz, Bastian Steder, Michael Ruhnke and Wolfram Burgard. *Monocular Camera Localization in 3D LiDAR Maps*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–6, 2016. (Cited on pages 21, 31, 41, 57, 81, 106 and 179.)

[Castellanos 1999] J.A. Castellanos, J.M.M. Montiel, J. Neira and J.D. Tardos. *The SPmap: a probabilistic framework for simultaneous localization and map building*. IEEE Transactions on Robotics and Automation, vol. 15, no. 5, pages 948–952, 1999. (Cited on page 55.)

[Celebi 2013] M. Emre Celebi, Hassan A.Kingravi and Patricio A.Vela. *Clustering Algorithm*. Expert Systems with Applications, vol. 40, no. 1, pages 200–210, 2013. (Cited on page 74.)

[Chatila 1985] R. Chatila and J. Laumond. *Position referencing and consistent world modeling for mobile robots*. In IEEE International Conference on Robotics and Automation, pages 138–145, 1985. (Cited on page 55.)

[Chen 1991] Y. Chen and G. Medioni. *Object modeling by registration of multiple range images*. In IEEE International Conference on Robotics and Automation, pages 2724–2729, 4 1991. (Cited on pages 68, 81, 82, 84, 86, 87, 89, 90, 106 and 108.)

[Chirca 2016] Mihai Chirca. *Perception pour la navigation et le contrôle des robots mobiles Application à un système de voiturier autonome*. PhD thesis, UNIVERSITÉ BLAISE PASCAL - CLERMONT II, 2016. (Cited on page 31.)

[Choi 2014] Jaebum Choi. *Hybrid map-based SLAM using a Velodyne laser scanner*. In 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 3082–3087, 2014. (Cited on pages 38 and 146.)

[Costa 2016] Carlos M. Costa, Héber M. Sobreira, Armando J. Sousa and Germano M. Veiga. *Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms*. Robotics and Autonomous Systems, vol. 76, no. C, pages 113–140, 2016. (Cited on pages 82, 83, 87, 115 and 181.)

[Crowley 1989] J.L. Crowley. *World modeling and position estimation for a mobile robot using ultrasonic ranging*. In IEEE Conference on Robotics and Automation (ICRA), pages 674–680, 1989. (Cited on page 55.)

[Curiel-Ramirez 2018] Luis A. Curiel-Ramirez, Ricardo A. Ramirez-Mendoza, Gerardo Carrera, Javier Izquierdo-Reyes and M. Rogelio Bustamante-Bello. *Towards of a modular framework for semi-autonomous driving assistance systems*. International Journal on Interactive Design and Manufacturing (IJIDeM), 2018. (Cited on page 3.)

[Das 2014] Arun Das, Michael Diu, Neil Mathew, Christian Scharfenberger, James Servos, Andy Wong, John S Zelek, David A Clausi and Steven L Waslander. *Mapping, planning, and sample detection strategies for autonomous exploration*. Journal of Field Robotics, vol. 31, no. 1, pages 75–106, 2014. (Cited on pages 84 and 135.)

[Dean Pomerleau 1995] Dean Pomerleau. *RALPH: Rapidly adapting lateral position handler*. In Proceedings of the Intelligent Vehicles Symposium, pages 506 – 511, 1995. (Cited on page 6.)

[Dellaert 1999] Frank Dellaert, Dieter Fox, Wolfram Burgard and Sebastian Thrun. *Monte Carlo localization for mobile robots*. In Proceedings 1999 IEEE International Conference on Robotics and Automation, pages 1322–1328, 1999. (Cited on page 55.)

[Dellaert 2012] Frank Dellaert. *Factor Graphs and GTSAM : A Hands-on Introduction*. Technical report, 2012. (Cited on page 55.)

[Deschaud 2018] Jean-Emmanuel Deschaud. *IMLS-SLAM: scan-to-model matching based on 3D data*. In International Conference on Robotics and Automation, pages 1–6, 2018. (Cited on page 40.)

[Dey 2001] T.K. Dey and J. Giesen. *Decimating samples for mesh simplification*. In 13th Canadian Conf. on Computational Geometry, pages 85–88, 2001. (Cited on page 167.)

[Dickmanns 1992] Ernst D. Dickmanns and Birger D. Mysliwetz. *Recursive 3-D Road and Relative Ego-State Recognition*, 1992. (Cited on page 6.)

[Dickmanns 2003] E. D. Dickmanns. *The development of machine vision for road vehicles in the last decade*. IEEE Intelligent Vehicles Symposium, Proceedings, vol. 1, pages 268–281, 2003. (Cited on page 7.)

[Dissanayake 2011] Gamini Dissanayake, Shoudong Huang, Zhan Wang and Ravindra Ranasinghe. *A Review of Recent Developments in Simultaneous Localization and Mapping*. In 6th International Conference on Industrial and Information Systems, pages 477–482, 2011. (Cited on page 55.)

[Donoso 2017] F.A. Donoso, K.J. Austin and P.R. McAree. *How do ICP variants perform when used for scan matching terrain point clouds?* Robotics and Autonomous Systems, vol. 87, pages 147 – 161, 2017. (Cited on pages 111 and 115.)

[Dubé 2016] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart and Cesar Cadena. *SegMatch: Segment based loop-closure for 3D point clouds*. arXiv preprint arXiv:1609.07720, 2016. (Cited on pages 84, 85 and 108.)

[Durrant-Whyte 2006] Hugh Durrant-Whyte and Tim Bailey. *Simultaneous localization and mapping (SLAM): part I The Essential Algorithms*. In Robotics & Automation Magazine, volume 2, pages 99–110, 2006. (Cited on pages 50 and 55.)

[Eggert 1997] D.W. Eggert, A. Lorusso and R.B. Fisher. *Estimating 3-D rigid body transformations: a comparison of four major algorithms*. Machine Vision and Applications, vol. 9, no. 5-6, pages 272–290, 1997. (Cited on page 69.)

[Elfes 1989] Alberto Elfes. *Using Occupancy Grids for Mobile Robot Perception and Navigation*. IEEE Computer, vol. 22, no. 6, pages 46–57, 1989. (Cited on page 47.)

[Elseberg 2012] Jan Elseberg, Stéphane Magnenat, Roland Siegwart and Nüchter Andreas. *Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration*. Journal of Software Engineering for Robotics (JOSER), vol. 3, no. 1, pages 2–12, 2012. (Cited on page 87.)

[Fabio 2003] R. Fabio. *From point cloud to surface: the modeling and visualization problem*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 34, page 11, 2003. (Cited on page 81.)

[Fang 2017] Zheng Fang, Shibo Zhao and Shiguang Wen. *A Real-time and Low-cost 3D SLAM System Based on a Continuously Rotating 2D Laser Scanner*. 2017. (Cited on page 36.)

[Feng 2016] Yu Feng, Alexander Schlichting and Claus Brenner. *3D feature point extraction from LiDAR data using a neural network*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 41 (2016), vol. 41, pages 563–569, 2016. (Cited on pages 82, 83 and 108.)

