Object recognition for locomotion and manipulation with a humanoid robot in an industrial environment

Thibaud Lasguignes

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)*

Présentée et soutenue le *31/08/2023* **par :**

Thibaud LASGUIGNES

Object recognition for locomotion and manipulation with a humanoid robot in an industrial environment

---

JURY

OLIVIER AYCARD	Maître de Conférences, Université Grenoble Alpes	Rapporteur
DAVID FILLIAT	Professeur, ENSTA Paris	Président du Jury et Rapporteur
JEAN-EMMANUEL DESCHAUD	Chargé de Recherche, Mines Paris	Membre du Jury
ARIANE HERBULOT	Maître de Conférences, Université Paul Sabatier	Membre du Jury
NICOLAS MANSARD	Directeur de Recherche, LAAS-CNRS	Membre du Jury
OLIVIER STASSE	Directeur de Recherche, LAAS-CNRS	Directeur de Thèse

---

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes
Directeur de Thèse :
Olivier STASSE
Rapporteurs :
Olivier AYCARD et *David FILLIAT*

Remerciements

Dans un premier temps, toute ma gratitude est dirigée vers Olivier Stasse, mon directeur de thèse, pour sa confiance et son accompagnement au cours de ces quatre années. Merci à lui et à Frédéric Lerasle qui m'ont d'abord accueilli en stage dans le cadre de mon Projet de Fin d'Étude à l'INSA. Merci de m'avoir ensuite proposé de continuer à travers cette thèse, d'avoir toujours trouvé le temps, même si cela était parfois très compliqué, pour m'aider dans mes questionnements.

Je suis reconnaissant à Olivier Aycard et David Filliat pour avoir accepté d'être rapporteurs de mes travaux. Je remercie également Jean-Emmanuel Deschaud, Ariane Herbulot et Nicolas Mansard pour avoir fait partie des membres du Jury. Les retours des rapporteurs, les questions du Jury et leurs retours finaux m'ont permis de faire quelques pas supplémentaires dans mes travaux et m'ont apporté d'autres réflexions pour l'avenir. Plus particulièrement, le soutien d'Ariane et Nicolas dans le cadre de mon Comité de Suivi a été primordial pour avancer personnellement et professionnellement, dans une période difficile. Je me dois de dire que sans leur aide, ces pages n'auraient peut-être pas vu le jour.

Je tiens à évoquer aussi tous ceux qui m'ont côtoyé le long de ce parcours : les enseignants du DUT GEII à Paul Sabatier qui m'ont aidé à maîtriser une certaine technicité, ceux de l'INSA qui m'ont ouvert à des théories plus complexes et à d'autres horizons, ceux qui m'ont accompagné sur mes premiers et seconds pas dans l'enseignement, qu'ils soient de l'INSA ou de Paul Sabatier, sans oublier les étudiants avec lesquels j'ai partagé ces années. Un grand merci à tous, et notamment à Amaury, Cyril, Torkel, Simon et toute l'équipe du TIM.

Ce travail n'aurait pas abouti sans toute l'équipe Gepetto: les anciens, qui m'ont accueilli chaleureusement et m'ont très rapidement intégré et ceux qui nous ont rejoints en cours de route. Ce fut toujours une joie de vous rencontrer, parfois avec un zeste de surprise, au détour de mon café matinal. J'ai essayé de recevoir à mon tour avec autant de chaleur. Même si certaines traditions se sont un peu perdues en route, la bonne ambiance est toujours au rendez-vous. En bref, un grand merci pour l'entrain apporté au bureau mais aussi dans d'autres occasions, autour d'une simple discussion de couloir, de gâteaux, de bières[1] ou de jeux. Je ne pourrais pas tous vous citer, il me faudrait beaucoup trop de lignes, mais je n'oublierai aucun de vous.

Une pensée particulière pour les membres de mon bureau, que ce soit le B65 ou l'Algeco A2. Mon aventure en doctorat a commencé dans le B65 avec Thomas Flayols, bientôt rejoint par Louise Scherrer et Nils Harreng : merci pour votre humour et votre gaité, j'en ai conservé une part après votre départ vers d'autres horizons. Un grand merci à mon homonyme Thibault Marsan[2] et mon prédécesseur Maximilien Naveau. Les membres

[1] L'abus d'alcool est dangereux pour la santé, à consommer avec modération mais surtout bien entouré !

[2] Même s'il ne sait pas écrire correctement notre prénom...

ont changé, la gaieté et le partage sont restés même quand nous avons été séparés. Merci à Nahuel Villa pour le court mais olfactivement dépaysant partage de l'Algeco A2.

Je pense aussi à Hugo Lefèvre, Guillaume Gobin, Dorian Baudu et Weikang Zeng avec qui j'ai eu l'occasion de travailler pendant leurs stages chez nous.

Depuis plusieurs années, ils m'aident à évacuer mon stress : les coachs du Club de Tri à l'arc de Balma qui n'ont pas lâché l'affaire, même quand j'étais moins assidu, les archers de Balma, de l'INSA, de Paul Sabatier ou d'autres clubs : merci à vous pour ce qu'on a partagé sur un pas de tir et ailleurs. Un merci tout particulier pour Morgane Castell qui a partagé un bout de chemin avec moi dans une période difficile et qui ne reste jamais loin...

Enfin, parce que je n'aurais pas fait tout cela s'ils n'avaient pas été là, un grand merci à ma famille, à mes parents, ma sœur et mon frère pour m'avoir accompagné depuis la première heure et m'avoir poussé à toujours aller plus loin. Je sais que je peux compter sur vous et que cela va continuer.

Encore merci à tous, et en route vers de prochaines aventures !

Contents

CHAPTER 1

Introduction

1.1 Context

1.1.1 An industrial environment

This work is done in the context of a partnership with Airbus to enable the robots to perceive it's environment in aircraft factories. In these factories, automation with robots bring many challenges. This is mainly due to the complexity of the tasks and the accuracy needed. The manufacturing process of an aircraft is originally designed to be done by humans. Due to the relatively slow evolution of the process, when compared to the automotive industry, automating the process requires to adapt to it rather than redefining it's steps.

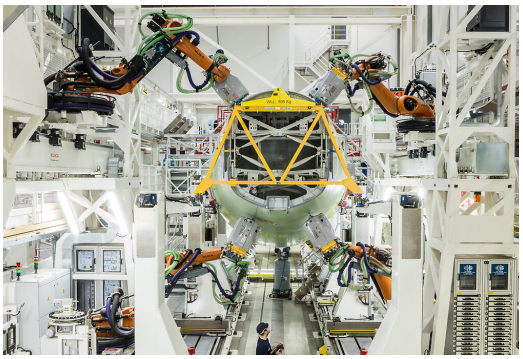
One of the main process realized in aircraft manufacturing is the drilling of holes. According to [Blac 19], Airbus drills 120 millions of holes per year in their aircraft factories, with 75% of these are manually done. Several attempts have been made in order to automatize this process.

In 2014, Boeing announced working with the german firm KUKA Systems to automate the assembly of the fuselage of their 777 series [Gate 19]. This automation was called "Fuselage Automated Upright Build" (FAUB) and composed of robotic arms, shown in Figure 1.1. The system has been deployed in the assembly line of the Everett site starting in 2015. It was composed of robotic arms working in pair from both the inside and outside of the fuselage to drill holes and place fasteners. However, after 6 years and millions of dollars spent on the development of the assembly line, the solution has been dropped. Boeing had struggle in having both robots from a pair working in sync. These struggles resulted in the need for human workers to fix the mismatches or completing part of the robot works. With the failure of the FAUB solution, Boeing decided to revert to a solution based on "Flextrack" robots, that were already used on the assembly of other parts.

In 2019, Airbus inaugurated a new assembly line in it's factory of Hamburg for the A320 structure [Airb 19]. This new assembly line integrates "Flextrack" robots and Kuka arms humanly operated, shown in Figure 1.2. These robots are able to drill and countersink holes and to insert rivets with an accuracy of 0.2 millimeters. However, the robots used in this assembly line are only able to operate from the outside of the fuselage, while human workers are still needed inside.

This process includes complex tasks that require high level of control and accuracy both in the final pose of the end effector and in the force applied to aircraft parts. In addition, fuselage parts are composed of large metal plates with a thickness of less than 2 centimeters. These plates are therefore flexible and require careful control of the forces applied during drilling or deburring. This is why the robots need to be controlled in torque, which is more challenging than position control and may lack accuracy.

Figure 1.1: Robots of the assembly line of Boeing, from [Gate 19].



(a) KUKA arms outside the fuselage

(b) “Flextrack” robot on the fuselage

Figure 1.2: Robots of the assembly line of Airbus, from [Airb 19].

Furthermore, aircraft factories are made of large workshops. These workshops often present texture-less structures and objects. Moreover, the plane structures, on which the robots are intended to work, are wide and raw metal structures. Therefore, in such environments, it is needed to be able to localize structures and objects of diverse sizes, going from a small tool to a fuselage part, while relying on poor textures.

Operating robots inside of the fuselage needs them to be more versatile and dexterous. Indeed, the tasks to perform need a high accuracy and the environment in which they evolve is going from a simple warehouse to the complex inner structure of a fuselage. This necessities and the fact that the process is designed to be performed by humans drove to the idea of using humanoid robots for such tasks. While working on humanoid robots instead of robotics arms increases the complexity of the control problem, it firstly aims at being adaptative to the environment that may not be suitable for a robotic arms on a fixed or mobile base. Moreover, it regroups the interest of multiple laboratories that allows to access the knowledge of experts in domains that may intersect while trying to perform

a task. Indeed, the industrial applications in aircraft manufacturing is not the only one needing to have precise and compliant control.

Moreover, with nowadays technologies and processes, each step of aircraft manufacturing can not be automated. Thus, physical interactions between the robots and humans workers stay possible and the control scheme applied on the robots must ensures the safety of people.

1.1.2 ROBotics for the Future of Aircraft Manufacturing

ROB4FAM is a joint laboratory between the Laboratory For Analysis and Architecture of Systems (LAAS-CNRS) and Airbus Operations inaugurated in 2019. Airbus Operations is a part of the Airbus Group working on aircraft factories.

The goal of the joint laboratory is to develop and integrate reactive methods on industrial robots in the process of aeronautic constructions. In this objective, the robot is wanted to process autonomously while it may interact with human workers. Thus, the robots needs to be able to perform complex tasks, such as climbing stairs or drilling, with precision and react safely to interactions with it's environments and surrounding humans. Therefore, it needs to be able to perceive its surroundings and extract useful informations.

A first axis of the project aims to perform reactive motion planning with considering uncertainties in the perception of the environment or in the model of the robots actuators [Nico 20]. For example, demonstrations of the online planning were performed with the robots Tiago and Talos. Tiago is moving a tool close to an airplane mast, aligning it with subsequent holes [Mira 21]. On another side, Talos is using both it's grippers to flip a wooden piece [Mira 20]. These two demonstrations show the capabilities of planning a motion online to perform a given task. However, while executing a planned motion, diverse uncertainties need to be taken into account. Visual Servoing has been used to track the objects involved in the manipulation. Both robots are localizing their objects thanks to a set of ArUco tags [Garr 14]. Thus this thesis aims to remove this dependency to ArUco markers by enabling the robot to localize objects of interests.

The second part concerns the development of a robot controller based on torque and forces measurement to enhance the efficiency of the task and the safety of the robot near human workers [Ramu 21, Ramu 22b, Ramu 22c].

The third part of the project, addressed in this thesis, is to give perception capabilities to the robot. Part of the objective is to provide a way for the robot to localize itself in a large environment [Lasg 21]. Moreover, the robot needs to localize itself relatively to objects of interests, such as the aircraft pylon in the demonstration of Talos performing the deburring task.

A fourth section of the project aims to address the problem of keeping balance while moving or performing tasks. In this part, the robot is expected to not only keep balance but evolve in complex environment with e.g. stairs. Thus, perception capabilities are again needed to determine available planes for contacts.

This projects has led to two defended CIFRE theses [Ramu 22a, Nico 22], two academic theses, including the present one and one relying on this work, and several engineering and post-doc positions.

1.1.3 Memory Of Motion

The project Memmo is supported by European Union within the Horizon 2020 Program under Grant Agreement No. 780684. This project is based on the combination of expertises related to motion planning, optimization, machine learning and robotics. Starting in January 2018, it involves multiple european laboratories, universities and companies. This thesis is carried out in particular with the support of PAL-Robotics (Spain), the University of Oxford (UK) and Airbus (France). As in the ROB4FAM project, Airbus provides the targeted application and environment. PAL-Robotics is building the Talos humanoid robot, visible on the bottom left of Figure 1.3. The University of Oxford develops solutions to perceive the environment and plan motions over it which are tested on the ANYmal robot, visible on the top right of Figure 1.3.

Figure 1.3: Illustration of the robots involved in Memmo with Atalante, ANYmal, Talos and SOLO.

The aim is to develop new methods of generating complex movements independently of the robot architecture. This project aims to generate complex movements for legged robots with three main steps. First, it relies on a pre-computation of optimal motions made offline. Second, it aims to use this offline knowledge and adapt to new situations in real-time. Third, it aims to use all sensors modalities available to feed the control with more information than only the robot state.

This thesis integrate in the third axis for the need of enhancing the use of exteroceptive sensors. However, the need of perception capabilities is also present in the idea of using pre-computed knowledge in real situations. Indeed, to adapt known reactions to a new situation, the robot needs to have data on this new environment that it's facing.

In the project, industrial demonstrations are specifically targeted and defined by the end-users involved: the usability in aircraft manufacturing, rehabilitation of paraplegic

patients and the inspection of wide engineering environments. The platforms involved are quadrupeds, an humanoid robot and an exoskeleton as shown in Figure 1.3.

This project is a laureate of the Stars of Europe 2022 for the collaboration of 11 European partners and the demonstrated results.

1.2 Scientific problem

Nowadays, robots are expected to evolve in complex environments while performing tasks ranging from the simplest inspection of places to more challenging precise manipulation. The Gepetto team at LAAS-CNRS, in which this work integrates, aims to develop the control scheme of robots to perform tasks with an *a priori* knowledge of the environment and the tasks while being reactive and adaptative to the perturbations and uncertainties.

To demonstrate advances in robotics, the Defense Advanced Research Projects Agency (DARPA) has organized multiple challenges. One of which is the DARPA Robotic Challenge (DRC) [DARP] that has been organized to demonstrate the control of semi-autonomous robots. This challenge aims to resolve tasks that present the same needs as the Gepetto team's, but in a different environment. In the DRC, the robots were expected to perform complex tasks in "dangerous, degraded, human-engineered environments".

In most of these situations, in order to progress in its tasks, the robot needs to adapt to its surroundings. Therefore, perception is necessary in robotics field and is addressed by many teams. This perception problem can be segmented in four main axis:

- Perception of the manipulation space
- Estimating the motion
- Localization in the environment
- Detection of the floor surfaces and obstacles

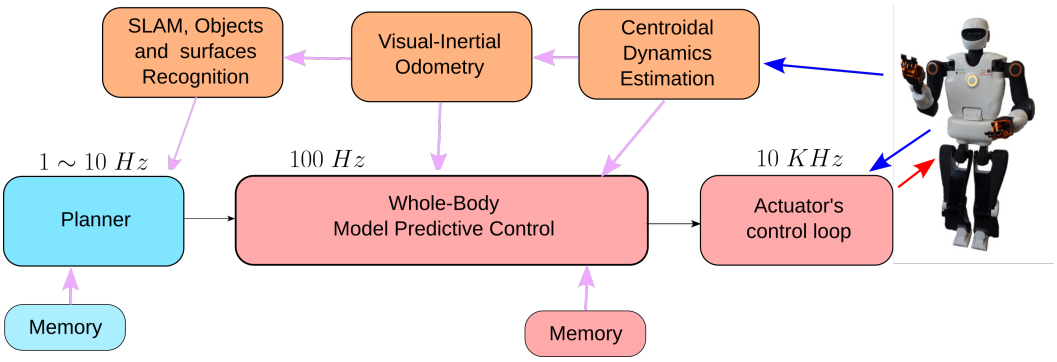


Figure 1.4: Targeted control scheme in the Gepetto team at LAAS-CNRS.

Figure 1.4 resumes the control scheme targeted in the robots of the Gepetto team with the interaction of the perception systems and the control. The Planner bloc defines the sequence of contact to realise in order to produce a targeted motion, based on the knowledge

of the task to do and the environment. This sequence of contact is then provided to the Whole-Body Predictive Control which defines the control to apply at each joint in order to realise the motion. This bloc is helped by state estimators, whether using proprioceptive or exteroceptive sensors, to ensure the safety and accuracy of the motion. Then, the control is provided to a low-level controller for each actuator. The lower is the block level, the higher the frequency it will run at. Thus, the planner is targeted at 10Hz while the low-level controller is running at $10k$Hz.

1.2.1 "Look what you do"

This perception problem in manipulation tasks is in two phases. The first part of the problem is that for the robot to operate a tool, it needs to acquire an estimation of the tool's pose relative to himself. Considering the robot space $\mathscr{R}$, this problem can be resumed as determining the pose ${}^{R}p_{obj}$ of an object in the robot space. Figure 1.5 illustrates this problem with the robot Tiago in front of an object it has to work on.