[Fernández-Moral 2013] Eduardo Fernández-Moral, Walterio Mayol-Cuevas, Vicente Arévalo and Javier Gonzalez-Jimenez. *Fast place recognition with plane-based maps*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 2719–2724. IEEE, 2013. (Cited on pages 85 and 108.)

[Fernández-Moral 2016] Eduardo Fernández-Moral, Patrick Rives, Vicente Arévalo and Javier González-Jiménez. *Scene structure registration for localization and mapping*. In Robotics and Autonomous Systems, pages 649–660, 2016. (Cited on pages 84 and 108.)

[Filipe 2014] Silvio Filipe and Luís A Alexandre. *A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset*. In Computer Vision Theory and Applications (VISAPP), 2014 International Conference on, volume 1, pages 476–483. IEEE, 2014. (Cited on page 82.)

[Filliat 2011] David Filliat. Robotique Mobile. 2011. (Cited on page 44.)

[Fischler 1981] Martin A Fischler and Robert C Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, vol. 24, no. 6, pages 381–395, 1981. (Cited on page 85.)

[Ford 2011] David N. Ford, Thomas J. Housel and Johnathan C. Mun. *Ship Maintenance Processes With Collaborative Product Lifecycle Management and 3D Terrestrial Laser Scanning Tools: Reducing Costs and Increasing Productivity*. Technical report, 2011. (Cited on page 38.)

[Forsberg 1995] Johan Forsberg, Ulf Larsson and Ake Wernersson. *Mobile robot navigation using the range-weighted Hough transform*. IEEE Robotics & Automation Magazine, vol. 2, no. 1, pages 18–26, 1995. (Cited on page 85.)

[Fridman 2018] Lex Fridman. *Self-Driving Cars*, 2018. (Cited on pages 39 and 41.)

[Geiger 2012] Andreas Geiger, Philip Lenz and Raquel Urtasun. *Are we ready for autonomous driving? the KITTI vision benchmark suite*. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3354–3361, 2012. (Cited on page 152.)

[Georgiev 2011] Kristiyan Georgiev, Ross T Creed and Rolf Lakaemper. *Fast plane extraction in 3D range data based on line segments*. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 3808–3815. IEEE, 2011. (Cited on page 85.)

[Godin 1994] Guy Godin, Marc Rioux and Rejean Baribeau. *Three-dimensional registration using range and intensity information*. In The International Society for Optical Engineering, SPIE, pages 279–290, 1994. (Cited on page 88.)

[Gordon 2013] M. Gordon and J. Meidow. *Calibration of a multi-beam Laser System by using a TLS-generated Reference*. In ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, pages 85–90, 2013. (Cited on page 38.)

[Grant 2013] W. Shane Grant, Randolph C. Voorhies and Laurent Itti. *Finding planes in LiDAR point clouds for real-time registration*. In IEEE International Conference on Intelligent Robots and Systems, pages 4347–4354, 2013. (Cited on pages 85, 106 and 108.)

[Gressin 2012] A. Gressin, C. Mallet and N. David. *Improving 3D Lidar Point Cloud Registration Using Optimal Neighborhood Knowledge*. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. I-3, no. September, pages 111–116, 2012. (Cited on pages 86 and 87.)

[Guo 2017] Xiuyan Guo. *Feature-Based Localization Methods for Autonomous Vehicles*. PhD thesis, University of Berlin, 2017. (Cited on page 25.)

[Häne 2017] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler and Marc Pollefeys. *3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection*. Image and Vision Computing, vol. 68, pages 14–27, 2017. (Cited on pages 24 and 58.)

[Hänsch 2014] R. Hänsch, T. Weber and O. Hellwich. *Comparison of 3D interest point detectors and descriptors for point cloud fusion*. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. II-3, no. September, pages 57–64, 2014. (Cited on page 82.)

[He 2016] Li He, Xiaolong Wang and Hong Zhang. *M2DP: A novel 3D point cloud descriptor and its application in loop closure detection*. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 231–237. IEEE, 2016. (Cited on pages 111 and 135.)

[Hentschel 2008] Matthias Hentschel, Oliver Wulf and Bernardo Wagner. *A GPS and laser-based localization for urban and non-urban outdoor environments*. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pages 149–154, 2008. (Cited on pages 41 and 56.)

[Heritage 2009] George L.; Heritage and Andrew R.G. Large. Laser Scanning for the Environmental Sciences. 2009. (Cited on page 37.)

[Holz 2015] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu and Sven Behnke. *Registration with the Point Cloud Library PCL*. IEEE Robotics & Automation Magazine, vol. 22, no. 4, pages 1–13, 2015. (Cited on pages 84, 87, 88, 94, 106, 108, 111, 114, 135, 152 and 200.)

[Horn 1987] Berthold K P Horn. *Closed-form solution of absolute orientation using unit quaternions*. Journal of the Optical Society of America A, vol. 4, no. 4, page 629, 1987. (Cited on page 69.)

[Hornung 2013] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss and Wolfram Burgard. *OctoMap: An efficient probabilistic 3D mapping framework based on octrees*. Autonomous Robots, vol. 34, no. 3, pages 189–206, 2013. (Cited on pages 47 and 48.)

[Hossain 2018] Sabir Hossain, Abdur R Fayjie, Oualid Doukhi and Deok-jin Lee. *Intelligent Computing*. In International Conference on Intelligent Computing & Optimization, pages 187–195. Springer International Publishing, 2018. (Cited on page 151.)

[Huber 1981] Peter J Huber. Robust statistics, chapitre The Basic Types of Estimates, pages 45–67. Wiley, New York, New York, 1981. (Cited on pages 88 and 115.)

[Huber 2009] Peter J Huber and Elvezio M. Ronchetti. Robust statistics. John Wiley & Sons, 2009. (Cited on pages 73, 88 and 115.)

[Igelbrink 2015] Tristan Igelbrink, Thomas Wiemann and Joachim Hertzberg. *Generating Topologically Consistent Triangle Meshes from Large Scale Kinect Fusion*. In IEEE European Conf. on Mobile Robots (ECMR), pages 1–6, 2015. (Cited on page 170.)

[Jacques Feldmar, Nicholas Ayache 1994] Fabienne Betting Jacques Feldmar, Nicholas Ayache. *3D-2D projective registration of free-form curves and surfaces*. Technical report, 1994. (Cited on page 87.)

[Jagbrant 2013] G. Jagbrant, J.P. Underwood, J. Nieto and S. Sukkarieh. *Lidar Based Localisation in Almond Orchards*. In 9th Conference on Field and Service Robotics, 2013. (Cited on page 54.)

[Jain 2010] Anil K Jain. *Data clustering : 50 years beyond K-means q*. PATTERN RECOGNITION LETTERS, vol. 31, no. 8, pages 651–666, 2010. (Cited on page 74.)

[Jeong 2018] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh and Ayoung Kim. *Complex Urban LiDAR Data Set*. In IEEE International Conference on Robotics and Automation (ICRA),, pages 1–8, 2018. (Cited on page 152.)

[Jolliffe 1986] I. T. Jolliffe. Principal Component Analysis for Special Types of Data, chapitre 11, pages 199–222. Springer New York, New York, NY, 1986. (Cited on pages 83, 111 and 180.)

[Jomrich 2017] Florian Jomrich, Aakash Sharma, Tobias Rückelt, Daniel Burgstahler and Doreen Böhnstedt. *Dynamic Map Update Protocol for Highly Automated Driving Vehicles*. In Proceedings of the International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS), pages 68–78, 2017. (Cited on page 45.)

[Joukhadar 2018] A Joukhadar, H Issa and Y Kalaji. *Design and implementation of auto car driving system with collision avoidance*. Cogent Engineering, vol. 33, no. 0, pages 1–19, 2018. (Cited on page 51.)

[Kampker 2018] Prof Achim Kampker, Jonas Hatzenbuehler, Lars Klein, Mohsen Sefati, Kai D Kreiskoether and Denny Gert. *Concept Study for Vehicle Self-Localization Using Neural Networks for Detection of Pole-Like Landmarks*. In Proceedings of the 15th International Conference on Intelligent Autonomous Systems, 2018. (Cited on page 54.)

[Kaufman 1990] L. Kaufman and P.J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis, volume 33. John Wiley & Sons, Inc., 1990. (Cited on pages 74, 170 and 171.)

[Khoshelham 2013] K Khoshelham, D Dos Santos and G. Vosselman. *Generation and weighting of 3D point correspondences for improved registration of RGB-D data*. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,, vol. II, no. 5, pages 11–13, 2013. (Cited on page 88.)

[Klasing 2009] Klaas Klasing, Daniel Althoff, Dirk Wollherr and Martin Buss. *Comparison of surface normal estimation methods for range sensing applications*. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 3206–3211. IEEE, 2009. (Cited on page 111.)

[Konrad 2012] Marcus Konrad, Dominik Nuss and Klaus Dietmayer. *Localization in digital maps for road course estimation using grid maps*. IEEE Intelligent Vehicles Symposium, Proceedings, pages 87–92, 2012. (Cited on page 57.)

[Korrapati 2013] Hemanth Korrapati, Jonathan Courbon, Serge Alizon and François Marmoiton. *"The Institut Pascal Data Sets" : un jeu de données en extérieur, multicapteurs et datées avec réalité terrain, étalonnage et outils logiciels*. In Orasis, Congrès des jeunes chercheurs en vision par ordinateur, 2013. (Cited on page 152.)

[Kostavelis 2015] Ioannis Kostavelis and Antonios Gasteratos. *Semantic mapping for mobile robotics tasks: A survey*. Robotics and Autonomous Systems, vol. 66, pages 86–103, 2015. (Cited on page 49.)

[Kuemmerle 2011] Rainer Kuemmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti and Wolfram Burgard. *Large Scale Graph-based SLAM Using Aerial Images As Prior Information*. Autonomous Robots, vol. 30, no. 1, pages 25–39, 2011. (Cited on pages 55 and 56.)

[Kummerle 2011] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige and Wolfram Burgard. *g2o: A General Framework for Graph Optimization*. In IEEE International Conference on Robotics and Automation (ICRA), pages 1–7, 2011. (Not cited.)

[Kurdej 2015] Marek Kurdej. *Exploitation of map data for the perception of intelligent vehicles*. PhD thesis, Université de Technologie de Compiègne, 2015. (Cited on pages 22 and 48.)

[Lalonde 2006] Jean François Lalonde, Nicolas Vandapel, Daniel F. Huber and Martial Hebert. *Natural terrain classification using three-dimensional ladar data for ground robot mobility*. Journal of Field Robotics, vol. 23, no. 10, pages 839–861, 2006. (Cited on page 87.)

[Lee 2001] K.H. Lee, H. Woo and T. Suk. *Point Data Reduction Using 3D Grids*. The Int. Journal of Advanced Manufacturing Technology, vol. 18, no. 3, pages 201–210, 2001. (Cited on page 166.)

[Lee 2006] Pai-Feng Lee, Chien-Hsing Chiang, Juin-Ling Tseng, Bin-Shyan Jong and Tsong-Wuu Lin. *Octree Subdivision Using Coplanar Criterion for Hierarchical Point Simplification*. In IEEE Region 10 Conf. TENCON, pages 54–63, 2006. (Cited on pages 166 and 167.)

[Lee 2008] Pai Feng Lee and Bin Shyan Jong. *Point-based simplification algorithm*. WSEAS Trans. on Computer Research, vol. 3, no. 1, pages 61–66, 2008. (Cited on pages 165 and 167.)

[Lenac 2017] Kruno Lenac, Andrej Kitanov, Robert Cupec and Ivan Petrović. *Fast planar surface 3D SLAM using LIDAR*. Robotics and Autonomous Systems, vol. 92, pages 197–220, 2017. (Cited on page 108.)

[Leonard 1991] J.J. Leonard and H.F. Durrant-Whyte. *Simultaneous map build- ing and localisation for an autonomous mobile robot*. In IEEE Int. Workshop on Intelligent Robots and Systems (IROS), pages 1442–1447, 1991. (Cited on page 55.)

[Levinson 2007] Jesse Levinson, Michael Montemerlo and Sebastian Thrun. *Map-Based Precision Vehicle Localization in Urban Environments*. In Robotics: Science and Systems III, pages 121–128, 2007. (Cited on pages 20, 32, 56, 57 and 178.)

[Levinson 2010a] Jesse Levinson and Sebastian Thrun. *Robust vehicle localization in urban environments using probabilistic maps*. In 2010 IEEE International Conference on Robotics and Automation, pages 4372–4378, 2010. (Cited on pages 56 and 178.)

[Levinson 2010b] Jesse Levinson and Sebastian Thrun. *Unsupervised calibration for multi-beam lasers*. In International Symposium on Experimental Robotics, pages 179–193, 2010. (Cited on page 38.)

[Levinson 2011] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling and Sebastian Thrun. *Towards Fully Autonomous Driving: Systems and Algorithms*. In IEEE Intelligent Vehicles Symposium (IV), pages 3–8, 2011. (Cited on pages 24, 38 and 58.)

[Li 2012] Yichen Li, Mingqiang Wei, Jianhuang Wu and Mingyong Pang. *Error-Controllable Simplification of Point Cloud*. In Trans. on Computational Science XVI, pages 149–162, 2012. (Cited on pages 165, 166, 167 and 168.)

[Li 2016] Minglei Li, Xinyuan Gao, Li Wang and Guangyun Li. *Automatic registration of laser-scanned point clouds based on planar features*. In 2nd ISPRS International Conference on Computer Vision in Remote Sensing (CVRS 2015), volume 9901, page 990103. International Society for Optics and Photonics, 2016. (Cited on page 108.)

[Li 2017] Jiaxin Li, Huangying Zhan, Ben M. Chen, Ian Reid and Gim Hee Lee. *Deep learning for 2D scan matching and loop closure*. In IEEE International Conference on Intelligent Robots and Systems, pages 763–768, 2017. (Cited on page 91.)

[Lim 2012] Jongwoo Lim, Jan Michael Frahm and Marc Pollefeys. *Online environment mapping using metric-topological maps*. International Journal of Robotics Research, vol. 31, no. 12, pages 1394–1408, 2012. (Cited on page 44.)