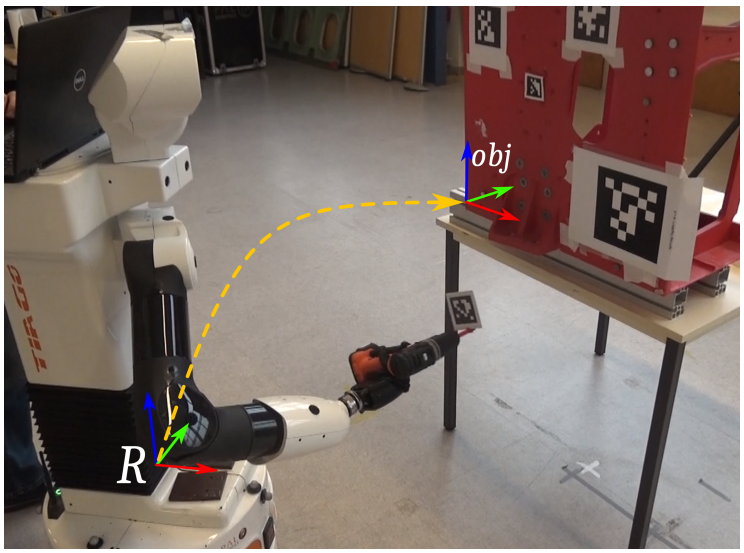
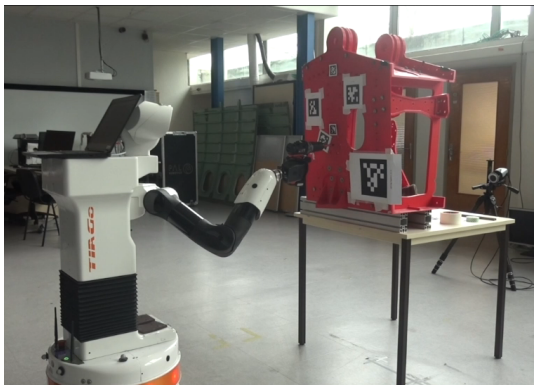
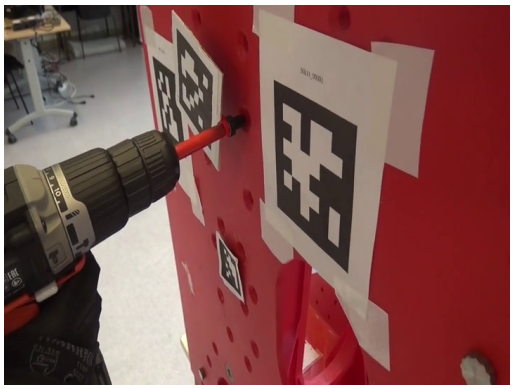


Figure 1.5: Problem of estimating the pose of an object to be used by a robot, here Tiago in front of the mast.

Taking as example demonstrations of control scheme made at LAAS-CNRS with Tiago, in Figure 1.6, or Talos, in Figure 1.7, aligning a tool with holes. In these experiments, the perception problem is resolved by off the shelf methods. In the demonstration with Tiago, ArUco tags, visible on the Figure 1.6b, are used to locate the mast and the tool. These tags enhance the information perceptible and are provided with a system that accurately detect and localize them. These tags were chosen to estimate the pose of the objects for the accuracy of the detected positions. On the other hand, they present potential errors when estimating the orientation due to singularities. However, using multiple tags for an object and comparing their estimations can resolve these singularities. In the demonstration with Tiago, the informations acquired by estimating the tags poses are fused thanks to the Visual Servoing Platform (ViSP).



(a) Tiago in front of the airplane mast



(b) The tool aligned with a hole

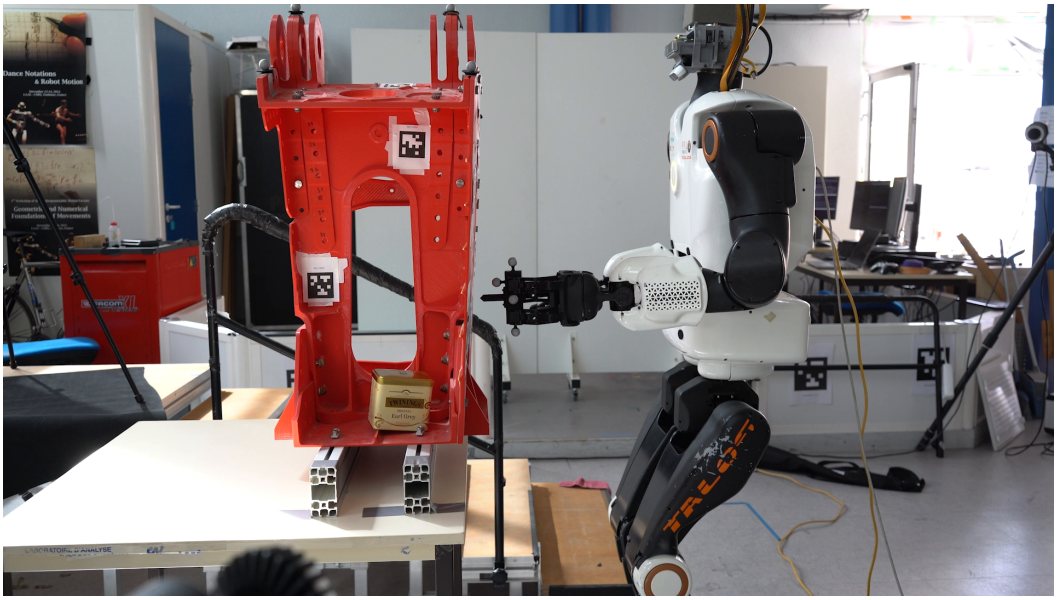
Figure 1.6: The deburring demonstration of an airplane mast with Tiago at LAAS-CNRS.

In the demonstration with Talos, Figure 1.7 from the work of Côme Perrot, a Motion Capture system using multiple infra-red cameras is used. This Motion Capture system is used to localize small reflecting balls, visible on the tool of Talos and on the mast corners.

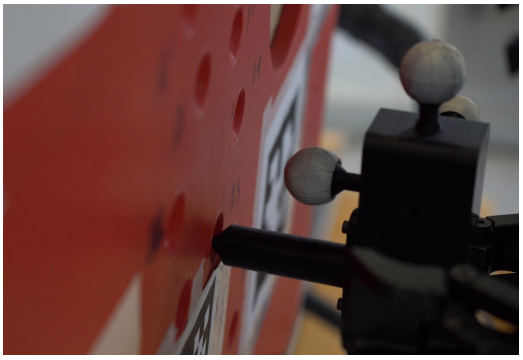
These demonstrations show that with accurate perception, accurate control can be achieved. Figure 1.6b and 1.7b show the positioning accuracy of both robots using either the ArUco or the Motion Capture. These demonstrations show that the robots have a capable control if the perception information is accurate.

The second part of the perception problem in manipulation tasks is that for better control during the motion, the estimation of the robot state may be needed. This is due to the flexibility and backlashes present in the joints. For the estimation of the robot state, introspection sensors such as Inertial Measurement Units (IMU), encoders or force sensors, are often used. However, these sensors may present noisy data and do not take account of all external perturbations. Thus, using these sensors provide the robot with a short-term approximation. Therefore, the use of exteroceptive sensors such as cameras is often discussed. In the deburring demonstration using Tiago, the robot estimates the tool pose thanks to the tag and corrects the control to achieve an accurate alignment with a few millimeters of precision in the orthogonal plane. On the other side, while using Talos, the control scheme relies on the pose acquired by the Motion Capture to determine the error between the tool's pose and the mast holes. In this work, the estimation of the robot state in manipulation tasks will not be discussed.

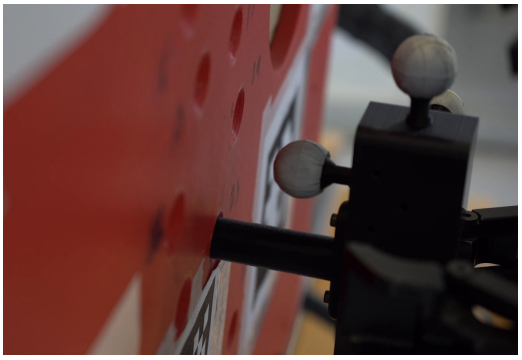
Lately, this demonstration has been deployed with an UR10 robotic arm [Lami 22] (Figure 1.8) in the framework of REACT, an extension of the ROB4FAM project. The aim of this extension is to adapt the deburring demonstration with Tiago to a demonstration that implies an UR10 from Universal Robots [Univ]. The UR10 robot, equipped with an Intel®RealSense™D435i camera, localizes holes in the plank before him and plan a motion to bring it's tool, represented by a wooden stick, to the holes. In this version, the perception part is performed using ViSP without using any external markers, as in the demonstration of Tiago, or any external tracking system, as in the demonstration of Talos. However, for the tracking of the object, ViSP is dependent of the initial pose provided.



(a) Talos in front of the airplane mast



(b) The tool aligned with a hole



(c) The tool pushed in a hole

Figure 1.7: The deburring demonstration of an airplane mast with Talos at LAAS-CNRS, from the work of C. Perrot.

These demonstrations show that the planning systems developed in the Gepetto Team can be deployed on multiple platforms with changing architectures and varying complexities like Tiago, Talos or the UR10. The common point of these demonstration is the need of a perception system to perceive the environment in order to plan the motion.

(a) In front of the plank (b) Closing the distance to the hole (c) Aligned to the hole

Figure 1.8: The UR10 demonstration of the motion planning, from [Lami 22].

In this thesis, we will only be interested in the localization of the necessary objects. This thesis aims at performing this localization without the need of a system extern to the robot. The challenge in solving this problem is to obtain a pose precise enough to perform the manipulation problem or to warm-start a tracking system such as ViSP.

1.2.2 "Look where you go"

Balancing and estimating the motion is of interest for roboticists to ensure the robot motion. The encoders and IMUs are used to estimate the motion of the robot. As an example, [Flay 17] benchmark base estimators on the HRP-2 humanoid robot. But due to their lack of external referees, they are limited to short time estimation.

Thus, to enhance the estimation of the odometry, visual system is often used at slower frequency. The principle of Visual Odometry is to estimate visual features in subsequent images and extrapolate the motion of the camera by estimating the features' motion. Figure 1.9, from [Scar 11], illustrates the Visual Odometry problem in the case of a system composed of two cameras.

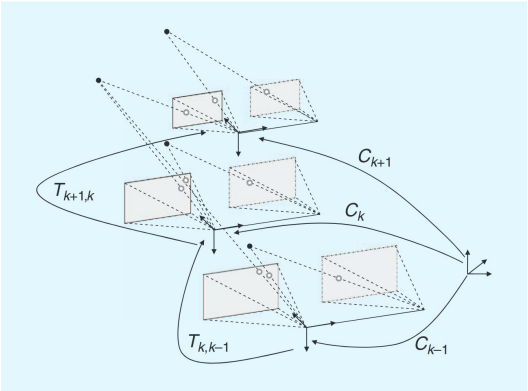


Figure 1.9: Illustration of the Visual Odometry problem with a system composed of two cameras, from [Scar 11].

The Visual Odometry problem consist of estimating the motion $T_{k+1,k}$ by matching the features in images $k+1$ and k. Moreover, these relative transformations between two subsequent images can be integrated to estimate the pose of the system, noted C_k in Figure 1.9, relatively to the initial pose of the system. Visual Odometry may relies on proprioceptive systems to warm-start the transformation estimation. For example, the Intel®RealSense™T265 tracking camera [Inte c], presented later, performs Visual-Inertial Odometry (VIO). In VIO, the motion estimation from an IMU is used as an initial estimate for the Visual Odometry system.

Simultaneous Localization And Mapping (SLAM) systems try to solve the localization problem, as Visual Odometry, alongside with the mapping problem. It aims, as it's name implies, at localizing the system in its environment while simultaneously keeping track of landmarks encountered to help in the localization solution. These landmarks are relevant elements detected in the images that can for example be visual features, as in ORB-SLAM [Mur- 15], or objects as in CosySLAM [Debe 21]. In the Gepetto team, the SLAM problem has also been addressed in [Four 22] in multiple fashions, including Visual-Inertial Odometry with object-level landmarks.

1.2.3 "Look where you are"

Place recognition is a capacity widely looked over to help vision-based navigation systems or robots to determine whether it is in a known place. It can be achieved in comparing the perceived environment with a representation of the known environment in order to determine if the system is in a place already visited. Moreover, in case the place is known, a refinement of the problem is to estimate the pose of the robot in this environment.

This problem has two main interests.

In SLAM systems, place recognition is used to perform loop closure detection and reduce the localization error. Indeed, while performing SLAM on long experiments or motions, the system may come back to an already visited place. However, SLAM systems, as based on biased sensors, are not perfect and will also show drift over the time. This drift can be corrected by detecting if the system goes by an already known place. Detecting the loop, the system can correct the error accumulated since the last time is came through the loop.

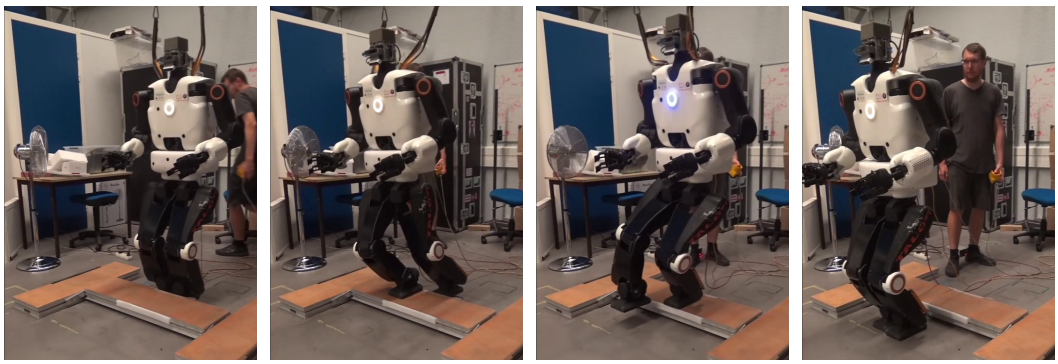
In order for a robot to be autonomous, it might be needed to determine it's pose in the environment at the starting point. This is related to the "kidnapped robot" problem where the robot is randomly moved to an arbitrary place, whether due to a localization error or to an external intervention. While working autonomously in a known workshop, the robot needs to know where it is when it first starts it's self-estimation algorithm.

1.2.4 "Watch your feet"

Another difficulty in legged robots applications is the definition of the contacts. This kind of robots, either being quadrupeds or bipeds, needs to know the possibilities to make support contacts.

At LAAS-CNRS, the robots are demonstrated as able to climb stairs. For example,

Talos is able to take the step shown in Figure 1.10 with whole body control, from the work of Ewen Dantec published in [Dant 22]. However, in this demonstrations, the robots needs to be placed in front of the stair to climb with a tolerance on the position of about 10 centimeters and on the orientation of about 10°. These tolerances have not been studied precisely as they are not the objective of their work.



(a) Talos in front of the step (b) Talos steping up (c) Talos steping down (d) Talos after the staircase

Figure 1.10: Talos going up and down a step.

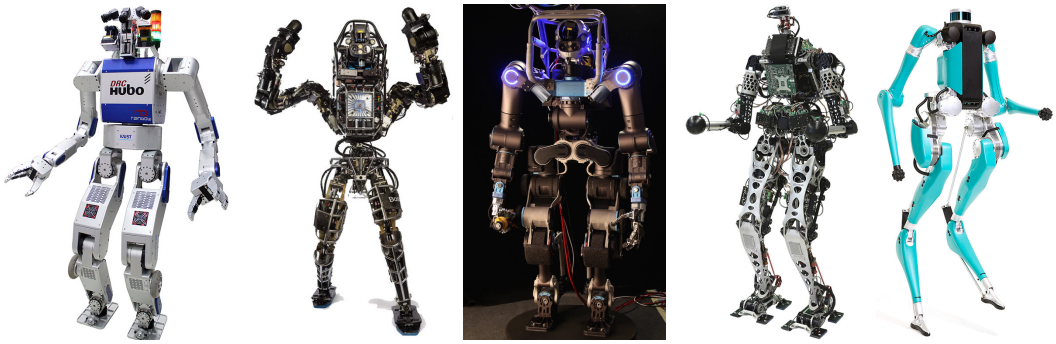
The control scheme, developed in [Dant 22], has not been tested yet on multiple stairs. But the team has already performed stairs climbing with the HRP-2 robot with a position in the environment. Moreover, in [Fern 20], the crossing of a complex environment has also been demonstrated in a motion planning point of view. In both cases, the objective is to demonstrate the capacity to plan the motion and the sequence of contact, with the knowledge of the environment considered perfect. Therefore, while experimenting, the accuracy needed for the initial position and orientation of the robot has been reduced to 2 centimeters. Indeed, these complex environments need precise motion of the feet to be crossed. Thus, to avoid this initial accuracy and to make the robots more autonomous, the feedback from the environment is needed to plan contacts.

1.3 Related works

This part will address the state of the art on humanoid robotics, in particular on the interaction with the environment and it's perception. A more focused state of the art will be presented in each chapter.

The DRC has driven multiple progress in tackling the perception problem for robots. Whereas it took place a few years ago, in 2013 and 2015, it has been a important step in the development of humanoid robots. Indeed, to progress in the challenge, the robot had to localize itself while performing specific tasks. These tasks were demonstrating locomotion problem with the need of going through uneven or congested areas. Moreover, the manipulation ability of the robots was put to the test with tasks of grasping tools, using

tools or valves. Figure 1.12 presents the DRC-HUBO+ robot performing the tasks of the DRC Finals, from [Lim 17].



(a) DRC-HUBO+, 170*cm* (b) Atlas (2013), 188*cm* (c) WALK-MAN, 191*cm* (d) LOLA, 176*cm* (e) Digit, 155*cm*

Figure 1.11: Example of existing humanoid robots.