[Lin 2013] Chien-Chou Lin, Yan-Deng Liao and Wun-Jhih Luo. *Calibration Method for Extending Single-Layer LIDAR to Multi_layer LIDAR*. In International Symposium on System Integration,, pages 677–681, 2013. (Cited on page 37.)

[Loevsky 2010] I. Loevsky and I. Shimshoni. *Reliable and efficient landmark-based localization for mobile robots*. Robotics and Autonomous Systems, vol. 58, no. 5, pages 520–528, 2010. (Cited on page 53.)

[Lothe 2010] Pierre Lothe. *Localisation et cartographie simultanées par vision monoculaire contraintes par un SIG : Application à la géolocalisation d'un véhicule*. PhD thesis, UNIVERSITÉ BLAISE PASCAL - CLERMONT-FERRAND II, 2010. (Cited on pages 50 and 74.)

[Low 2004] Kl Low. *Linear Least-squares Optimization for Point-to-plane ICP Surface Registration*. Chapel Hill, University of North Carolina, no. February, pages 2–4, 2004. (Cited on page 90.)

[Lowe 1999] David G. Lowe. *Object recognition from local scale-invariant features*. In Proceedings of the Seventh IEEE International Conference on Computer Vision, pages 1150–1157, 1999. (Cited on page 82.)

[Luck 2000] Jason P Luck, Charles Q Little and William Hoff. *Registration of Range Data Using a Hybrid Simulated Annealing and Iterative Closest Point Algorithm.* In IEEE International Conference on Robotics and Automation, Icra, pages 3739–3744, 2000. (Cited on pages 88 and 90.)

[Lutin 2018] Jerome M Lutin. *Not If, but When: Autonomous Driving and the Future of Transit*. Journal of Public Transportation, vol. 21, no. 1, pages 92–103, 2018. (Cited on page 4.)

[Ma 2003] Y. Ma, S. Soatto, J. Kosecka̋ and S.S. Sastry. *Representation of a Three-Dimensional Moving Scene*. In An Invitation to 3D Vision, pages 15–43. 2003. (Cited on pages 63 and 65.)

[Ma 2004] Y. Ma, S. Soatto, J. Košecká and Shankar S Sastry. An invitation to 3-d vision. Springer, 2004. (Cited on page 116.)

[Maddern 2015] Will Maddern, Geoffrey Pascoe and Paul Newman. *Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities*. In IEEE International Conference on Robotics and Automation (ICRA),, pages 1–8, 2015. (Cited on page 179.)

[Maddern 2016] Will Maddern and Paul Newman. *Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo*. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 2181–2188. IEEE, 2016. (Cited on pages 82 and 106.)

[Madhavan 2004] R. Madhavan and H. F. Durrant-Whyte. *Natural landmark-based autonomous vehicle navigation*. Robotics and Autonomous Systems, vol. 46, no. 2, pages 79–95, 2004. (Cited on page 53.)

[Magnusson 2007] Martin Magnusson, Achim Lilienthal and Tom Duckett. *Scan registration for autonomous mining vehicles using 3D-NDT*. Journal of Field Robotics, vol. 24, no. 10, pages 803–827, 2007. (Cited on pages 84 and 108.)

[Maimone 2007] Mark Maimone, Yang Cheng and Larry Matthies. *Two years of visual odometry on the Mars Exploration Rovers*. Journal of Field Robotics, vol. 24, no. 3, pages 169–186, 2007. (Cited on page 41.)

[Marani 2016] Roberto Marani, Vito Reno, Massimiliano Nitti, Tiziana D'Orazio and Ettore Stella. *A modified iterative closest point algorithm for 3D point cloud registration*. Computer-Aided Civil and Infrastructure Engineering, vol. 31, no. 7, pages 515–534, 2016. (Cited on pages 81 and 82.)

[Markelj 2012] Primoz Markelj, Dejan Tomaževič, Bostjan Likar and Franjo Pernuš. *A review of 3D/2D registration methods for image-guided interventions*. Medical image analysis, vol. 16, no. 3, pages 642–661, 2012. (Cited on page 81.)

[Marmoiton 2016] François Marmoiton and Morgan SLADE. *Toward Smart Connected Cars*. In Intelligent Transportation Systems From Good Practices to Standards, pages 1–40. 2016. (Cited on page 17.)

[Masuda 1995] Takeshi Masuda and Naokazu Yokoya. *A Robust Method for Registration and Segmentation of Multiple Range Images*. Computer Vision and Image Understanding, vol. 61, no. 3, pages 295–307, 1995. (Cited on page 167.)

[Masuda 1996] T. Masuda, K. Sakaue and N. Yokoya. *Registration and integration of multiple range images for 3-D model construction*. In Proceedings of 13th International Conference on Pattern Recognition, pages 879–883, 1996. (Cited on page 86.)

[Mellado 2014] Nicolas Mellado, Dror Aiger and Niloy J. Mitra. *Super 4PCS fast global pointcloud registration via smart indexing*. Computer Graphics Forum, vol. 33, no. 5, pages 205–215, 2014. (Cited on page 108.)

[Merriaux 2016] Pierre Merriaux. *Contribution à la localisation robuste embarquée pour la navigation autonome*. PhD thesis, Université de Rouen, 2016. (Cited on page 59.)

[Mian 2010] Ajmal Mian, Mohammed Bennamoun and Robyn Owens. *On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes*. International Journal of Computer Vision, vol. 89, no. 2, pages 348–361, 2010. (Cited on page 83.)

[Montemerlo 2006] Michael Montemerlo and Sebastian Thrun. *Large-scale robotic 3-D mapping of urban structures*. Springer Tracts in Advanced Robotics, vol. 21, pages 141–150, 2006. (Cited on page 43.)

[Mou 2018] Shenyu Mou, Yan Chang, Wenshuo Wang and Ding Zhao. *An Optimal LiDAR Configuration Approach for Self-Driving Cars*. In arXiv, pages 1–7, 2018. (Cited on pages 38 and 146.)

[Mur-Artal 2015] Raul Mur-Artal, J. M.M. Montiel and Juan D. Tardos. *ORB-SLAM: A Versatile and Accurate Monocular SLAM System*. IEEE Transactions on Robotics, vol. 31, no. 5, pages 1147–1163, 2015. (Cited on pages 57 and 179.)

[Murphy 1999] Kevin P Murphy and Others. *Bayesian Map Learning in Dynamic Environments*. In Nips, pages 1015–1021, 1999. (Cited on page 55.)

[Nicolai 2016] Austin Nicolai, Ryan Skeele, Christopher Eriksen and Geoffrey A Hollinger. *Deep Learning for Laser Based Odometry Estimation*. In Science and Systems Conf. Workshop on Limits and Potentials of Deep Learning in Robotics, 2016. (Cited on page 91.)

[Nieto 2006] Juan Nieto, Tim Bailey and Eduardo Nebot. *Scan-SLAM: Combining EKF-SLAM and scan correlation*. Springer Tracts in Advanced Robotics, vol. 25, pages 167–178, 2006. (Cited on pages 83 and 85.)

[Nizard 2017] Ange Nizard. *Planification et commande pour véhicules à deux trains directeurs en milieu encombré To cite this version : Thèse Planification et commande pour véhicules à deux trains directeurs en milieu encombré*. PhD thesis, Université Clermont Auvergne, 2017. (Cited on page 17.)

[Nuchter 2004] A Nuchter, H Surmann, K Lingemann, J Hertzberg and S Thrun. *6D SLAM with an application in autonomous mine mapping*. In IEEE International Conference on Robotics and Automation, ICRA, pages 1998–2003, 2004. (Cited on pages 41 and 55.)

[Nüchter 2015] A. Nüchter, D. Borrmann, P. Koch, M. Kühn and S. May. *a Man-Portable, Imu-Free Mobile Mapping System*. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. II-3, no. W5, pages 17–23, 2015. (Cited on page 43.)

[Núñez 2008] P. Núñez, R. Vázquez-Martín, J. C. del Toro, A. Bandera and F. Sandoval. *Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation*. Robotics and Autonomous Systems, vol. 56, no. 3, pages 247–264, 2008. (Cited on page 53.)

[Obst 2012] Marcus Obst, Sven Bauer, Pierre Reisdorf and Gerd Wanielik. *Multipath detection with 3D digital maps for robust multi-constellation GNSS/INS vehicle localization in urban areas*. In IEEE Intelligent Vehicles Symposium, Proceedings, pages 184–190, 2012. (Cited on pages 48 and 49.)

[Ochieng 2003] W.Y. Ochieng, M. Quddus and R.B. Noland. *Map-matching in complex urban road networks*. Brazilian Journal of Cartography, vol. 55, no. 2, pages 1–14, 2003. (Cited on page 178.)

[Ohn-Bar 2016] Eshed Ohn-Bar and Mohan Manubhai Trivedi. *Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles*. IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pages 90–104, 2016. (Cited on page 11.)

[Olivares-Mendez 2011] Miguel A. Olivares-Mendez, Ignacio Mellado, Pascual Campoy, Ivan Mondragon and Carol Martinez. *A visual AGV-urban car using Fuzzy control*. In Proceedings of the 5th International Conference on Automation, Robotics and Applications, ICARA, pages 145–150, 2011. (Cited on pages 51 and 52.)

[Pandey 2011] Gaurav Pandey, Silvio Savarese, James R McBride and Ryan M Eustice. *Visually bootstrapped generalized ICP*. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 2660–2667. IEEE, 2011. (Cited on pages 108 and 152.)

[Pandžić 2014] J Pandžić, V Erić, B Božić and M Pejić. *The Accuracy Analysis of Leica ScanStation P20 Data by Means of Point Cloud Fitting Algorithm*. In 6th International Conference on Engineering Surveying, pages 101–106, 2014. (Cited on page 141.)

[Paparoditis 2012] Nicolas Paparoditis, Jean Pierre Papelard, Bertrand Cannelle, Alexandre Devaux, Bahman Soheilian, Nicolas David and Erwann Houzay. *Stereopolis II a multi-purpose and multi-sensor 3d mobile mapping system for street visualisation and 3d metrology*. In Revue Francaise de Photogrammetrie et de Teledetection, pages 69–79, 2012. (Cited on page 43.)

[Paromtchik 1996] I. Paromtchik and C. Laugier. *Autonomous parallel parking of a nonholo- nomic vehicle*. In Proceedings of the Intelligent Vehicles Symposium, pages 13–18, 1996. (Cited on page 7.)

[Paskin 2003] Mark A Paskin. *Thin Junction Tree Filters for Simultaneous Localization and Mapping Thin Junction Tree Filters for Simultaneous Localization and Mapping*. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI, 2003. (Cited on page 55.)

[Patidar 2016] Ayush Patidar and Prof Sanjiv. *A Review Paper on Self - Driving Car ' s and its Applications*. In National Conference on Innovations in Micro-electronics, Signal Processing and Communication Technologies IJIRST, pages 33–35, 2016. (Cited on page 106.)

[Paul 2018] Nicholas Paul and ChanJin Chung. *Application of HDR algorithms to solve direct sunlight problems when autonomous vehicles using machine vision systems are driving into sun*. Computers in Industry, vol. 98, pages 192–196, 2018. (Cited on page 39.)

[Pauly 2002] M. Pauly, M. Gross and L.P. Kobbelt. *Efficient simplification of point-sampled surfaces*. In 13th IEEE Visualization conference., pages 163–170, 2002. (Cited on page 165.)

[Pfeiffer 2017] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart and Cesar Cadena. *From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots*. In Proceedings - IEEE International Conference on Robotics and Automation, pages 1527–1533, 2017. (Cited on page 91.)

[Pomerleau 1996] D. Pomerleau and T. Jochem. *Rapidly adapting machine vision for automated vehicle steering*. Ieee Expert Intelligent Systems And Their Applications, vol. 11, no. 2, pages 19–27, 1996. (Cited on page 7.)

[Pomerleau 2011] François Pomerleau, Stéphane Magnenat, Francis Colas, Ming Liu and Roland Siegwart. *Tracking a Depth Camera : Parameter Exploration for Fast ICP*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), page 7, 2011. (Cited on page 148.)

[Pomerleau 2013a] François Pomerleau. *Applied registration for robotics*. PhD thesis, ETH ZURICH, 2013. (Cited on page 81.)

[Pomerleau 2013b] François Pomerleau, Francis Colas, Roland Siegwart and Stéphane Magnenat. *Comparing ICP variants on real-world data sets*. Autonomous Robots, vol. 34, no. 3, pages 133–148, 2013. (Cited on page 84.)

[Pomerleau 2015] François Pomerleau, Francis Colas and Roland Siegwart. *A Review of Point Cloud Registration Algorithms for Mobile Robotics*. Foundations and Trends in Robotics, vol. 4, no. 1-104, pages 1–104, 2015. (Cited on pages 81, 84 and 106.)

[Pronobis 2011] Andrzej Pronobis. *Semantic Mapping with mobile robots*. PhD thesis, KTH Royal Institute of Technology Stockholm, Sweden, 2011. (Cited on page 44.)

[Pulli 1999] Kari Pulli. *Multiview registration for large data sets*. In 3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on, pages 160–168. IEEE, 1999. (Cited on pages 87 and 89.)

[Puttonen 2013] Eetu Puttonen, Matti Lehtomäki, Harri Kaartinen, Lingli Zhu, Antero Kukko and Anttoni Jaakkola. *Improved sampling for terrestrial and mobile laser scanner point cloud data*. Remote Sensing, vol. 5, no. 4, pages 1754–1773, 2013. (Cited on pages 36, 131, 165, 166 and 167.)

[Qi 2016] Charles R. Qi, Hao Su, Kaichun Mo and Leonidas J.Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 601–610, 2016. (Cited on page 196.)

[Quddus 2007] M. Quddus, W. Y Ochieng and R. B Noland. *Current Map Matching Algorithms for Transport Applications: State-of-the art and Future Research Directions*. Transportation Research Part C: Emerging Technologies, vol. 15, no. 05, pages 312–328, 2007. (Cited on page 178.)