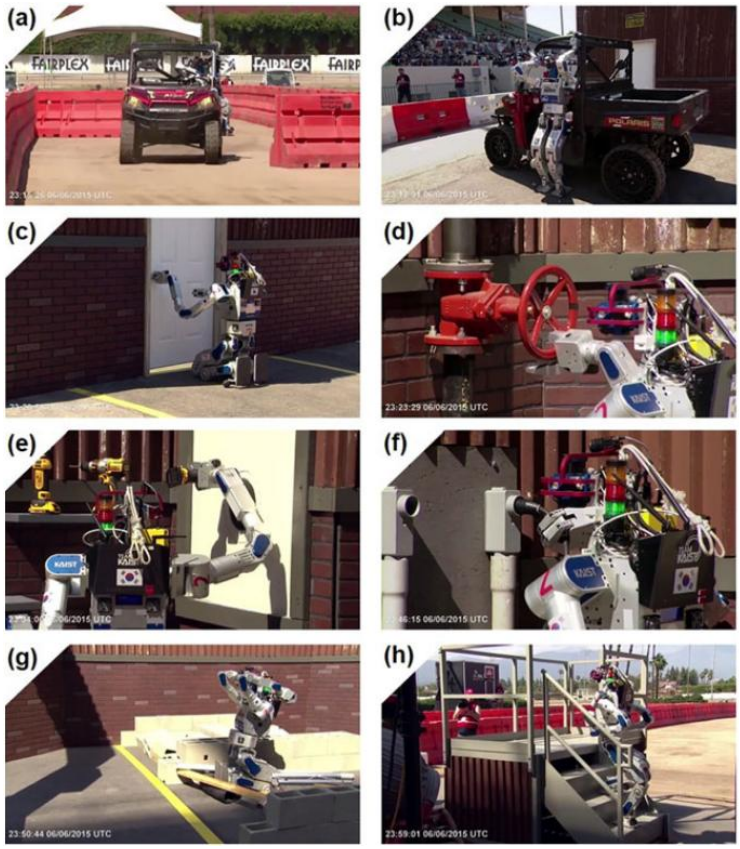


Figure 1.12: Tasks to realise at the DRC: Driving (a), Exiting vehicle (b), Opening a door (c), Opening a valve (d), Cutting a wall (e), a surprise task (f), Going through rubbles (g) and climbing stairs (h); from [Lim 17].

The winning robot DRC-HUBO+, from the Korea Advanced Institute of Science and Technology (KAIST), combined the mobility and dexterity of the humanoid robot with the drivability of the mobile base. The technics used to deal with the diverse perception problem of the DRC are presented in [Lim 17]. To accompany this uncommon but efficient mechanical architecture, the robot is equipped to perceive it's environment with 2 cameras and a LiDAR placed on it's head. These sensors' data are processed to determine a Region Of Interest (ROI) and then perform objects poses estimation or surface estimation. For the estimation of surfaces, the ROI is uniformly divided in segments which are then grouped by their respective normals to reconstruct global surfaces. For the valve manipulation, the closest segment is extracted from the ROI and the parts of the valve are then estimated with the assumption that the handle part is similar to "thin doughnut" with an uncertain number of spokes. Last but not least, to estimate the pose of a tool, the targeted drill is modeled as a rectangle square and a cylinder. From a ROI estimated in the image, the structural planes of the wall and the shelf are estimated separately. Then the points considered from the drill are classified and a circle is fitted to the handle part whereas the base part is fitted to two orthogonal lines to estimate the final pose.

Others relevant robots are the work from the Team IHMC [John 17] and the Team MIT [Fall 15a], both using the Atlas robot from Boston Dynamics, an hydraulically powered robot. Particularly, this thesis discusses in Chapter 6 the integration on Talos of the plane segmentation used by the Team MIT for the DRC. In their published work, the MIT Team presents the calibration applied to the sensors for the DRC Trials. Indeed, the robot was equipped with a pair of stereo cameras and a spinning LiDAR on the head, alongside a pair of wide-FOV cameras on the chest of the robot. For the Trials, their robot was human assisted for the affordance estimation. The human used an interface to provide a rough morphology, location and orientation of each object and an optimisation algorithm then optimise the estimation. During the challenge, WALK-MAN [Tsag 17], has also demonstrated it's locomotion and manipulation capabilities. WALK-MAN is equipped with a Multisense M7 sensor set, composed of a stereo vision system, an IMU and a LiDAR. For rapid estimation of the object poses, generic state of the art method were used, as an Euclidean distance segmentation [Trev 13] or a simple plane removal step. Over these filtered data, the RANSAC method [Fisc 81] was used for simple model fitting for the valve detection whereas geometric feature comparison is used for model fitting of more complex objects.

Another relevant humanoid robot is LOLA v1.1 of the University of Munich. In [Seiw 21], the authors presented an upgrade of the robot with an evolution of the perception. Their work on the perception problem is particularly shown in [Wahr 19] where they deal with the reconstruction of the environment to allows autonomous walking of the robot. This reconstruction is performed by classifying the perceived environment into walkable planes or obstacles. Moreover, a scene reconstruction is performed with an object extraction and a semantic completion, presented in [Wu 20]. The main objective of their work is to perform the scene reconstruction and semantic completion online and not as an offline optimization. The scene is represented as a volumetric occupancy map consisting of empty, unknown or occupied voxels. Each measured depth frame is fed to a neural network to complete the occupancy map. Sub-maps are then filtered out of the occupancy map. The

sub-maps which have been modified by the depth frame processing go through a classifier network to complete the representation in a semantic fashion. Lastly, a SLAM algorithm is integrated using the Iterative Closest Point method presented in KinectFusion [Newc 11].

Humanoids robots may also be inspired by other natural species. As an example, the Digit robot [Hurs 19] from Agility Robotics is based on the Cassie walker. The mechanical architecture of these robots is inspired from birds that are able to run, such as ostriches. However, their manipulation capabilities are limited as they are not equipped with grippers. The Digit robot is equipped to perceive it's environment and perform manipulation tasks. Its capabilities have for example been demonstrated in the last ProMat2023 exhibition during a fully autonomous demonstration [Agil 23].

1.4 Hardware Context: the Humanoid Robot Talos

1.4.1 A demonstrative platform

TALOS is an humanoid robot developed by the PAL-Robotics following the specifications made by the Gepetto team at LAAS-CNRS. The specifications and results are presented in [Stas 17]. This robot succeed to the HRP-2 robot in the team experiments. The goal of TALOS' development is to create a robot able to perform complex locomotion and bi-handed manipulation while involving high payloads. This robot is 1.75m high with a weight of around 100kg. Figure 1.13 shows TALOS and how it's 32 degrees of freedom are distributed.

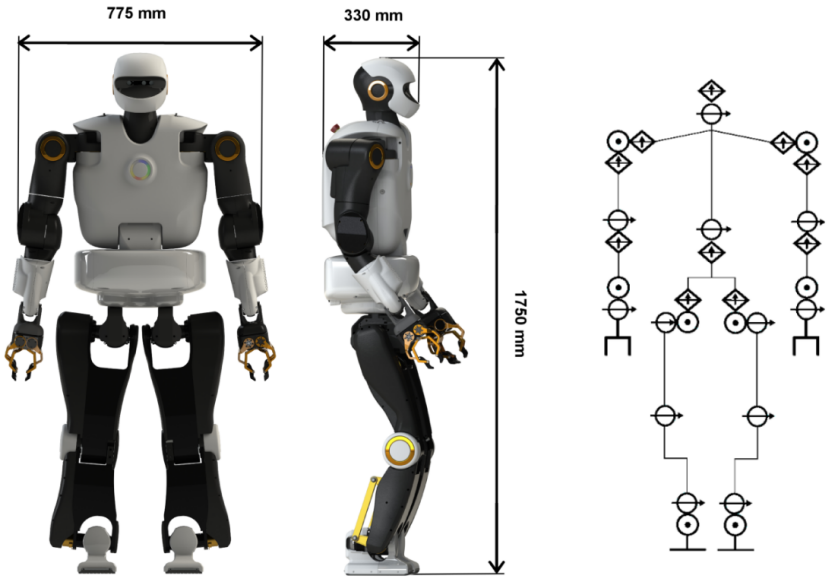


Figure 1.13: The TALOS humanoid robot and its joint architecture.

1.4.2 The evolution of it's perceptive capacities

In the context of enhancing the ability of the robot to localize it's environment, a new head as been developed to use new sensors. This new head integrates a LiDAR with a wide field of view, a fisheye-based stereo camera and an infrared-based RGB-D camera. Figure 1.14 shows the testing head and it's three new sensors presented bellow.

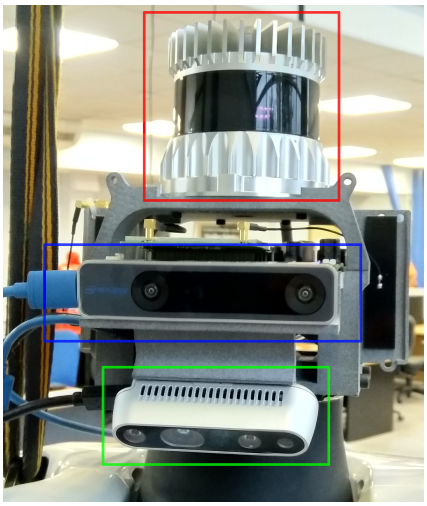


Figure 1.14: The new head of Pyrene with the LiDAR framed in red, the fisheye-based stereo camera in blue and the infrared-based RGB-D camera in green.

1.4.2.1 "Look what you do"

The new head contains an RGB-D camera, the Intel®RealSense™D435i depth camera [Inte a]. This active camera combines an RGB monocular camera and an infrared projector alongside two infrared imagers.

The objective of this camera is to observe the manipulation workspace of the robot. For that, it has been placed in the bottom, with an inclination of 45°. This is done to avoid moving the head to observe the workspace of the robot and thus change the perception angle of the tracking camera and the LiDAR presented latter.

It is defined to be able to look with accuracy at a workbench or any other object within handling range. Thus, it intends to replace the previous head's camera of Talos for the diverse manipulation performed.

1.4.2.2 "Look where you go"

Upside of the RGB-D camera, an Intel®RealSense™T265 tracking camera [Inte c] has been added. This camera uses two greyscale fish-eye lenses in a Stereo way. Thanks to the stereo vision, it may be able to provide a depth measurement of the scene, however the most important feature of this camera is the embedded Vision Processing Unit (VPU). This VPU integrates a Visual-Inertial Odometry able to follow the robot movements with low

drift and high frequency. This VIO system is specifically used to warm-start the LiDAR-based localization system presented in Chapter 2.

1.4.2.3 "Look where you are"

As a hat, the robot has an Ouster OS1-64 LiDAR [Oust]. This LiDAR has a 360° horizontal Field Of View (FOV) and a 33.2° vertical FOV. It's position on the top of the head and it's vertical FOV do not permits to look directly at the floor behind him. However, it's 360° horizontal FOV and 120m range make it able to perceive a huge part of its environment as shown by Figure 1.15.

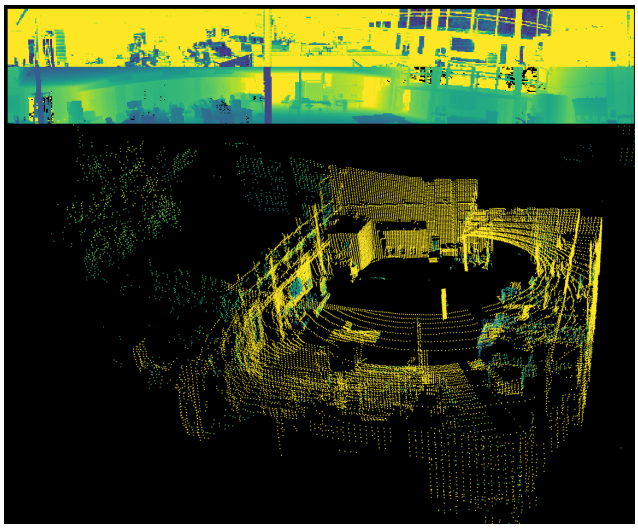


Figure 1.15: View of the Bauzil room at LAAS-CNRS from the LiDAR handheld. Bottom is a point cloud representation colored by the intensity level, top is the intensity image reconstructed, obtained with Ouster's Software Development Kit.

This LiDAR is the main sensor used in this thesis. It is used in Chapter 2 to perform an ICP-based localization, in Chapter 4 to resolve the place recognition problem in order to initialize the localization system. The use of data acquired by this LiDAR is studied to perform long range object localization in Chapter 3.

1.5 Objectives and Contributions

This thesis is an academic work carried out in the framework of the MEMMO and ROB4FAM projects. Both projects seek to allow robots to interact with their environment. For this to be possible, the robot needs to recognise its environment and to locate what it need for its tasks.

Thus, the goal of this thesis is to evaluate and integrate solutions allowing the robot to use external informations for its tasks along the 4 previously presented axis:

- Estimating the motion

- localizing the robot in a known environment
- localizing known objects for manipulation or locomotion tasks
- Estimating contact surface for locomotion tasks

This thesis is particularly oriented on the use of LiDAR technologies within the Memmo project and the industrial context. Indeed, industrial environments such as aircraft factories are wide places with more geometric information than texture. The LiDAR technologies are supposed to be the best solutions for these problems.

The problems addressed in this work contain the self-localization of the robot into the environment to perform locomotion tasks (Chapter 2) and the initial localization of the robot (Chapter 4). The main problematic in this thesis is the pose estimation of objects for locomotion or manipulation tasks by an humanoid robot. This problematic is studied with the use of geometric information for long distance pose estimation in Chapter 3. Moreover, learnt solutions for object pose estimation are discussed and evaluated in Chapter 5. Alongside these thematics, the integration of a LiDAR-based camera to allow performing online contact planning on unknown environment with the humanoid robot Talos is studied in Chapter 6.

The work achieved in this thesis as been published in:

- [Lasg 21] **Thibaud Lasguignes**, Isabelle Maroger, Maurice Fallon, Milad Ramezani, Luca Marchionni, Olivier Stasse, Nicolas Mansard, and Bruno Watier. “ICP Localization and Walking Experiments on a TALOS Humanoid Robot”. In *2021 20th International Conference on Advanced Robotics (ICAR)*, December 2021, pp. 800-805.

CHAPTER 2

ICP Localization and Walking Experiments on a TALOS Humanoid Robot

In this chapter is presented the implementation of an ICP-based localization system using LiDAR data on the TALOS humanoid robot. This system was tested in a walking experiment were the robot had to reach defined position in our experimental room. This work has been published in [Lasg 21]. My contribution in this work is the integration of the LiDAR-based SLAM system developed at the Oxford Dynamic Robot Systems Group with a Visual-Inertial Odometry on the TALOS robot.

2.1 Introduction

In order to enable humanoid robots to autonomously perform useful behaviors in a semi structured environment, it is necessary to provide them with a way to autonomously localize themselves in a 3D environment. This is necessary to inspect specific important points in a factory, or to execute some manipulation tasks such as those performed during the DARPA Robotics Challenge (DRC). The difficulty is to maintain a sufficient precision for the targeted tasks while being in an environment whose parts are changing due to human activities. A similar problem occurs for autonomous cars navigating in a city. Indeed even if the architectural elements are consistent, the traffic is constantly perturbing the laser measurements. It is therefore important to have a robust and yet efficient way to localize the robot in a large 3D environment and to update the current status of the map.

In this chapter, the problem we are interested in is to assess the localization of the humanoid robot TALOS [Stas 17] such that it is able to perform a behavior such as manipulating an object or walking over debris. The behaviors can be adaptive through perception, control and planning, and they may have various requirements in terms of localization precision.

Finally, the capability to embed the localization software in the robot to make it autonomous is an important aspect to consider.

2.1.1 State of the art

The research community working on the Simultaneous Localization And Mapping (SLAM) problem is very active and consider many instances of the problem. Historically, for humanoid robots, it is based on computer vision and is targeting affordable perception sys-

tems and therefore uses monocular camera [Davi 07]. Since then, the DRC has popularized multiple sensors systems including stereo camera and LiDAR [Fall 15a] such as the MultiSense SLB product from Carnegie Robotics, that was recently incorporated in the Walkman humanoid robot from IIT [Tsag 17].

In [Horn 10], a laser-based localization was proposed for a NAO robot in a small multilevel indoor environment, later combined with a monocular camera in [Oß 12]. This localization was based on the Monte Carlo Localization technique presented in [Dell 99]. Nowadays the general framework of factor graph [Dell 17] allows to represent coherently the relationships between the robot state, its control vector and measurements of various sensors. Using this representation, it is possible to utilize a general non-linear solver to solve localization, calibration, state estimation or map building problems. However, sensors have different measurement frequencies and different memory complexities. For instance, in this setting the IMU provides 6 real accelerations and gyrometer measurements at 1kHz, while the LiDAR provides 65536 range measurements at only 10Hz. They need to be preprocessed in different ways before being incorporated in such representations.

For LiDAR-based measurements, Iterative Closest Point (ICP) and 3D points accumulation maps are efficient methods to propose such pre-processing [Pome 13]. Such methods have been recently refined and deployed in the field of legged robots and more precisely for quadruped robots [Nobi 17]. Indeed, it is now possible to find reliable industrial robots such as ANYmal from ANYbotics or Spot from Boston Dynamics. This chapter focuses on the practical implementation of such subsystem and its implementation on a research humanoid robot platform, here the TALOS robot from the company PAL-Robotics. Similar work has been realized using a Dense SLAM system [Tang 19], with a HRP-4 humanoid robot and an off-board computer equipped with a GPU. In this work we focus on the localization performance using an onboard computation.

In this work, we want to localize the TALOS humanoid robot in an indoor industrial-like environment. We use a modification of the initial TALOS humanoid robot head design including a flash LiDAR system.

2.1.2 Problem

For a given map m, the k-th LiDAR measurement $o_k \in \mathbb{R}^{64\times1024}$, and the Visual Inertial Odometry (VIO) pose estimation $p_k^{\mathscr{O}}$ in the fixed odometry frame $\mathscr{O}$, the goal of the localization system is to find the pose $p_k^{\mathscr{M}} \in SE(3)$ of the head in the map frame $\mathscr{M}$ such that $p_k^{\mathscr{M}} = ICP(m, o_k, p_k^{\mathscr{O}})$. The overall scheme is presented in Figure 2.1.