[Quigley 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler and Andrew Mg. *ROS: an open-source Robot Operating System*. In ICRA Workshop on Open Source Software, pages 1–6, 2009. (Cited on page 199.)

[Raoui 2011] Younes Raoui. *Indexation d'une base de données images : Application à la localisation et la cartographie fondées sur des etiquettes et des amers visuels pour la navigation d'un robot en milieu intérieur*. PhD thesis, Université de Toulouse, 2011. (Cited on page 48.)

[Razlaw 2015] Jan Razlaw, David Droeschel, Dirk Holz and Sven Behnke. *Evaluation of registration methods for sparse 3d laser scans*. In Mobile Robots (ECMR), 2015 European Conference on, pages 1–7. IEEE, 2015. (Cited on pages 106 and 108.)

[Recchiuto 2017] Carmine Tommaso Recchiuto and Antonio Sgorbissa. *Post-disaster assessment with unmanned aerial vehicles: A survey on practical implementations and research approaches*. Journal of Field Robotics, no. June 2016, pages 1–32, 2017. (Cited on page 43.)

[Rezaei 2014] Mahdi Rezaei and Reinhard Klette. *Look at the driver, look at the road: No distraction! No accident!* In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 129–136, 2014. (Cited on page 11.)

[Romero 2018] Adrian Rechy Romero, Paulo V K Borges, Andreas Pfrunder and Alberto Elfes. *Map-Aware Particle Filter for Localization*. In IEEE International Conference on Robotics and Automation (ICRA),, pages 2940–2947, 2018. (Cited on page 178.)

[Ronzoni 2011] Davide Ronzoni, Roberto Olmi, Cristian Secchi and Cesare Fantuzzi. *AGV global localization using indistinguishable artificial landmarks*. In IEEE International Conference on Robotics and Automation, pages 287–292, 2011. (Cited on pages 53 and 54.)

[Rosten 2006] Edward Rosten and Tom Drummond. *Machine learning for high-speed corner detection*. Computer Vision–ECCV 2006, pages 430–443, 2006. (Cited on page 82.)

[Royer 2007] Eric Royer, Maxime Lhuillier, Michel Dhome and Jean Marc Lavest. *Monocular vision for mobile robot localization and autonomous navigation*. International Journal of Computer Vision, vol. 74, no. 3, pages 237–260, 2007. (Cited on page 195.)

[Rublee 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. *ORB: An efficient alternative to SIFT or SURF*. In Computer Vision (ICCV), 2011 IEEE international conference on, pages 2564–2571. IEEE, 2011. (Cited on page 82.)

[Rusinkiewicz 2001] S. Rusinkiewicz and M. Levoy. *Efficient variants of the ICP algorithm*. In Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM, pages 145–152, 2001. (Cited on pages 85, 86, 87, 88 and 89.)

[Rusu 2008] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton and Michael Beetz. *Aligning Point Cloud Views using Persistent Feature Histograms*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pages 3384–3391, 2008. (Cited on pages 83 and 108.)

[Rusu 2009a] Radu Bogdan Rusu. *Semantic 3D object maps for everyday manipulation in human living environments*. PhD thesis, Technische Universität München, Diss., 2009, 2009. (Not cited.)

[Rusu 2009b] Radu Bogdan Rusu, Nico Blodow and Michael Beetz. *Fast Point Feature Histograms (FPFH) for 3D registration*. In IEEE International Conference on Robotics and Automation, pages 3212–3217, 2009. (Cited on pages 83, 87 and 108.)

[Rusu 2010] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux and John Hsu. *Fast 3d recognition and pose using the viewpoint feature histogram*. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 2155–2162. IEEE, 2010. (Cited on pages 83 and 108.)

[Rusu 2011] Radu Bogdan Rusu and S. Cousins. *3D is here: point cloud library*. In IEEE International Conference on Robotics and Automation (ICRA), pages 1 – 4, 2011. (Cited on pages 82, 85, 94, 114, 122, 171 and 200.)

[Salas-Moreno 2013] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly and Andrew J Davison. *Slam++: Simultaneous localisation and mapping at the level of objects*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1352–1359, 2013. (Cited on pages 55, 84 and 108.)

[Salvi 2007] Joaquim Salvi, Carles Matabosch, David Fofi and Josep Forest. *A review of recent range image registration methods with accuracy evaluation*. Image and Vision computing, vol. 25, no. 5, pages 578–596, 2007. (Cited on page 81.)

[Schoettle 2017] Brandon Schoettle. *SENSOR FUSION: A COMPARISON OF SENSING CAPABILITIES OF HUMAN DRIVERS AND HIGHLY AUTOMATED VEHICLES*. Technical report, The University of Michigan, 2017. (Cited on page 40.)

[Schreiber 2013] Markus Schreiber, Carsten Knoppel and Uwe Franke. *LaneLoc : Lane Marking based Localization using Highly Accurate Maps*. In IEEE Intelligent Vehicles Symposium (IV), pages 449–454, 2013. (Cited on page 24.)

[Schutz 1998] C. Schutz, T. Jost and H. Hugli. *Multi-feature matching algorithm for free-form 3D surface registration*. Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170), vol. 2, pages 982–984, 1998. (Cited on page 87.)

[Schwarz 2013] Chris Schwarz, Geb Thomas, Kory Nelson, Micheal McCary and Nocholas Schlarnmann. *Towards Autonomous Vehicles*. Technical report, 2013. (Cited on pages 8 and 12.)

[Se-Ho 2012] Lee Se-Ho, Jeong Seong-Gyun, Chung Tae-Young and Kim Chang-Su. *Real-time acquisition and representation of 3D environmental data*. In Signal & Information Processing Association Annual Summit and Conf. (APSIPA ASC), Asia-Pacific, pages 1–5, 2012. (Cited on page 166.)

[Seetharaman 2006] Guna Seetharaman, Arun Lakhotia and Erik Philip Blasch. *Unmanned Vehicles Come of Age: The DARPA Grand Challenge*. Computer, vol. 39, no. 12, pages 26–29, 2006. (Cited on page 7.)

[Segal 2009] Aleksandr Segal, Dirk Haehnel and Sebastian Thrun. *Generalized-ICP*. In Robotics: Science and Systems, volume 5, pages 168–176, 2009. (Cited on pages 84, 89, 90 and 108.)

[Sehgal 2010] Anuj Sehgal, Daniel Cernea and Milena Makaveeva. *Real-time scale invariant 3D range point cloud registration*. In International Conference Image Analysis and Recognition, pages 220–229. Springer, 2010. (Cited on page 108.)

[Serafin 2015] Jacopo Serafin and Giorgio Grisetti. *NICP: Dense normal based point cloud registration*. In IEEE International Conference on Intelligent Robots and Systems, volume 2015-Decem, pages 742–749, 2015. (Cited on pages 83, 84, 108 and 195.)

[Serafin 2016] Jacopo Serafin, Edwin Olson and Giorgio Grisetti. *Fast and Robust 3D Feature Extraction from Sparse Point Clouds*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4105–4112, 2016. (Cited on pages 82, 83, 108 and 111.)