2.1.3 Contributions

The contributions of this chapter are the following:

- Successfully implement a LiDAR-based localization combined with a Visual-Inertial Odometry on a TALOS humanoid robot in order to accurately guide it. All computations are performed in real time on the robot's computer.

- Benchmark the localization system against a Motion Capture system.

2.2 Localizing in the map

Figure 2.1 shows the localization pipeline implemented on the robot that is presented in the following section.

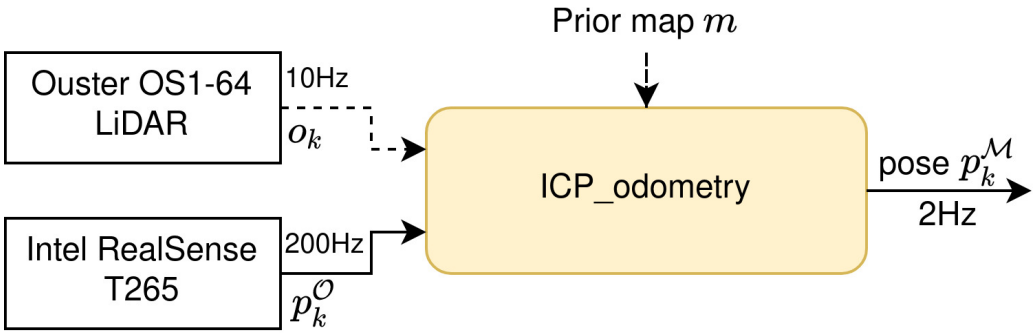


Figure 2.1: Block diagram of the localization system, dash-arrows show the flow of point clouds.

2.2.1 Iterative Closest Point

The ICP algorithm [Besl 92] is used to register two point clouds with overlapping parts. It has 4 main steps: pre-filtering, correspondence estimation, outlier filtering and lastly error-minimization. The main issue of this algorithm is the high influence of outliers that can result in convergence to a local minimum.

2.2.2 ICP-based localization

We use the localization system from [Rame 20] and [Nobi 17] over a known map.

The Autotuned-ICP (AICP) is an ICP registration method that adjusts the outlier filter of the ICP basis by computing an overlap parameter between the reading cloud and the reference cloud. It allows the registration of consecutive point cloud reads to a reference cloud, which in our case is a pre-built map.

This system uses the ICP implementation proposed by [Pome 13] and publicly available under the name *libpointmatcher*[1].

The system needs to be initialized. For this chapter, this pose is given by an approximation of the motion capture estimation. Further researches will be conducted on the estimation of this initial pose based on the map and LiDAR data.

At each subsequent steps the ICP refine the transformation between the measure o_k and the map m. As an estimation of the robot's pose $p_k^{\mathcal{O}}$ is given by the VIO, the measure o_k can be expressed in the $\mathcal{O}$ frame. The robot's pose in the map frame $\mathcal{M}$ is then obtained as $p_k^{\mathcal{M}} = c_k p_k^{\mathcal{O}}$, with c_k the alignment transformation between $o_k^{\mathcal{O}}$ and m computed by the ICP.

[1] `https://github.com/ethz-asl/libpointmatcher`

2.2.3 Visual-Inertial Odometry initialization

Kinematic-Inertial Odometry is effective for high-frequency estimation over a short time interval, whereas Visual-Inertial Odometry, through the use of lower frequencies exteroceptive sensors, remains reliable over a longer period. As both odometries are available on the robot and the experiments do not require a high-frequency state estimation, it was decided to use the Visual-Inertial Odometry instead of the Kinematic-Inertial Odometry, used in the initial system [Nobi 17, Rame 20].

The Visual-Inertial Odometry is given by an Intel RealSense T265 tracking camera with a frequency of 200 Hz. Figure 2.2 shows the comparison of the VIO to the Motion Capture estimation over 70 seconds, starting from the same initial point.

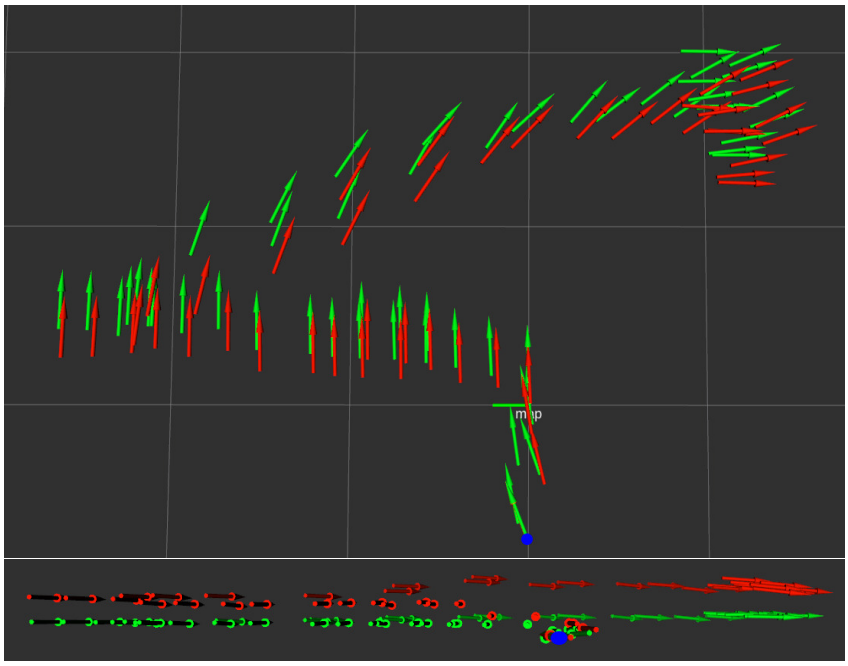


Figure 2.2: Downsampled visualization of the Visual-Inertial Odometry (red) and the Motion Capture estimation (green) over the two first targets. Top: view on the (x, y) plane, x upward. Bottom: view on the (z, y) plane, z upward.

2.2.4 Building the map

In this chapter, the indoor environment is known through a given map. In the following sections, different maps are discussed. Some of these maps were tested and compared in the experiment described in Section 2.4.

2.2.4.1 Architectural 3D model

Starting directly from an architectural 3D model can be a source of errors. Indeed, as shown in Figure 2.3, an architectural 3D model can contain occluded walls, e.g. on the

left of the figure, or small errors, e.g. the pillars in the middle. When comparing with the points accumulated using the ICP algorithm, mismatches appear with these features and may disrupt the convergence. As the walls in this kind of models are usually parallel in pairs, they create local minima.

A solution to that problem is to use a prior map that has been extracted from the 3D model, in order to remove the occlusions. It can be done manually or by mean of a simulation, provided that the sensor model is known. In this work, it was decided to remove the occluded parts manually to compare with the other proposed map introduced in the following section.

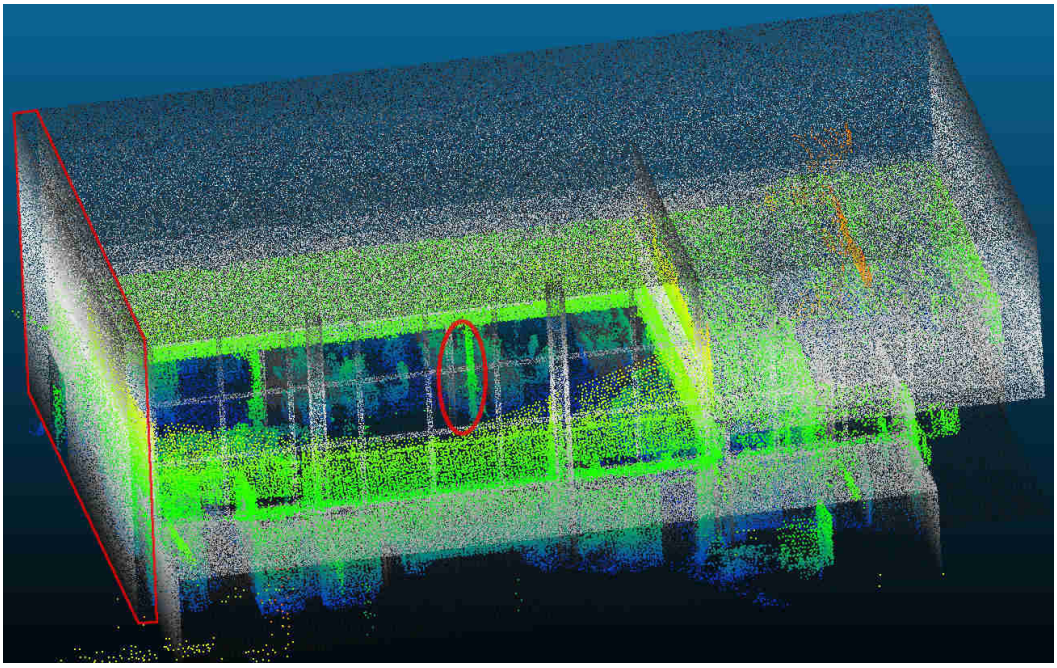


Figure 2.3: Comparison between a 3D model (gray scale) and an accumulation of LiDAR scans (color scale). Examples of occluded walls and errors are highlighted.

2.2.4.2 Using a scanning device

The map can be built prior to the robot localization using a scanning device. As no such specific device was available, it was decided to use the data returned by the LiDAR on the new head of TALOS. Figure 2.4 displays the map.

This kind of model can present some missing parts, e.g. in this experiment some parts of the floor and the ceiling are missing from the map, but it is easier to produce than an architectural map if an appropriate device is available.

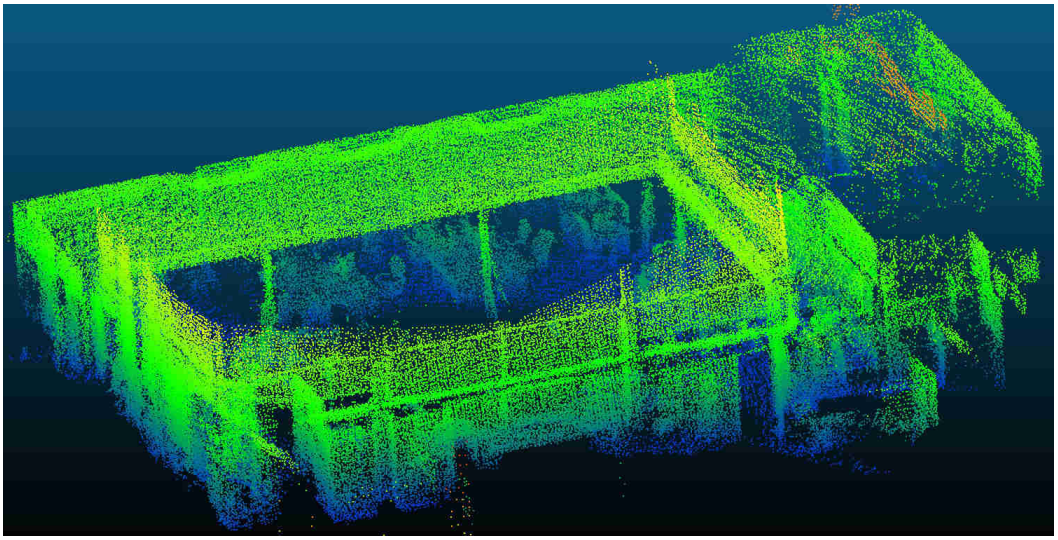


Figure 2.4: Map obtained by accumulating multiple LiDAR scans from multiple positions and orientations.

2.3 Walking

In order to assess the efficiency of the ICP-based localization system, experiments where the robot has to walk towards multiple given positions were performed. For these experiments, the walking algorithm used to generate the motion is the one provided by PAL-Robotics. It was used to send a velocity command to the robot on the topic `/walking_controller/cmd_vel` at a 2 Hz frequency. The velocity command was merely computed as the distances along the x and y axis in the coordinate frame linked to the robot and angle between the robot and its goal orientation around the z axis. Thresholds of $0.1\,\mathrm{m/s}$ for the linear velocities and $0.12\,\mathrm{rad/s}$ for the angular velocity were applied on the command in order to not reach the robot limits. Moreover, the command was designed so that the robot stops walking when it is at small enough distances and angle from its target, respectively $0.08\,\mathrm{m}$ and $0.07\,\mathrm{rad}$. This stopping distance was implemented to prevent the robot to trample on its target instead of stopping on it, this allows swift stop once on spot. Once the robot stops on a target, the robot is programmed to stay still during $8\,\mathrm{s}$ before moving on to its next target.

2.4 Experiments on the robot

2.4.1 Experimental setup

The experimental room is equipped with a Motion Capture system (MoCap) including 20 infrared Qualisys Miqus M3 cameras sampling at up to 650 Hz with a $3 \times 10^{-4}\,\mathrm{m}$ accuracy on the viewed area. This motion capture system was used to record the positions and orientations of the robot's head. We placed 5 passive markers on its torso and 1 on the top of its head. In this study, the MoCap measurements are hypothesized to be the ground truth data. Those measurements were compared to the ones made with the ICP-based

localization system in order to assess its accuracy and its performances.

A new head was designed for the TALOS robot and is shown on Figure 2.5. From top to bottom, it is equipped with an Ouster OS1-64 LiDAR, an Intel RealSense T265 and an Intel RealSense D435i. The latter is not used in the experiment.

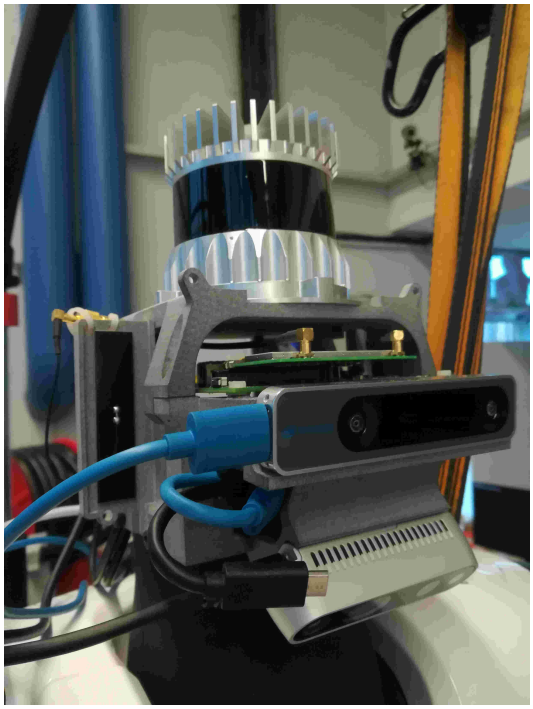


Figure 2.5: TALOS equipped with an Ouster OS1-64 LiDAR, an Intel RealSense T265 tracking camera and an Intel RealSense D435i RGB-D camera

The LiDAR has a range between 0.8 m to 120 m, an horizontal field-of-view of 360° and a vertical field-of-view of 33.2° using its 64 laser beams. For the experiment, the LiDAR was set to take 1024 horizontal samples per scan and to return point clouds at 10 Hz.

The Intel RealSense T265 is a tracking camera embedding two Fish-Eye cameras, an IMU and a Vision Processing Unit (VPU). This camera runs a V-SLAM algorithm on the VPU with an output frequency of 200 Hz.

2.4.2 Experimental protocol

The performed experiment consists of a series of targets the robot has to successively reach. Thus, six goal positions were defined in the experimental room. Those positions are represented on Figure 2.6. They have been chosen in such a way that the robot ranges the whole MoCap viewed area and faces diverse orientation changes. The position of the robot at the beginning of the experiment is not defined in advance, it could be anywhere in the viewed area. Note that, if the robot starts from the origin of the MoCap reference frame, it will have to travel around 10.6 m to perform the whole experiment.

During the experiments, a dataset was recorded on a *rosbag*, a recording of the mes-

sages exchanged through the middleware *ROS*, including the measurements made by the MoCap, the velocity orders sent to the robot, the LiDAR data, the tracking camera images and estimations and the ICP-based localization results.

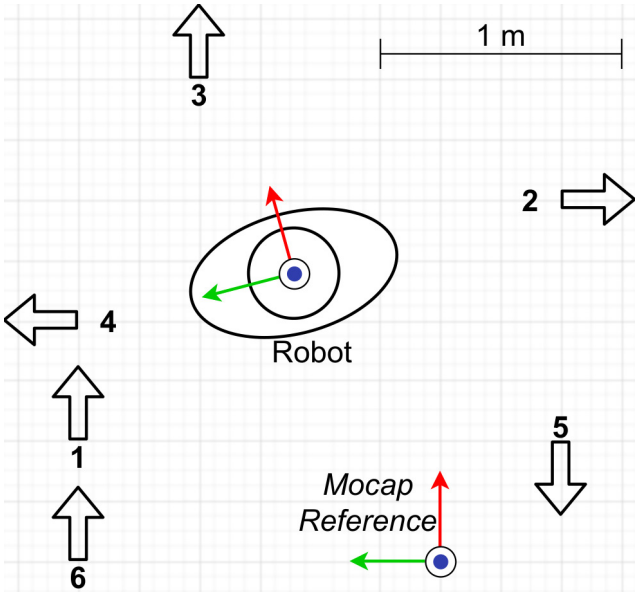


Figure 2.6: Goal positions in the experimental room

This experiment was successfully performed 3 times with a TALOS robot. Due to technical problems and hardware issues on the robot legs and ankles, further experiments resulted in failures and the fall of the robot, preventing the authors to gather more data. Among those successfully performed experiments, the first one was performed with the map built from the architectural 3D model (Section 2.2.4.1) whereas the second and the third ones were performed using the map built by using a scanning device (Section 2.2.4.2).

A screenshot from the video of the first experiment is shown in Figure 2.7. A video explaining the experimental protocol and showing the second experiment is available at `https://youtu.be/0t1bBjDTqMA`.

2.5 Discussion

2.5.1 Comparison between the motion capture system and the ICP-based localization system

During the experiments on the robot, two data sets were collected simultaneously: one with the positions and orientations of the robot head recorded by the MoCap and another recorded by the ICP localization system. As both localization systems do not have the same sampling frequencies, the raw datasets cannot be compared directly. Indeed, the MoCap dataset includes around 40 times the amount of data of the ICP one as Figure 2.8 shows. This is why both data sets were interpolated in order to have the same length.

First of all, for the three datasets, a delay between the ICP system and the MoCap can be

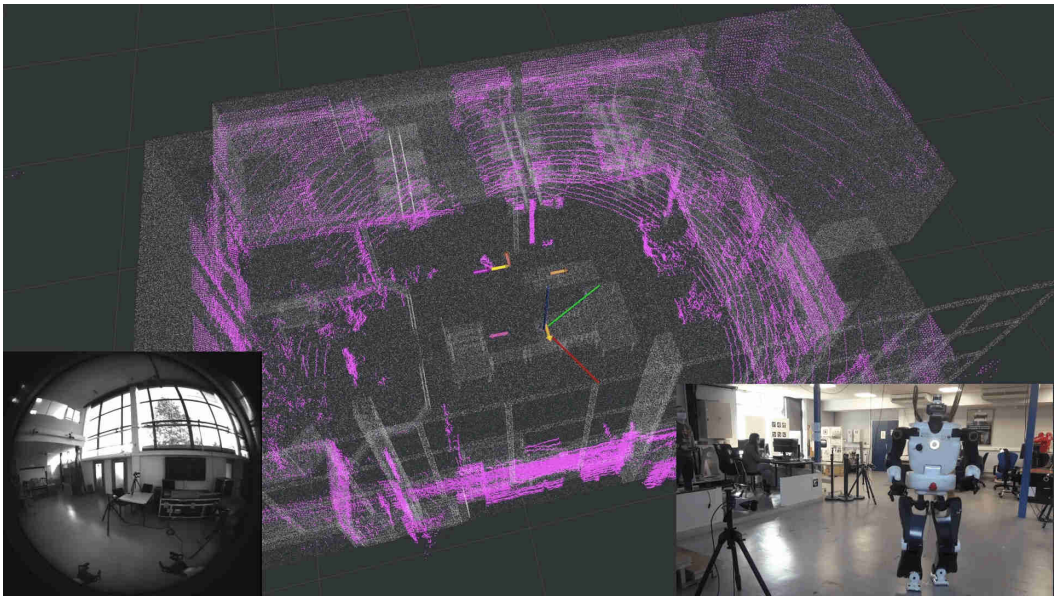


Figure 2.7: Frame extracted from the video of the first experiment. The video shows the ICP data (gray: prior map, pink: read point cloud aligned, axis: estimated pose) and the targeted poses (arrows) in the middle, the image from a sensor of the Intel RealSense T265 on the left and the video of the experimental room and the robot on the right.

Figure 2.8: Visualization of the recorded data on Rviz (in *red* the MoCap set and in *green* the ICP set)

measured shifting the two datasets to minimize the difference between the (x, y, z) positions and the θ orientation measured by the two localization systems. This delay scores between 0.51 s and 0.62 s depending on the experiment and is mainly due to the computation time of the ICP. In the following, this delay has been removed from the ICP set.

	On x (mm)	On y (mm)	On z (mm)	Around z (rad)
Exp. 1	18.7 ± 17.2	24.9 ± 19.3	9.2 ± 7.2	0.025 ± 0.019
Exp. 2	14.8 ± 17.3	19.9 ± 17.8	5.4 ± 4.4	0.023 ± 0.027
Exp. 3	13.8 ± 15.9	18.9 ± 15.7	5.6 ± 4.8	0.024 ± 0.027

Table 2.1: Computed errors between the MoCap and the ICP data sets for all the experiments

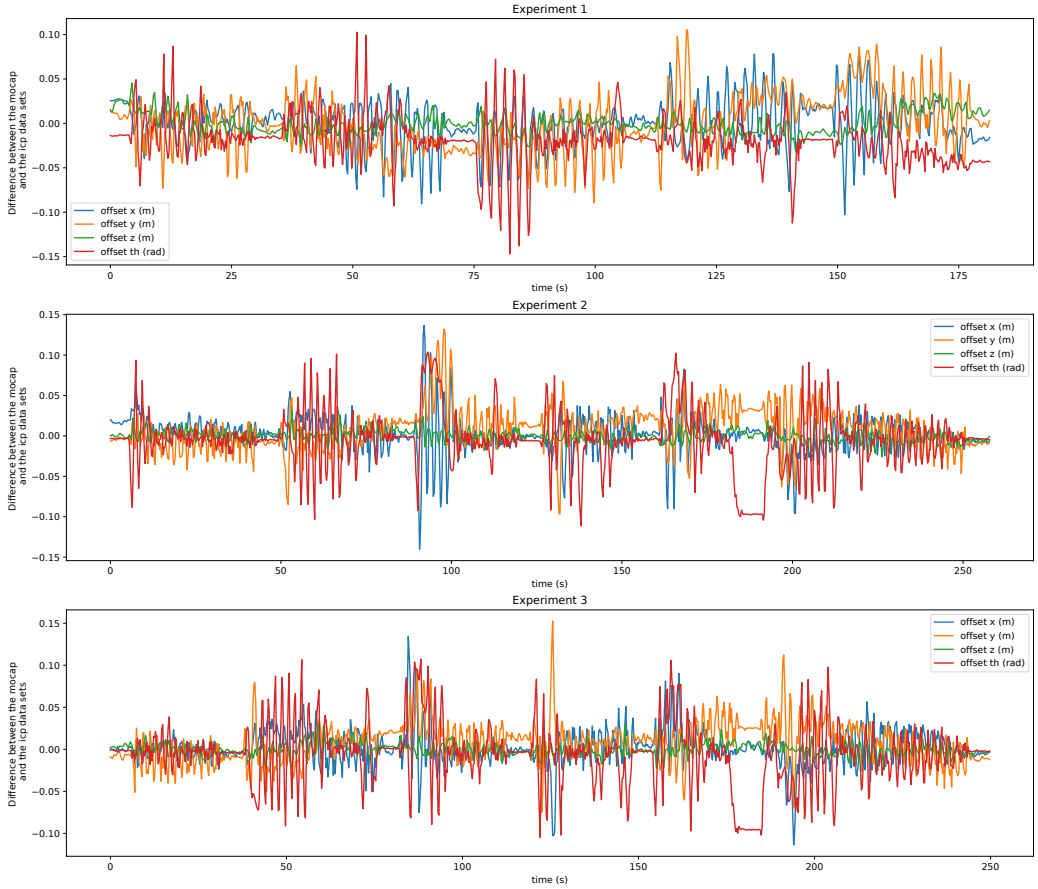


Figure 2.9: Errors between the 2 datasets during the 3 experiments (x in *blue*, y in *orange*, z in *green* and θ in *red*)

Then, the differences between the two datasets on the position and on the orientation can be computed. These differences will now be referred as offsets. The evolution of these offsets over the whole experiments is shown in Figure 2.9. These offsets are of two kinds

- Structural offsets due to a difference between the coordinate frames of both localization systems. We make the assumption that these offsets are constant as it seems to be the case in Figure 2.9 and can easily be computed as the mean of the differences between the datasets over the experiment duration. Thus, these offsets score 0.19 m, 0.058 m, 0.097 m respectively along the x, y and z axis and 1.12×10^{-4} rad around the z axis for the first experiment. These offsets are lower for the second and the third experiment (respectively 0.045 m, 0.021 m, 0.058 m and 0.048 rad) because the coordinate systems were reset after the first experiment. In the following, these structural offsets are considered equal to zero as they have been removed to the ICP dataset as the MoCap was taken as the ground truth. The results after removing these offsets are represented in Figure 2.10.

- Errors due to real differences of results between the two localization systems. They can be computed as the mean of the differences between the two datasets after removing the structural offset. These errors are presented in Table 2.1.

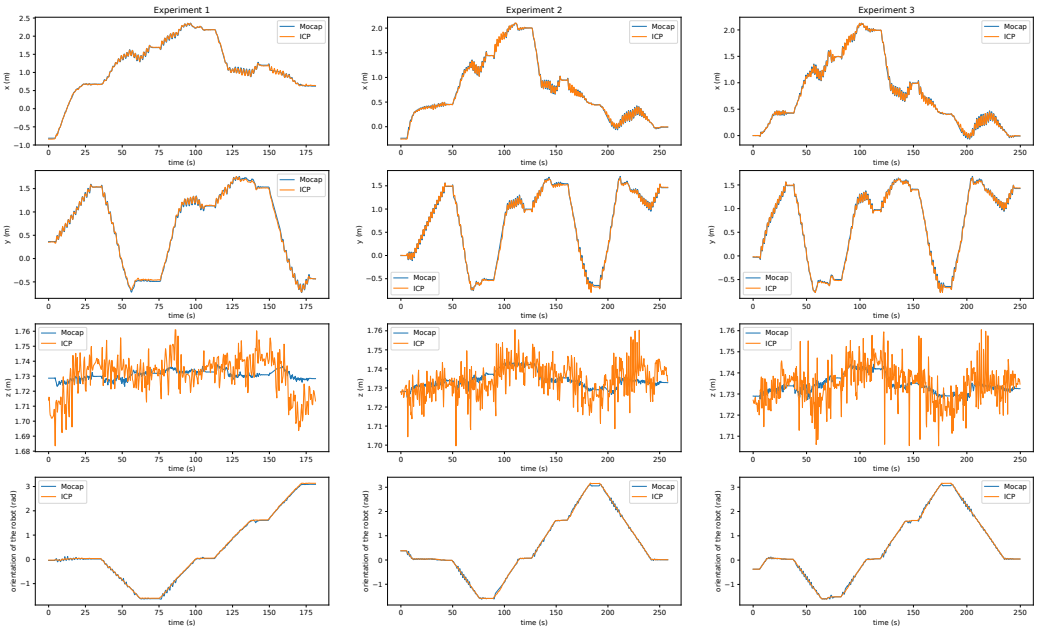


Figure 2.10: Position (x on the first row, y on the second row, z on the third row) and orientation (θ on the last row) of the robot measured by the MoCap (in *blue*) and by the ICP localization system with the removed delays and structural offsets (in *orange*) during the 3 experiments

During the 3 experiments, the robot started from 3 different positions with diverse orientations. As the results shown in Table 2.1 are similar regardless of the experiment, we can conclude that the ICP-based localization system does not depend on the starting pose. Moreover, the robot traveled different distances according to the experiments, respectively 8.93 m, 11.05 m and 10.69 m. As Figure 2.10 shows, the error does not seem to increase with the traveled distance, at least for distances around 10 m.

2.5.2 Accuracy on the goal positions

The goal positions can easily be identified on the datasets as they match with the points with zero velocity. To assess the accuracy of the ICP-based localization system on achieving the goal positions, the difference between the measured position and the desired goal position can be computed. The absolute average error on the goal positions ($\pm$ standard deviation) is $0.025 \pm 0.016\,\text{m}$ on the x axis, $0.039 \pm 0.042\,\text{m}$ on the y axis and $0.028 \pm 0.012\,\text{rad}$ around the z axis. All these errors are lower than the tolerance admitted on stops stated in Section 2.3 which means that the accuracy of the ICP localization system is satisfactory for the targeted application. Moreover, with such accuracy, more complex localization tasks could be considered. In future work, the ICP-based localization system will be used to place the robot in front of an object to perform a task requiring accuracy such as drilling or crossing debris.

2.5.3 Effect of the map on the ICP-based localization system

As Table 2.1 shows, there is no significant difference (less than the standard deviations) in the errors between the first experiment and the second and the third experiments whereas the ICP-based localization system was performed with different maps. This means that, even if the map built from the architectural 3D model is a priori less accurate than the map built using a scanning device, the ICP localization system has an accurate behavior in both cases.

2.6 Conclusion

We presented a pipeline to localize the TALOS robot in an indoor environment based on LiDAR and Visual-Inertial Odometry. The VIO is used to initialize the ICP steps registering read point clouds to a map point cloud. The system was successfully deployed on the robot and tested in a walking situation. The results were compared to a MoCap system considered here as the ground truth.

The ICP-based localization system shows an average error of $0.016\,\text{m}$, $0.021\,\text{m}$ and $0.0067\,\text{m}$ along the x, y and z axis and of $0.024\,\text{rad}$ around the z axis with respect to the motion capture system taken as the ground truth. Those results show that the ICP is accurate enough to target, in future work, more challenging localization tasks such as precisely placing the robot in front of an object to interact with it.

The evaluation of the use of a state-of-the-art descriptor to localize known objects in LiDAR measurements is discussed in the next chapter.

CHAPTER 3

FPFH based object recognition and localization

3.1 Introduction

This chapter deals with the problem of object recognition and localization in a near or distant environment using geometric information. Furthermore, since the objective is to plan a movement around or using the object of interest, it is reasonable to assume that in an industrial environment the model of the object is known, e.g. in the form of a CAD model.

Localizing an object in a robotic application resides in expressing its model pose ${}^{R}p_{obj}$ in the robot space R. This pose is an $SE(3)$ object, either an homogeneous matrix or a position and a quaternion, representing the position and the orientation of the object in the environment of the robot. This problematic is represented in Figure 3.1. To do so, it is usual to first express the object pose in the sensor's frame $\mathscr{S}$, written as ${}^{\mathscr{S}}p_{obj}$. As the link between the robot space R and the sensor's frame S is kinematic, the final pose of the object in the robot space ${}^{R}p_{obj}$ can be estimated.

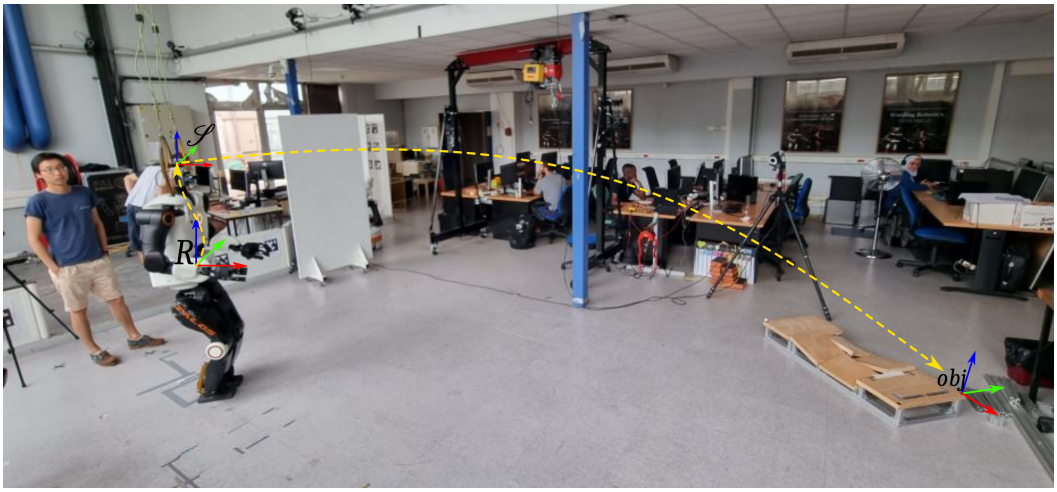


Figure 3.1: Problem of estimating the pose of an object to be used by a robot, here the platforms and Talos.

This work focuses then on dense perception using sensors such as LiDARs or RGB-D cameras. From the many descriptors that exists, the use of the Fast Point Feature Histogram presented in [Rusu 08a] is studied. FPFH descriptors are presented as fast and discriminant descriptors and are used in multiple other works [Zhou 16, Li 21, Yang 20]. This descriptor

is used to determine the pose of the object. First it is used to determine which part of the scene is to be analysed, then to estimate an initial guess for a simple yet locally efficient Iterative Closest Point algorithm. In order to perform these needs, we need to know how to define the correspondence between descriptors thanks to a distance and how to interpret this distance.

This chapter first present related works on 3D descriptors in Section 3.2. Section 3.3 summarises the definition of the Fast Point Feature Histogram [Rusu 08a]. Section 3.4 presents previous work on the object localization done during my master internship. Section 3.5 describes the occlusion problem induced by cameras and LiDAR. Section 3.6 presents our design of a multi-view and multi-level model to localize objects in LiDAR data. Section 3.7 presents the results of the experiment conduced with our model and the improvement proposed. Lastly, Section 3.8 concludes this chapter and discuss its results.

3.2 Related work

A common method to analyse large amount of data generated by depth sensors is to use descriptors. Descriptors are a way to reduce and normalize the information presented by a set. In depth analysis, descriptors will be used to compare two data sets with possibly different level of details or resolution, using a common and normalized representation. This section discuss descriptors used to analyse and compare point clouds extracted from depth sensors. The presentation will go through unlearned descriptors and methods using neural networks.

3.2.1 3D descriptors

In order to define correspondences between point clouds, we looked on the available state-of-the-art descriptors. The idea behind these descriptors is to describe a 3D region, represented as a set of points, as a unique term, such as an histogram. Thus, the objective is to be able to evaluate the similarity between two regions by comparing the resulting descriptors. This feature is interesting when doing point cloud matching and registration because it helps determining correspondences between the clouds, reducing the computation cost by avoiding the need of a point-to-point comparison.

Nowadays, reviews of the existing descriptors can be found like [Spez 20], evaluating some handcrafted and learnt descriptors, or [Stan 21], concentrated on the descriptors and detectors available in the *Point Cloud Library*. The descriptors can be firstly separated between *global* or *local* descriptors. *Global* descriptors like [Osad 02, Aldo 12] intend to describe a complete 3D model and is usually used with a segmented scene to avoid the influence of occlusions. On the other hand, *local* descriptors, on which this work concentrate, describe a small part of the surface, small compared to the whole scene of model being used. These 3D descriptors can further be classified as *spatial distribution histograms* and the *geometric characteristic histograms*.