[Shalal 2013] Nagham Shalal, Tobias Low, Cheryl McCarthy and Nigel Hancock. *A preliminary evaluation of vision and laser sensing for tree trunk detection and orchard mapping*. In Australasian Conference on Robotics and Automation, ACRA, pages 2–4, 2013. (Cited on pages 53 and 54.)

[Sharp 2002] GC Sharp, SW Lee and DK Wehe. *ICP registration using invariant features*. IEEE Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pages 90–102, 2002. (Cited on page 87.)

[Siciliano 2009] Bruno Siciliano, Oussama Khatib and Frans Groen. The DARPA Urban Challenge. Springer-Verlag Berlin Heidelberg, 2009. (Cited on page 7.)

[Silva 2005] Luciano Silva, Olga R P Bellon and Kim L. Boyer. *Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 5, pages 762–776, 2005. (Cited on page 90.)

[Smith 1986] Randall C. Smith and Peter Cheeseman. *On the Representation and Estimation of Spatial Uncertainty*. The International Journal of Robotics Research, vol. 5, no. 4, pages 56–68, 1986. (Cited on page 55.)

[Smith 1987] R Smith, M Self and P Cheeseman. *A stochastic map for uncertain spatial relationships*. In Proceedings of the 4th international symposium on Robotics Research, pages 467–474, 1987. (Cited on page 55.)

[Song 2009] Hao Song and Hsi Yung Feng. *A progressive point cloud simplification algorithm with preserved sharp edge data*. Int. Journal of Advanced Manufacturing Technology, vol. 45, no. 5-6, pages 583–592, 2009. (Cited on pages 165 and 167.)

[Steder 2011] B Steder, R B Rusu, K Konolige and W Burgard. *Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries Bastian*. In IEEE International Conference on Robotics and Automation (ICRA),, pages 2601–2608, 2011. (Cited on page 82.)

[Sun 2018] Shaohui Sun, Ramesh Sarukkai, Jack Kwok and Vinay Shet. *Accurate Deep Direct Geo-Localization from Ground Imagery and Phone-Grade GPS*. In arXiv, 2018. (Cited on pages 21 and 39.)

[Tang 2018] Tim Y. Tang, David J. Yoon, François Pomerleau and Timothy D. Barfoot. *Learning a Bias Correction for Lidar-only Motion Estimation*. In arXiv, pages 1–8, 2018. (Cited on page 157.)

[Tao 2013] Zui Tao, Philippe Bonnifait, Vincent Fremont and Javier Ibañez-guzman. *Lane marking aided vehicle localization*. In 6th International IEEE Conference on Intelligent Transportation Systems-(ITSC), pages 1509–1515, 2013. (Cited on page 24.)

[Tardos 2002] J. D. Tardos, J. Neira, P. M. Newman and J. J. Leonard. *Robust Mapping and Localization in Indoor Environments Using Sonar Data*. The International Journal of Robotics Research, vol. 21, no. 4, pages 311–330, 2002. (Cited on page 53.)

[Taylor 2001] G. Taylor, D. Steup, A. Car, G. Blewitt and S. Corbett. *Road reduction filtering for GPS-GIS navigation*. Transactions in GIS, vol. 5, no. 3, pages 193–207, 2001. (Cited on page 178.)

[Tazir 2016] Mohamed Lamine Tazir, Paul Checchin and Laurent Trassoudaine. *Color-based 3D Point Cloud Reduction*. In the 14th International Conference on Control, Automation, Robotics and Vision, ICARCV, pages 1–7, 2016. (Cited on pages 94, 113, 119, 128, 174 and 180.)

[Teoh 2017] Eric R Teoh and David G Kidd. *Rage against the machine ? Google ' s self-driving cars versus human drivers*. Journal of Safety Research, vol. 63, pages 57–60, 2017. (Cited on page 14.)

[Thomson Reuters 2016] Thomson Reuters. *The 2016 State of Self-Driving Automotive Innovation*. Technical report, 2016. (Cited on page 10.)

[Thorpe 1988] C Thorpe, S Shafer, T Kanade, Lab and the Members of the Strategic Computing Vision. *Vision and Navigation for the Carnegie Mellon Navlab.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 3, pages 362–373, 1988. (Cited on page 6.)

[Thrun 1993] Sebastian B. Thrun. *Exploration and model building in mobile robot domains*. IEEE International Conference on Neural Networks - Conference Proceedings, vol. 1993-Janua, pages 175–180, 1993. (Cited on page 55.)

[Thrun 2004] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani and Hugh Durrant-Whyte. *Simultaneous localization and mapping with sparse extended information filters*. International Journal of Robotics Research, vol. 23, no. 7-8, pages 693–716, 2004. (Cited on page 55.)

[Thrun 2007] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge, chapitre 1, pages 1–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. (Cited on page 106.)

[Thrun 2008] Sebastian Thrun. *Simultaneous Localization and Mapping*. In Springer Handbook of Robotics, pages 361–389. 2008. (Cited on page 55.)

[Tibshirani 2001] Robert Tibshirani, Guenther Walther and Trevor Hastie. *Estimating the number of clusters in a data set via the gap statistic*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 63, no. 2, pages 411–423, 2001. (Cited on pages 75 and 113.)

[Tipaldi 2012] G D Tipaldi and D Meyer-Delius. *Lifelong Localization and Dynamic Map Estimation in Changing Environments*. In RSS Workshop on . . . , pages 1–2, 2012. (Cited on page 58.)

[Tipaldi 2014] Gian Diego Tipaldi, Manuel Braun and Kai O. Arras. *FLIRT: Interest regions for 2D range data*. In Springer Tracts in Advanced Robotics, pages 695–710, 2014. (Cited on page 53.)

[Ulrich 2014] Lawrence Ulrich. *Top ten tech cars*. IEEE Spectrum, pages 38–47, 2014. (Cited on page 12.)

[Van Brummelen 2018] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer and Homayoun Najjaran. *Autonomous vehicle perception: The technology of today and tomorrow*. Transportation Research Part C: Emerging Technologies, pages 1–23, 2018. (Cited on pages 3, 9, 10, 11, 12, 14, 19, 21, 24, 31, 45 and 57.)

[Velas 2016] Martin Velas, Michal Spanel and Adam Herout. *Collar Line Segments for fast odometry estimation from Velodyne point clouds*. In IEEE International Conference on Robotics and Automation (ICRA), pages 4486–4495, 2016. (Cited on pages 83, 85, 106, 108 and 184.)

[Vicente 2015] Francisco Vicente, Zehua Huang, Xuehan Xiong, Fernando De La Torre, Wende Zhang and Dan Levi. *Driver Gaze Tracking and Eye Off the Road Detection System*. IEEE Transaction on Intelligent Transportation Systems, vol. 16, no. 4, pages 2014–2027, 2015. (Cited on page 11.)

[Vitor 2014] Giovani Bernardes Vitor. *Perception de l'environnement urbain et navigation s'appuyant sur la vision robotique : La conception et la mise en oeuvre appliquées au véhicule autonome*. PhD thesis, Université de Technologie de Compiègne, 2014. (Cited on page 7.)