In the first category lies descriptors like the *Spin Image* [E.Jo 99], the *3D Shape Context (3DSC)* [From 04], an extension of the *2D Shape context* [Belo 02], or the *Unique Shape*

Context (USC) [Tomb 10a]. These descriptors represent region by histograms generated by the spatial coordinates, segmenting the region with respect to a Local Reference Frame or Axis (LRF/LRA). For example, *3D Shape Context (3DSC)* divide the space around the query point by defining a sphere centered on the point with its' direct reference being the surface normal at the point. Then this sphere is divided regularly along the azimuth and elevation and logarithmically along the radial dimension. Each of these division correspond to a bin in the descriptor, representing the spatial distribution of points in this division.

The second category concentrate descriptors like the *Point Feature Histogram (PFH)* [Rusu 08a, Rusu 08b] and its variants like the *Fast Point Feature Histogram (FPFH)* [Rusu 09], or the *Signature of Histograms of OrienTation (SHOT)* [Tomb 10b, Tomb 11]. This kind of descriptor focus on geometric attributes such as normals or curvatures. In the case of the *Signature of Histograms of OrienTation (SHOT)* descriptor, the neighborhood of a query point is divided by a spherical grid as for the *3DSC*. Then for each space subdivision, a local histogram is made of the points classified by the angle between their normals and the normal of the query point. These local histogram are then put together to create the final signature.

3.2.2 Learned descriptors

The success of neural networks based machine learning techniques has motivated numerous works on point cloud processing. [Spez 20] classified these methods in 4 groups: *View-based*, *Voxel-based* and *Point-based* methods and *Geometric Deep Learning* approaches.

The *View-based* approaches use 2D views of an object and 2D Convolutional Neural Networks to define features. For instance, [Su 15] uses rendered views of an object provided to 2D CNNs. The learnt features are then merged by a view pooling method and are sent through a last CNN. This work was shown to be efficient to classify shapes.

Point-based methods works directly with point clouds. PointNet [Qi 17a] was a pioneer in this way. Point clouds are directly fed to a neural network, and then a symmetric function is applied to ensure invariance to permutation. The network gives a classification score for each learned classes. This work has been shown useful for data classification but also for partial segmentation of objects or scene segmentation. However, PointNet is not able to capture local structure. The authors then introduced a multi-scale architecture, PointNet++[Qi 17b], where PointNet is applied to the nested partitions of the point cloud.

Voxel-based techniques are based on 3D voxels, grid of occupancy or density of points. For example, VoxelNet [Matu 15] takes a point cloud, builds an occupancy grid which is then fed to successive CNN and result in a classification of the input data.

The limitations of these learning based methods is in the training process. They need to have a large amount of data to cover a wide spectrum and obtain satisfactory results. Whereas these methods learn descriptors "from scratch", it is possible to learn define features, like those presented previously, in a self-supervised manner to enhance the predictability of the training result.

In the state-of-the-art, the main interest on point cloud processing lies in data classification and segmentation. These methods may be used in the object localisation problem to build an attention-based system. The classification networks can provide an estimate of

where to look at the object.

3.3 Fast Point Feature Histograms

3.3.1 Point Feature Histogram

Point Feature Histograms, PFH are features that describe the neighborhood points distribution of a query point p. It is computed with a fixed radius r which define the sphere around p in which each point is considered as part of the k-neighborhood.

Figure 3.2 shows the neighborhood of a query point and the interaction considered for the feature computation. Each pair taken into consideration for the feature computation is shown by a line with the query point p_q.

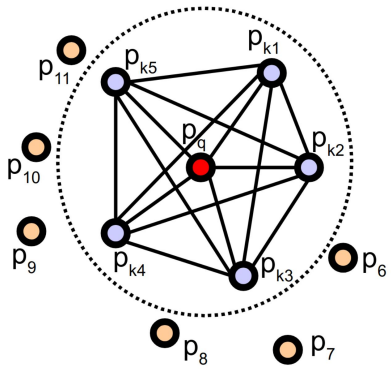


Figure 3.2: Neighborhood interactions considered in PFH definitions, from [Rusu 09].

Taking each pair (p_i, p_j) in this k-neighborhood (p_i being the point in the pair with the shortest angle between its normal and the vector linking the points), a Darboux $u \times v \times n$ frame is defined as:

$$u = n_i, v = (p_j - p_i) \times u, w = u \times v \tag{3.1}$$

From the points, their normals and this Darboux frame, 4 features are computed:

$$\begin{aligned} f_1 &= v \cdot n_j \\ f_2 &= ||p_j - p_i|| \\ f_3 &= (u \cdot (p_j - p_i))/f_2 \\ f_4 &= \arctan(w \cdot n_j, u \cdot n_j) \end{aligned} \tag{3.2}$$

With these 4 features, the evaluated points and their local sphere are classified in an histogram. The bin index is first computed as follow [Rusu 08a]:

$$idx = \sum_{i=1}^{4} step(s_i, f_i) \cdot 2^{i-1} \tag{3.3}$$

with $step(s,f)$ defined as 0 if $f < s$, 1 otherwise. Then, s_i can be defined to be the center of the definition interval of f_i. The points are then classified by two categories for each features and the percentage of pairs which have the same category for all four features is saved.

In [Rusu 08b] this definition is replaced by:

$$idx = \sum_{i=1}^{4} \left[\frac{f_i \cdot d}{f_{i_{max}} - f_{i_{min}}} \right] \cdot d^{i-1} \tag{3.4}$$

with []-operator being the integer part operation, d the number of subdivision of the feature's maximum theoretical value range $(f_{i_{max}} - f_{i_{min}})$. After being increased by 1, each bin's value is normalised with the total number of point pairs $(k \cdot (k+1)/2)$.

In the first way of computing idx, the pairs are classified in $2^4 = 16$ bins. The second expression permits to increase the number of bins by defining d (e.g. with $d = 3$, the pairs are classified in $d^4 = 81$ bins). Figure 3.3 shows the Point Feature Histogram obtained for different geometric shapes and how they compare to each other with an euclidean distance. These Point Feature Histograms are computed with a total of 27 bins, 9 bins per feature, without using the euclidean feature of the pair f_2.

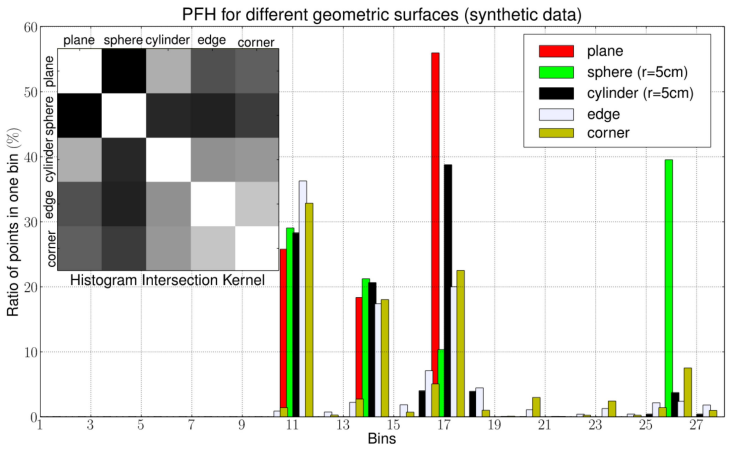


Figure 3.3: Example of Point Feature Histogram for points on basic 3D geometric shapes, from [Rusu 09].

3.3.2 Fast Point Feature Histogram

Fast Point Feature Histograms (FPFH) [Rusu 09] are features proposed as an improvement of the Point Feature Histogram (PFH) [Rusu 08a, Rusu 08b] which reduces the computational complexity.

PFH has a theoretical computational complexity for a given point cloud with n points of $O(n \cdot k^2)$ [Rusu 09], with k the number of neighbors for each point p in the point cloud. FPFH reduces this computational complexity to $O(n \cdot k)$ by reducing the pairs computed. For that, two simplifications are applied.

First, only three features are computed from (3.2):

$$\alpha = f_1,\ \phi = f_3,\ \theta = f_4 \tag{3.5}$$

Then, these features are computed only on pairs composed of the seek point p and its neighbors. Figure 3.4 shows the new handling of the neighborhood of the query point for FPFH. Each pair taken into consideration for the feature computation is shown by a line with the query point p_q. On the figure, direct interactions of the point p_q are in red and thick lines are interactions considered twice.

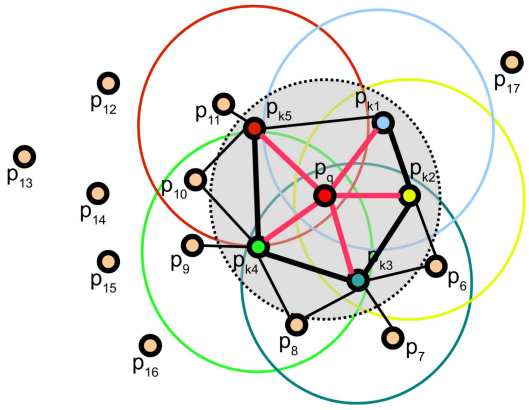


Figure 3.4: Neighborhood interactions considered in FPFH definitions, from [Rusu 09].

These features are classified in bins as for Point Feature Histogram. This histogram is called Simplified Point Feature Histogram, SPFH. To compute the FPFH, the SPFH of a seek point is weighted by its neighboring SPFH values:

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_i}\cdot SPFH(p_i) \tag{3.6}$$

where ω_i represents the distance between p and p_i in a given metric space.

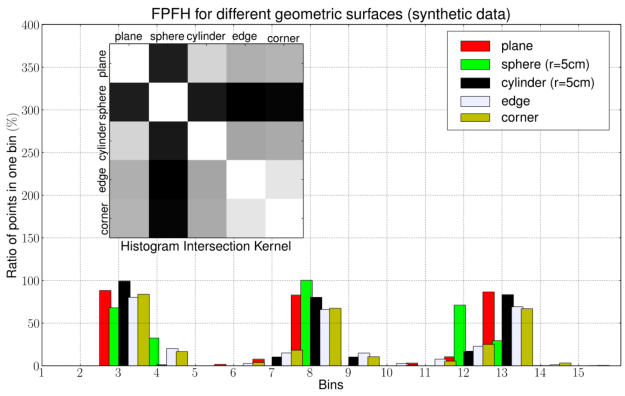


Figure 3.5: Example of Fast Point Feature Histogram for points on basic 3D geometric shapes, from [Rusu 09].

Then the final histogram is normalized for each features for comparison purposes. Figure 3.5 shows the Fast Point Feature Histogram obtained for different geometric shapes and how they compare to each other with an euclidean distance. The FPFH is computed with 33 bins, 11 per feature.

3.3.3 Strengths and weaknesses

FPFH features has a low theoretical computational complexity compared to the previous PFH formulation. Because it does not fully interconnect all neighbors of p, it does not model precisely the surface around p, but, because it values are weighted with neighboring SPFH values, it includes information outside the r-radius sphere.

For our work, FPFH has a major limitation which is that it can not be computed directly on the model if it has parts thinner than 2 times the neighboring radius r. Indeed, if it has thin parts, features on a surface can be noisy because of the computation over the opposite surface. This limitation is related to the occlusion problem presented in Section 3.5.

3.3.4 Feature space definition

The definition of the features as a normalized histogram that represents the distribution of the considered pair of points on certain criteria allows defining the feature space. It is denoted as $\mathscr{F}_k^n$ with n the dimension of the space and k a constant parameter constraining the 1-norm of the feature and defined in the implementation of the descriptor.

3.3.4.1 Space definition

This space is a subspace of $\mathbb{R}_+^n$ defined by:

$$\mathscr{F}_k^n = \left\{ f \in \mathbb{R}_+^n, \|f\|_1 = k \right\} \tag{3.7}$$

As this space is a subspace of $\mathbb{R}_+^n$, it can be defined that $|f_i| = f_i$ in our space, and so:

$$\|f\|_1 = \sum_{i=0}^{n-1} |f_i| = \sum_{i=0}^{n-1} f_i \tag{3.8}$$

This space is by essence a bounded space as the 1-norm of an element is constant and it implies that for an element $x \in \mathscr{F}_k^n$, $x_i \leq k$.

3.3.4.2 Membership of $k * \{e_i\}$

Let $\{e_i\}$ be the canonical basis of $\mathbb{R}_+^n$ and considering $k\{e_i\}$, then:

$$\|ke_i\|_1 = k \|e_i\|_1 = k, \forall i \tag{3.9}$$

Thus $k\{e_i\}$ is part of $\mathscr{F}_k^n$.

3.3.4.3 Convexity

Let x and y be two elements of $\mathscr{F}_k^n$, a scalar $t \in [0,1]$ and $p = tx + (1-t)y$.

It is known that $\forall i,\ (t, x_i, y_i) \geq 0$, thus $\forall i, p_i \geq 0$.

$$\begin{aligned} \|p\|_1 &= \sum_{i=0}^{n-1} p_i = \sum_{i=0}^{n-1} (tx_i + (1-t)y_i) \\ &= t\sum_{i=0}^{n-1} x_i + \sum_{i=0}^{n-1} y_i - t\sum_{i=0}^{n-1} y_i \\ &= tk + k - tk \\ \|p\|_1 &= \sum_{i=0}^{n-1} p_i = k \end{aligned} \tag{3.10}$$

Thus $p = tx + (1-t)y$ is part of $\mathscr{F}_k^n$ and $\mathscr{F}_k^n$ is convex.

3.3.5 Distance analysis

As defined previously, the FPFH descriptor is composed of 3 feature's histograms that are elements of $\mathscr{F}_k^n$. Let's denote the descriptor as $d = [{}^d f_1\ {}^d f_2\ {}^d f_3]$ with $f_1, f_2, f_3 \in \mathscr{F}_k^n$, this vector is of size $m = 3n$.

3.3.5.1 Distance definition

By definition, the euclidean distance between two descriptors v and w is:

$$\delta_{v,w} = \|v - w\|_2 = \sqrt{\sum_{i=0}^{m-1} (v_i - w_i)^2} \tag{3.11}$$

This can be reduced with Eq. 3.12 as stated in Eq. 3.13.

$$\delta_{{}^v f, {}^w f} = \|{}^v f - {}^w f\|_2 = \sqrt{\sum_{i=0}^{n-1} ({}^v f_i - {}^w f_i)^2} \tag{3.12}$$

$$\delta_{v,w} = \sqrt{\sum_{i=1}^{3} \delta^2_{{}^v f_i, {}^w f_i}} \tag{3.13}$$

Thus the descriptor distance's range $[min_{\delta_{fpfh}}(=0), max_{\delta_{fpfh}}]$ can be evaluated thanks to the feature's one $[min_{\delta_{f_i}}(=0), max_{\delta_f}]$ as:

$$max_{\delta_{fpfh}} = \sqrt{\sum_{i=1}^{3} max^2_{\delta_{f_i}}} = max_{\delta_f}\sqrt{3} \tag{3.14}$$

3.3.5.2 max_{δ_f} determination

As said previously, the features can be seen as elements of the $\mathscr{F}_k$. Let be x, y two elements of $\mathscr{F}_k^n$ and from the definition of the set in 3.3.4.1, there exists two elements $\lambda, \mu \in \mathbb{R}_+^n$ with $\lambda_i, \mu_i \in [0;1]$ such as $x = k\lambda$ and $y = k\mu$.

As $\|x\|_1 = \|k\lambda\|_1 = k\|\lambda\|_1 = k$, then it can be written that $\|\lambda\|_1 = 1$ and the same applies to y and μ.

$$\begin{aligned}
\|x-y\|_2 &= \sqrt{\sum_{i=0}^{m}(x_i-y_i)^2} \\
&= \sqrt{\sum_{i=0}^{m}(k\lambda_i-k\mu_i)^2} \\
&= k\sqrt{\sum_{i=0}^{m}(\lambda_i-\mu_i)^2} \\
&= k\sqrt{\sum_{i=0}^{m}\lambda_i^2+\sum_{i=0}^{m}\mu_i^2-2\sum_{i=0}^{m}\lambda_i\mu_i}
\end{aligned} \tag{3.15}$$

As it is known that $\lambda_i \leq 1$, $\mu_i \leq 1$, it can be considered that $\lambda_i^2 \leq \lambda_i$, $\mu_i^2 \leq \mu_i$ and $\|x-y\|_2$ can be majored by:

$$\begin{aligned}
\|x-y\|_2 &= k\sqrt{\sum_{i=0}^{m}\lambda_i^2+\sum_{i=0}^{m}\mu_i^2-2\sum_{i=0}^{m}\lambda_i\mu_i} \\
&\leq k\sqrt{\sum_{i=0}^{m}\lambda_i+\sum_{i=0}^{m}\mu_i-2\sum_{i=0}^{m}\lambda_i\mu_i} \\
&\leq k\sqrt{1+1-2\sum_{i=0}^{m}\lambda_i\mu_i} \\
&\leq k\sqrt{2}
\end{aligned} \tag{3.16}$$

On another side, the distance between 2 different elements of $k\{e_i\} \in \mathscr{F}_k^n$, with $i, j < n$ and $i \neq j$, is:

$$\left\|ke_i - ke_j\right\|_2 = k\left\|e_i - e_j\right\|_2 = k\sqrt{2} \tag{3.17}$$

As shown, $k\sqrt{2}$ is an upper-bound of δ_f and is the distance between two elements of $k\{e_i\}$ so the maximum distance can be defined as $max_{\delta_f} = k\sqrt{2}$.

3.3.5.3 Conclusions on $max_{\delta_{fpfh}}$

Thanks to the previous analysis, $max_{\delta_{fpfh}}$ can be determined as:

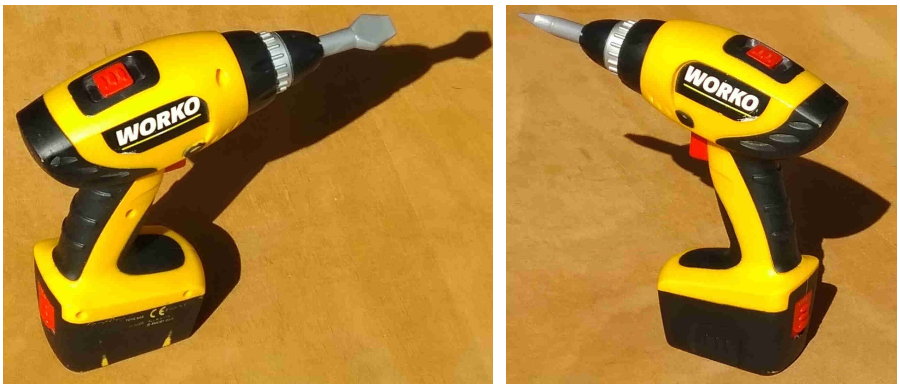
$$max_{\delta_{fpfh}} = max_{\delta_f}\sqrt{3} = k\sqrt{6} \tag{3.18}$$

With k defined by the implementation of the descriptors, e.g. $k = 100$ in the Point Cloud Library and $k = 1$ in Open3D.

This knowledge is helpful when trying to estimate correspondences between two sets of descriptors as it is now known the expected order of distances between two features. The closer the distance is to 0, the more identical the descriptors are and the more similar the regions represented can be considered. On the opposite, the closer the distance is to $max_{\delta_{fpfh}}$, the less similar the regions represented can be considered.

3.4 Previous work on object localization realised at LAAS

During a 6-months internship preceding this work, research were done on a model-based localization system for objects in order to perform manipulation tasks. Two objects were tested: a children's toy that looks like a screwdriver and a drill. Figure 3.6 shows the toy used and can be compared to the model that is shown later. The project was using RGB-D cameras such as an Orbbec Astra Pro, that was equipping the humanoid robot Talos, and a Microsoft Kinect v2. These two cameras are active RBG-D cameras, i.e. equipped with IR projectors and cameras.



(a) (b)

Figure 3.6: Visualisation of the children's toy used in the internship.

3.4.1 Building a model

This study had a first part with the hypothesis that the object model is not known. Moreover, because of the inaccuracies of the RGB-D cameras and the auto-occlusions of the object, it is not possible to see the whole object with a single shot. Thus, how to build the model from previously acquired measures was studied.

The principle of the model generation is then to take multiple shots of the object, allowing to cover the majority of the object, and to realign these views in order to acquire a complete model. Figure 3.7 shows several views of an object. This figure illustrates the need to take several views to know the whole object.

In order to ease the construction of the model and the realignment of the different views and to avoid misalignment when basing the realignment on the object only, the views' information were enhanced using a table on which the object is placed. Considering that the object will stay at a fixed pose during the measurement for the model, tags were put on the table in order to estimate the transformation between two views. When this estimation is done, the model will then be realigned using these estimations as initial guess for the ICP method.

Figure 3.7: Several shots acquired with the Microsoft Kinect v2 of the child toy.

As the object is fixed during a complete measure, some parts of the object may be missing in the measured model. Thus, the same process was repeated with different poses of the object on the table. The multiple partial models created are then realigned to create a complete representation, as a point cloud, of the object. For example, Figure 3.8 displays a model of the toy presented previously. This model was built with 3 poses of the object, i.e. lying on each side and straight, for a total of 112 frames.

(a) Side view (b) Back view (c) Front view

Figure 3.8: Model generated for the children's toy (112 frames taken).

3.4.2 localizing the object

Once the model built, it can be used to localize the object in a scene. For this work, I have taken over the work of a previous intern. The architecture is presented in Figure 3.9 with its data pre-processing steps in red and the localization steps in green.

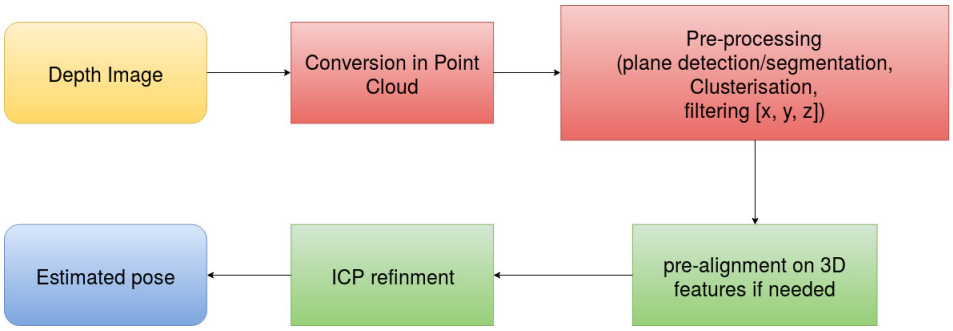


Figure 3.9: localization algorithm used.

The pre-processing steps consists of converting the acquired data, that are an RGB image and a Depth image, in a point cloud. Then, this point cloud is filtered to extract a pre-defined looking zone, supposed to be a square zone representing the work space of the robot. After this filtering, a plane detection is applied to define the main planar structure, e.g. a table or the floor. The objects present "on" this plane are extracted as clusters and are used to perform the localization.

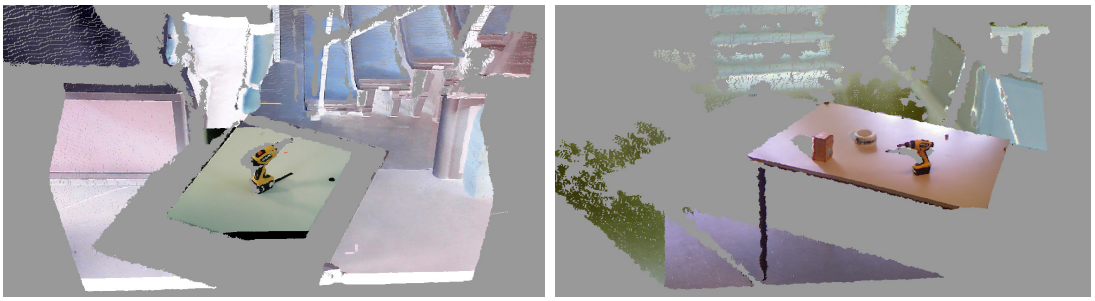
The localization step takes all the clusters determined by the pre-processing and try to align to it a given model, e.g. the children's toy built before. For each of these objects, a pre-alignment is performed using FPFH descriptors and a RANSAC algorithm. The result is used as an initial guess for a plane-to-plane ICP algorithm, i.e. an ICP method trying to align using also normals to the points. The alignment are qualified in term of *fitness* following Eq. 3.19, with a_i a point in the measured object and b_i it's closest point in the aligned model.

$$\Delta = \frac{1}{N}\sum_{i=0}^{N-1}\left(\sqrt{(a_{i,x}-b_{i,x})^2+(a_{i,y}-b_{i,y})^2+(a_{i,z}-b_{i,z})^2}\right) \tag{3.19}$$

3.4.3 Localization results

The localization was evaluated visually, i.e. the results was printed and the localization was defined as a success if the model was visually matching with the view. This evaluation was performed on two example cases. An "easy" case, where the object is alone on a table and the exploration limits are fixed to the table, and a "hard" case where the exploration limits are extended, the table is further away from the camera and there are 2 more objects on the table: a tea box and a roll of tape. Figure 3.10 shows these two cases, the exploration zone is represented by the points with colors that is not inverted.

In term of computation time, the "Easy" case was solved in 3 iterations on average, with a mean time of $5s$. The pre-processing took $100ms$ per iteration, with 90% for the



(a) “Easy” case (b) “Hard” case

Figure 3.10: Visualisation of the evaluation cases.

clustering step. On the other hand, the “Hard” case was solved in a mean time of 10s, with on average 10 iterations. In this case, the pre-processing took 225ms per iteration, with half of the time on the clustering part.

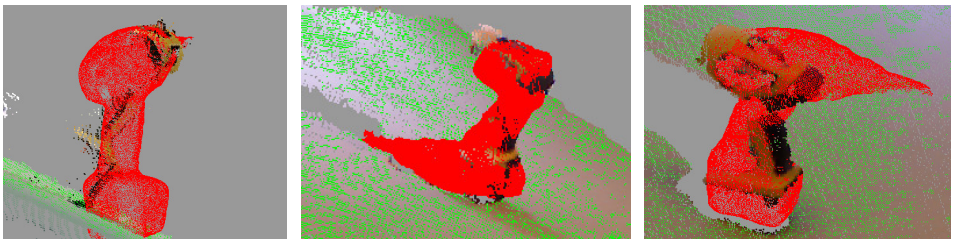
During this evaluation, four main errors were observed, Figure 3.11 shows three of them:

Skewed alignment Because of the similarity between the two main faces of the object, when the handler part is partially occluded, both the head and the foot of the object can be aligned on different faces of the model.

Head-to-tail Because of the similarity between the back of the head and the foot of the object, in case of a strong occlusion of the sides of it, the object can be vertically inverted.

Wrong direction As for the “Skewed alignment”, the similarity of the two faces can produce an alignment error in the wrong direction, inverting the aligned faces.

Wrong object In the case with multiple objects present, the model can be aligned to another one.



(a) Skewed alignment (b) Head To Tail (c) Wrong direction

Figure 3.11: Main alignment errors on the object. The alignment error on the wrong object is not represented here.

The evaluation campaign is resumed in Table 3.1. It consists of 50 measures of the “Easy” case and 40 of the “Hard” case, with different poses of the object in each measure.

This first campaign shows that the results are promising but there is still improvements possible regarding the robustness to the error.

As a matter of fact, because in this localization system, only the geometric information provided by the depth measurement is used. The errors may be avoided with an analysis of the color information that is not studied here.

Case	"Easy"	"Hard"
Number of tests	50	40
Distance object-camera	70cm	120cm
Number of objects on the plane	1	3
satisfactory alignment	37 (74%)	21 (52.5%)
Skewed alignment errors	2 (4%)	9 (22.5%)
Head To Tail errors	6 (12%)	5 (12.5%)
Wrong direction errors	5 (10%)	4 (10%)
Wrong object errors	0	1 (2.5%)

Table 3.1: Rough evaluation of the localization robustness in the two cases.

3.5 A sensor's physics story: the occlusion problem

When doing model-based object localization, a problem that occurs is due to the occlusions. An occlusion appear when part of an object is hidden by another object or is hidden behind itself. This work tries to address the last problem, called auto-occlusions.

3.5.1 The occlusion problem

This problem appears because every part of the object cannot be seen in the same time. As represented in the figure 3.12 in the case of a LiDAR, the beam is reflected by the first part of the object encountered.

The occlusions bring different types of error.

The first part of this occlusion problem is on the neighborhood definition of the tools used as the descriptors. As shown by the figure 3.13, some occluded parts represented on the model can disrupt the descriptor and thus make the estimation of correspondences difficult. In the figure, two views of a same object can be seen, respectively the representation of a measure from a LiDAR and the complete model. The neighborhood of the same point is represented by the circles. As shown on the representation of the model, parts that are completely absent from the measure are in this neighborhood and are considered for the descriptor computation.

In order to overcome this occlusion problem, two possibilities can be imagined. Firstly, a viewpoint parameter could be added in the descriptor computation; e.g., using the complete model of the object, state-of-the-art methods could be used to remove any occluded

part in the neighborhood of a query point in order to define a descriptor that only uses potentially visible points. Secondly, a simulation tool could be used to generate a dataset of realistic views of the object thanks to a simulation of the sensor used, the descriptors are then computed on each view and the neighborhood of a query point is only composed of visible points.

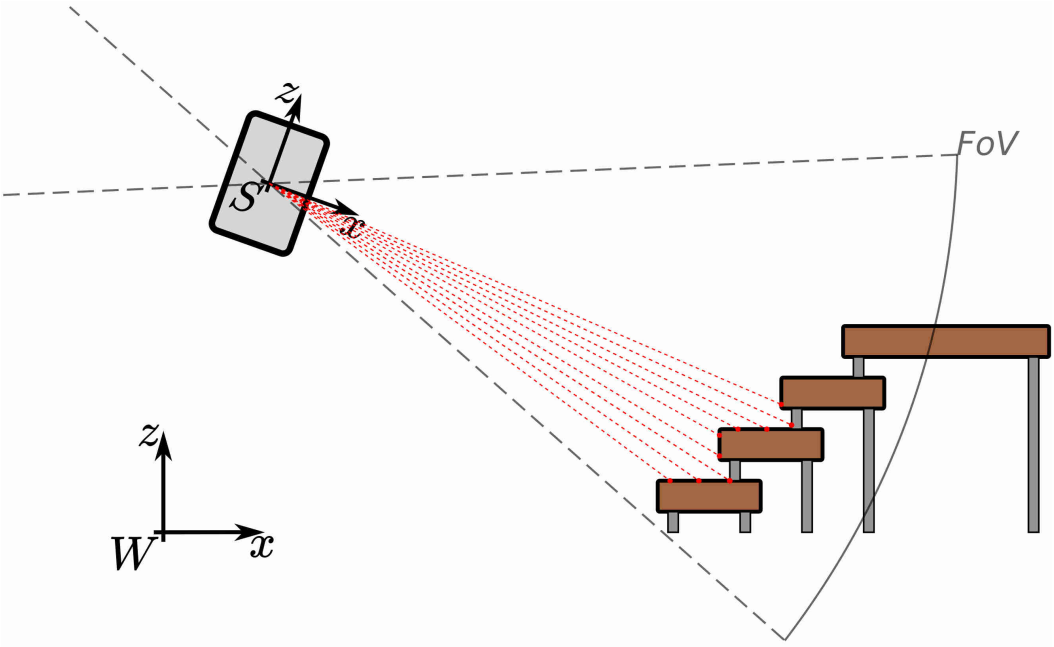
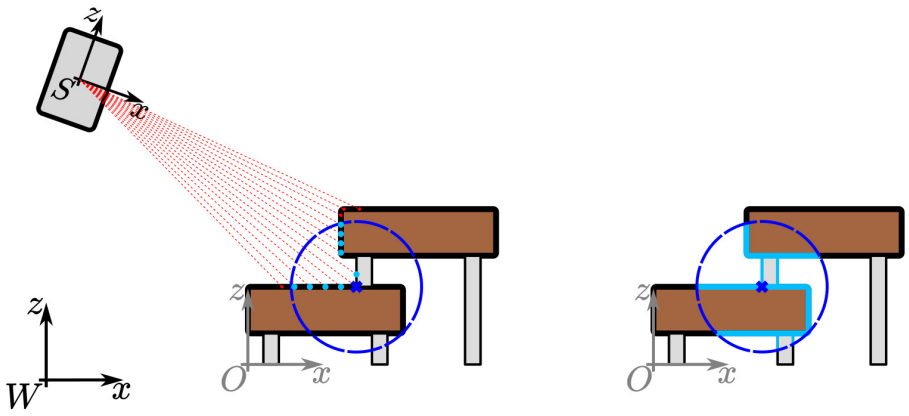


Figure 3.12: Representation of the occlusion system, example of a LiDAR.



(a) LiDAR measurement (b) Complete model

Figure 3.13: Representation of the influence of the occlusions on the neighborhood. Blue: query point and neighborhood borders, Turquoise: considered neighborhood.

In this work, it was decided to use the second solution, using a software like Gazebo to simulate views of our object from a LiDAR. Indeed, this solution makes it easier to take

the sensor model into account.

The second part of the occlusion problem is that a measure represents only a part of the object while the model is based on a complete representation of the object. Thus it is not necessary to compare the measure with all the model data for the registration phase. To lower the complexity, it is desirable to compare the measure with only the appropriate part of the object model. When using a dataset of realistic views of the object, this problem can be reduced by finding and using the closest view to the measure in the dataset.

3.5.1.1 Resolution of the sensor

The resolution of our sensor is defined by its number of vertical and horizontal samples and its FOV. With the characteristic listed on Table 3.2.

Description	Value
Vertical FOV	$FOV_V = 33.2°$
Horizontal FOV	$FOV_H = 360°$
Vertical samples	$n_V = 64$
Horizontal samples	$n_H = 512/1024/2048$

Table 3.2: Table of the Ouster OS1-64 LiDAR characteristics

By approximating the angles between two consecutive beams either vertically or horizontally, θ_V and θ_H, the vertical and horizontal distances, respectively noted Δ_V and Δ_H in the measure between two points of a surface can be approximated by:

$$\begin{aligned} \theta_V &= \frac{FOV_V}{n_V} \\ \theta_H &= \frac{FOV_H}{n_H} \\ \Delta_V &= r_{sensor} * tan(\theta_V) \\ \Delta_H &= r_{sensor} * tan(\theta_H) \end{aligned} \tag{3.20}$$

With r_{sensor} the distance of the points to the sensor given thanks to ${}^F p_{sensor}$ and ${}^F p_{object}$, expressed in the same frame F, the respective positions of the sensor and the object, by:

$$r_{sensor} = \left\| {}^F p_{object} - {}^F p_{sensor} \right\| \tag{3.21}$$

The definitions of Δ_V and Δ_H shows us, as can be imagined by the physics of the sensor, that the further away the sensor is from an object or the smaller the object is, the fewer points corresponding to the object are measured. This limit is in addition to the one represented by the occlusions.

This resolution bound the distance at which a multi-beam sensor can "see" an object, i.e. have enough points on the object to be able to recognize it efficiently. The same goes

for the simulation, the distance between the sensor and the object needs to be tuned in order to be realistic with the use cases and to admit enough points to represent the object.

3.5.1.2 Approximation of the measure's resolution

Given the previous information and the diameter d_{MBS} of the minimal bounding sphere of the object of interest, the number of points that can be expected in the measurements for the object can be roughly approximated. Before going into it, it is needed to note that this approximation take a lot of hypothesis thus it is not expected to be precise but it is expected to bound the value. These hypothesis are:

- the world is noiseless, this can be true in simulation but will never be in the real world,
- the object is a single disc of diameter d_{MBS},
- the sensor's beams are all in normals direction to the surface.