[Vitter 1984] Jeffrey Scott Vitter. *Faster Methods for Random Sampling*. Commun. ACM, vol. 27, no. 7, pages 703–718, 7 1984. (Not cited.)

[Vivacqua 2017] Rafael Peixoto Derenzi Vivacqua, Massimo Bertozzi, Pietro Cerri, Felipe Nascimento Martins and Raquel Frizera Vassallo. *Self-Localization Based on Visual Lane Marking Maps: An Accurate Low-Cost Approach for Autonomous Driving*. IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pages 582–597, 2017. (Cited on pages 20, 24, 32, 34, 45 and 57.)

[Walker 1991] Michael W. Walker, Lejun Shao and Richard A. Volz. *Estimating 3-D location parameters using dual number quaternions*. CVGIP: Image Understanding, vol. 54, no. 3, pages 358–367, 1991. (Cited on page 69.)

[Wang 2017] Zhe Wang, Yang Liu, Qinghai Liao, Haoyang Ye, Ming Liu and Lujia Wang. *Characterization of a RS-LiDAR for 3D Perception*. In arXiv, 2017. (Cited on page 38.)

[Weber 2015] T Weber, R Hänsch and O Hellwich. *Automatic registration of unordered point clouds acquired by Kinect sensors using an overlap heuristic*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 102, pages 96–109, 2015. (Cited on pages 83 and 115.)

[White 2000] Christopher E. White, David Bernstein and Alain L. Kornhauser. *Some map matching algorithms for personal navigation assistants*. Transportation Research Part C: Emerging Technologies, vol. 8, no. 1-6, pages 91–108, 2000. (Cited on page 178.)

[Wiemann 2014] Thomas Wiemann, Marcel Mrozinski, Dominik Feldschnieders, Kai Lingemann and Joachim Hertzberg. *Data Handling in Large-Scale Surface Reconstruction*. In 13th International Conference on Intelligent Autonomous Systems, pages 1–12, 2014. (Cited on pages 113, 165 and 170.)

[Will 2017] Devid Will, Lutz Eckstein, Steven Von Bargen, Tessa T Taefi and Roland Galbas. *State of the art analysis for Connected and Automated Driving within the SCOUT project KEYWORDS : 2 . Methodology : State of the Art Analysis for Connected and Automated Driving*. In ITS World Congress, pages 1–8, 2017. (Cited on page 13.)

[Wolcott 2014] Ryan W Wolcott and Ryan M Eustice. *Visual localization within LIDAR maps for automated urban driving*. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 176–183. IEEE, 2014. (Cited on pages 57, 106 and 179.)

[Wolcott 2015] Ryan W. Wolcott and Ryan M. Eustice. *Fast LIDAR localization using multiresolution Gaussian mixture maps*. In Proceedings - IEEE International Conference on Robotics and Automation, pages 2814–2821, 2015. (Cited on pages 31 and 179.)

[Wolcott 2017] Ryan W Wolcott and Ryan M Eustice. *Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving*. The International Journal of Robotics Research, vol. 36, no. 3, pages 292–319, 2017. (Cited on page 179.)

[Wolfson 2004] O. Wolfson and Huabei Yin. *A weight-based map matching method in moving objects databases*. In Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004., pages 437–438, 2004. (Cited on page 178.)

[Xiang 2018] Huaikun Xiang and Antao Ming. *Research on Precise Positioning Algorithm of Intelligent and Connected Vehicles with High-Speed Moving*. In International Conference on Transportation and Development, pages 289–298, 2018. (Cited on page 22.)

[Xiao 2015] Wen Xiao, Bruno Vallet, Mathieu Bredif and Nicolas Paparoditis. *Street environment change detection from mobile laser scanning point clouds*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 107, no. May, pages 38–49, 2015. (Cited on page 58.)

[Yang 2013] Jiaolong Yang, Hongdong Li and Yunde Jia. *Go-icp: Solving 3d registration efficiently and globally optimally*. In Proceedings of the IEEE International Conference on Computer Vision, pages 1457–1464, 2013. (Cited on pages 83 and 108.)

[Yang 2016] Bisheng Yang, Zhen Dong, Fuxun Liang and Yuan Liu. *Automatic registration of large-scale urban scene point clouds based on semantic feature points*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 113, pages 43–58, 2016. (Cited on page 83.)

[Yin 2018] Huan Yin, Yue Wang, Li Tang, Xiaqing Ding and Rong Xiong. *LocNet: Global localization in 3D point clouds for mobile robots*. In arXiv, 2018. (Cited on page 196.)

[Yousif 2016] Khalid Yousif. *3D Simultaneous Localization and Mapping in Texture-less and Structure-less Environments Using Rank Order Statistics*. PhD thesis, 2016. (Cited on page 34.)

[Zeng 2016] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao and T Funkhouser. *3DMatch: Learning the matching of local 3D geometry in range scans*. arXiv, vol. 1603, 2016. (Cited on page 108.)

[Zhang 1992] Z. Zhang. *Iterative point matching for registration of free-form curves*. Technical report, INRIA, 1992. (Cited on page 167.)

[Zhang 1994] Zhengyou Zhang. *Iterative point matching for registration of free-form curves and surfaces*. International journal of computer vision, vol. 13, no. 2, pages 119–152, 1994. (Cited on pages 86, 89 and 115.)

[Zhang 2012] F. Zhang, X. Che and W. Zuo. *A Simplification Algorithm for Point Cloud Data Based on B-spline Curve*. Journal of Computational Information Systems, pages 1821–1828, 2012. (Cited on page 173.)

[Zhang 2014] Ji Zhang and Sanjiv Singh. *LOAM : Lidar Odometry and Mapping in Real-time*. In Robotics: Science and Systems, 2014. (Cited on pages 37 and 145.)

[Zhang 2015] Tianwei Zhang. *Surface Reconstruction with Sparse Point Clouds of Velodyne Sensor*. In The 14th IFToMM World Congress, 2015. (Cited on pages 38, 106 and 146.)

[Zhong 2009] Yu Zhong. *Intrinsic shape signatures: A shape descriptor for 3d object recognition*. In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pages 689–696. IEEE, 2009. (Cited on pages 83 and 108.)

[Zhou 2017] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. (Cited on page 196.)

[Zhou 2018] Qian-Yi Zhou, Jaesik Park and Vladlen Koltun. *Open3D: A Modern Library for 3D Data Processing*, 2018. (Cited on page 195.)

[Ziegler 2014] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G. Keller, Eberhard Kaus, Ralf G. Herrtwich, Clemens Rabe, David Pfeiffer, Frank Lindner, Fridtjof Stein, Friedrich Erbs, Markus Enzweiler, Carsten Knoppel, Jochen Hipp, Martin Haueis, Maximilian Trepte, Carsten Brenk, Andreas Tamke, Mohammad Ghanaat, Markus Braun, Armin Joos, Hans Fritz, Horst Mock, Martin Hein and Eberhard Zeeb. *Making bertha drive-an autonomous journey on a historic route*. IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, pages 8–20, 2014. (Cited on pages 8, 24, 45 and 58.)