Considering $\underset{\sim}{\theta}$ as the worst beam FOV, i.e. the greater value between θ_V and θ_H for a given configuration, ρ_{lin}, the theoretical linear resolution of acquired points at a distance r_{sensor}, can be estimated as:

$$\rho_{lin}\left(r_{sensor}\right) = \frac{d_{MBS}}{r_{sensor} * tan\left(\underset{\sim}{\theta}\right)} = \frac{d_{MBS}}{\underset{\sim}{\Delta}} \tag{3.22}$$

For example, in the case of our LiDAR, the different θ, dependent of the horizontal setting, can be computed as below:

$$\begin{aligned} \theta_V &= 33.2/64 = 0.51875^\circ \\ \theta_{H|512} &= 360/512 = 0,703125^\circ \\ \theta_{H|1024} &= 360/1024 = 0,3515625^\circ \\ \theta_{H|2048} &= 360/2048 = 0,17578125^\circ \end{aligned} \tag{3.23}$$

So in the case of our LiDAR, $\underset{\sim}{\theta}$ is equal to θ_V unless it is decided to setup the sensor at 512 horizontal samples in which case $\theta_H > \theta_V$.

3.6 Multi-view and multi-level description

This section describes the first implementation of the two main solutions proposed: a multi-view model to resolve the problem of occlusions and a multi-level model to resolve the problem of resolutions and distances.

3.6.1 Multi-views generation

In a first time, a “perfect” sensor was used to simulate a number of coherent views of the object of study. Thus, the use of the sensor physics avoids the noise induced by the perception of occluded points.

3.6.1.1 Implementation

Gazebo is used to simulate how the sensor see the objects. To avoid any influence from external objects, a simple world is used. This world is created without any floor, external light and gravity.

To define the view points, the observable space of the object is firstly defined. This observable space is defined as the surface of a sphere with the object in its center. To complete the definition of the observable space, some constraints are fixed as follows:

- r is fixed to be equal to the diameter of the minimal bounding sphere of the object, thus the view points are on a spherical zone.
- The observable space is divided uniformly on a (r, θ, φ) space, r being fixed and the θ and φ subdivision being experimentally determined by the user.

To have coherent viewpoints, their positions are defined by sampling the observable space defined. Moreover, contextual constraints are added to filter out the surface of the observable space. For example, if the object should not be upside-down in the environment, the bottom views are not needed. The radius of the sphere, is defined in accordance with the needed point density for the descriptor and the possible resolution of our sensor.

3.6.1.2 Example of Memmo’s platforms

Figure 3.14 shows the initial position of the world with the platforms and the LiDAR.

In the case of these platforms, the situation where the object is "upside-down" is considered to never occur. Thus the observable space can be defined as the z-positive space in the case of a frame defined with the (x, y) plane on the bottom part of the object and the z axis upward. From this space, an half-sphere is defined and segmented over latitude and longitude parameters to define the poses of the simulated sensors. However, going with this definition over express a part of the space compared to the other parts, here the top of the half-sphere compared to the bottom. Figure 3.15 shows the distribution of the sensors for the views generated.

Figure 3.16 shows some examples of the views of the object obtained with different views. As in Figure 3.15, the axis on an angle of the object is the origin of our model, the second axis is the pose of the sensor for each view. It shows that each part of the object is not seen in each views.

3.6.2 Multi-view and multi-level descriptors generation

Each view is then used to generate a set of FPFH for each point using a set of *scales*. The scales are defined as in Section 3.6.2.2, with $S_M = 1.34$ (the diameter of the minimal

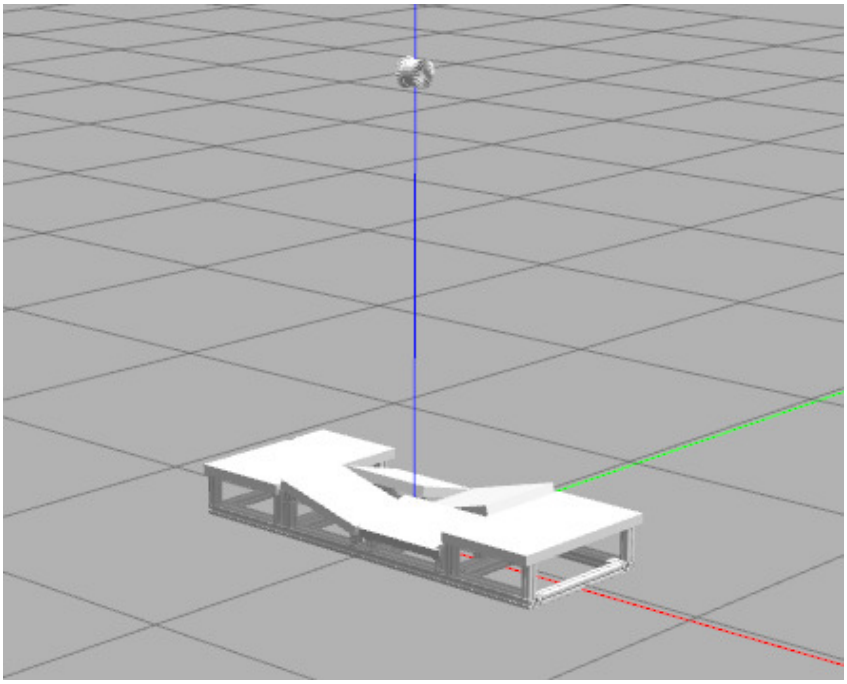


Figure 3.14: Gazebo world with the object and the LiDAR. Axis: x blue, y green, z red.

bounding sphere) and $s_m = 0.01$.

3.6.2.1 Multi-Scale Spherical Neighborhoods

The Multi-scale Spherical Neighborhoods defined in [Thom 18], inspired by the KNN space division [Hack 16], is a way to divide the space to create space features in a multi-scale way, keeping them undistorted and with a sufficient density. Taking a point cloud C, it defines the neighborhoods of a point p_0 in C with a r-radius as:

$$\mathfrak{S}_r(p_0, C) = \{p \in C \mid \|p - p_0\| \leq r\} \tag{3.24}$$

The multi-scale architecture is then defined using 3 parameters: φ the ratio between the radius of consecutive scales, S the number of scales considered, r_0 the radius of the smallest neighborhood. Then the neighborhood of a point p at a s-scale is:

$$N_s(p_0) = \mathfrak{S}_{r_s}((p_0), C) \tag{3.25}$$

with $r_s = r_0 * \varphi^s$ and $s \in \{0, .., S-1\}$.

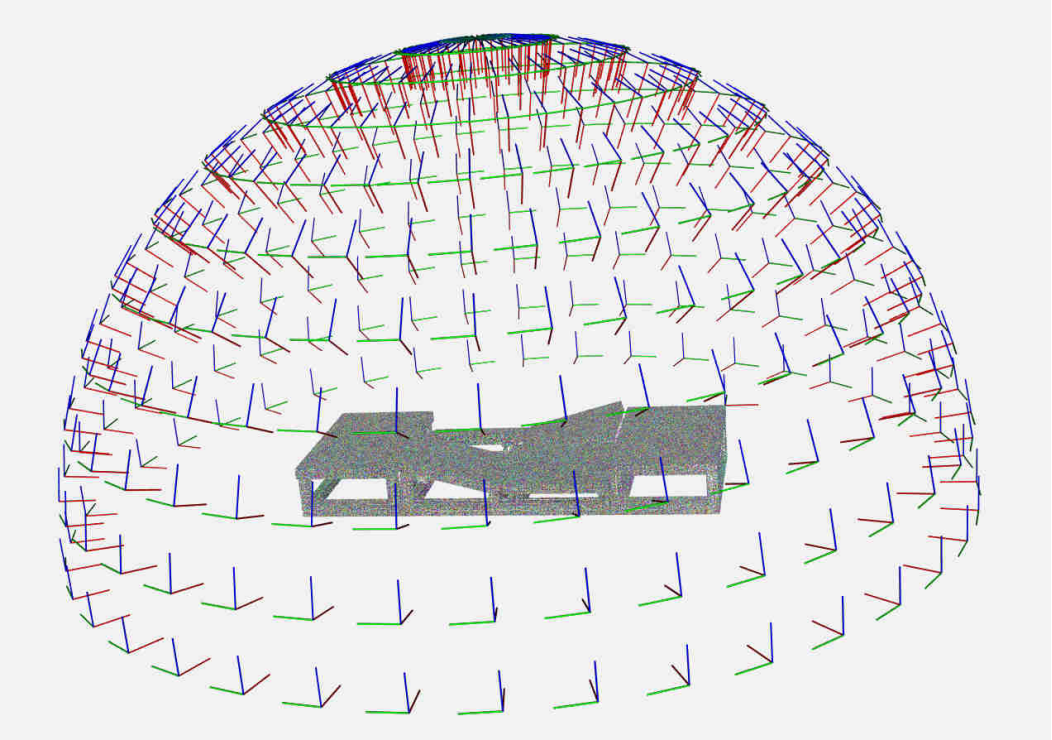


Figure 3.15: Visualization of the sensors poses for each view generated. The 3-axis on an angle of the object is the origin used to express the model, each other axis is the pose of a sensor for a generated view.

3.6.2.2 Definition of S_r

For our application, the scales $S_r = \{s_0, .., s_M\}$ need to be defined. In the original work [Thom 18], this set is established as:

$$S_r = \left\{ s_i | s_i = s_0 \varphi^i \right\} \tag{3.26}$$

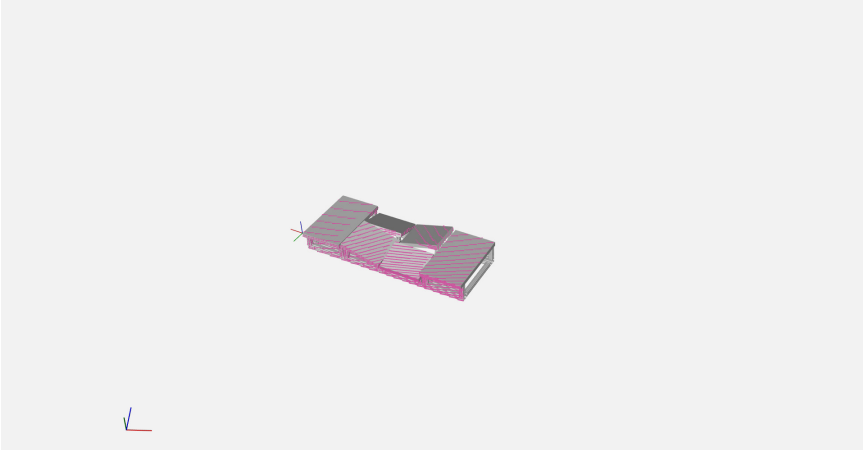
As the objective is to represent an entire object, the biggest descriptors' scale may be influenced by the point cloud C and should admits all the visible points of our object. Given this definition, the biggest scale s_M of the set S_r is defined as follow, with $M = |S_r|$:

$$s_M = \max \left\| p_i - p_j \right\|, (p_i, p_j) \in C \tag{3.27}$$

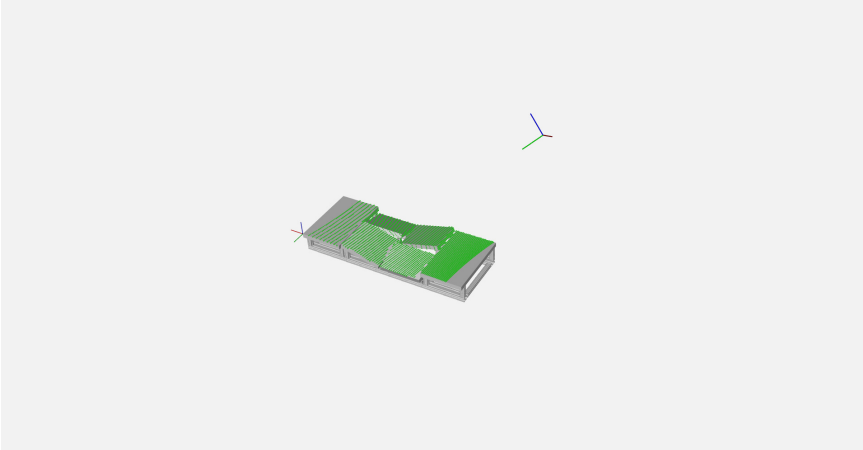
Given this constraint, Eq. 3.26 can be modified according to the fact that the flow goes from the maximal scale value and not the minimal one. Thus the radius definition become:

$$S_r = \left\{ s_i \mid s_{i+1} = \frac{s_i}{2}, s_0 = s_M \right\} \tag{3.28}$$

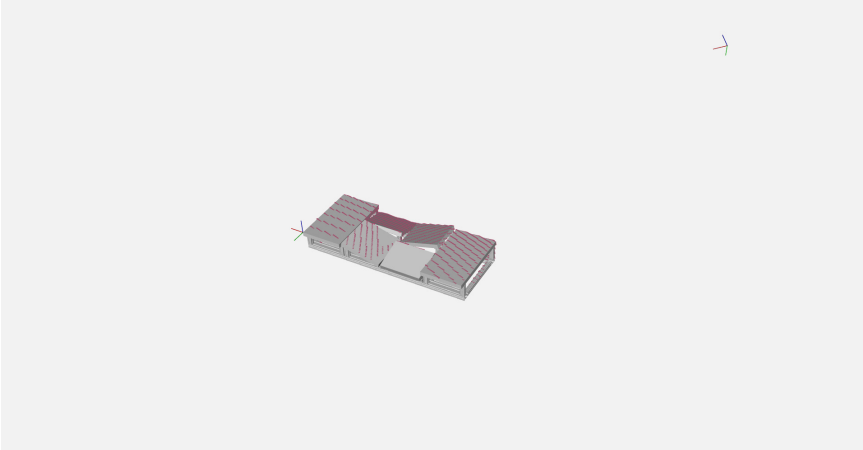
A minimal radius s_m can be defined to determine the number of radii in accordance and not as a parameter of the descriptor. This s_m can be defined in order to have a theoretical number of points which is considered as a minimal need for a descriptor. As the objective



(a)



(b)



(c)

Figure 3.16: Example of views of the object given by Gazebo.

is to represent a surface and as, in dense perception, a surface needs at least 3 points to be determined, it was decided to fix the minimal number of points as 3.

Thus the definition of s_m could be dependent of the mean distance between two points δ as:

$$s_m = 2 * \delta \tag{3.29}$$

This definition allows to consider that at a radius s_m, there should be 3 points in the neighborhood. But it is not always true as the approximation of the image resolution with dense sensors is dependent to the distance between the sensor and the object, the pose of the object and the object itself, as presented in Section 3.5.1.2. In the experiment presented later, s_m has been fixed empirically.

3.6.2.3 Number of scales

Going with this logic of having a starting point s_M and a stop criteria $s_i > s_m$, and with using a by-two division at each scale, The number of scales in the model is determined with:

$$n_S = \|S_r\| = \left\lfloor \log_2 \left(\frac{s_M}{s_m} \right) \right\rfloor + 1 \tag{3.30}$$

3.6.2.4 Generation of a model

Using this definition, a model generation was defined as shown in Algorithm 1.

Algorithm 1: Generation of a model

Input: Set of views of the object as point clouds V, Set of scales S, number of clusters to use k

Output: Multi-view and multi-level model

```
foreach s_i in S do
    foreach view V_j in V do
        D_{V_j,s_i} = FPFH(V_j, s_i)
        K_{V_j,s_i} = clusterize(D_{V_j,s_i}, k)
        save(V_j, s_i, D_{V_j,s_i}, K_{V_j,s_i})
```

Where *FPFH(P,s)* is the method to compute the FPFH descriptors of each points of P with a s-radius neighborhood and *clusterize(D,k)* is the clustering reducing a set of points or descriptors D into a smaller set of size k, using for example the K-means algorithm.

3.6.3 Object detection via scale drop

A new model has been presented, the way to use it efficiently is now studied. As the objective is to localize a given object, the multi-scale implementation can so be used to firstly reduce the part of the world in which the object should be searched.

This means that, given a point cloud P acquired thanks to the sensor and representing the scene observed, the objective is to extract a subpart in which the object should be.

Giving that our model is composed of n_{views} point clouds and n_{scales} set of features for each point cloud with a feature for each point at each scale, the number of points in the point cloud i is noted $n_{p|i} = \|\mathfrak{C}_i\|$. The number of features for each complete scale m can be estimated as $n_{FPFH|m} = \sum_{i=0}^{n_{views}} n_{p|i}$.

Algorithm 2: Scale drop object detection

Input: Point cloud of the scene $\mathscr{C}$, object model (views, set of scales S and descriptors $\mathscr{F}$) and a selection function *bests*
Output: Reduced scene $\mathscr{C}_r$

$\mathscr{P} = \mathscr{C}$; // point cloud to be processed
/* Split the point cloud in candidate zones and get their center points */
$\mathscr{K}$ = subsample($\mathscr{P}$, s)
$i_S = 0$
$s = S[i_S]$
while *stop()* $\neq$ *True* **do**
  $kern_s$ = get_clusters($\mathscr{F}_s$) ; // Get the clusters of all the views
  for $p \in \mathscr{K}$ **do**
    $\mathscr{N}_p$ = extract_neighborhood($\mathscr{P}$, p, s)
    F_p = get_scaled_fpfh($\mathscr{N}_p$, p, s)
    $cand$ = get_closest_core($kern_s$, F_p)
    $d[p]$ = distance($cand$, F_p)
  sort(d)
  $corr$ = bests(d) ; // get the set of p selected for this scale as "inliers"
  /* Continue to the next scale if the criterion are not validated. */
  $\mathscr{K}$, $\mathscr{P}$ = new_search_zone($\mathscr{C}$, s, $S[i_S + 1]$, $corr$)
  i_S ++
  $s = S[i_S]$
$\mathscr{C}_r = \mathscr{P}$; // Give the reduced scene as the last selected zone

The pipeline is for a given scale is presented in Algorithm 2. Some steps of this algorithm permits to tune it.

subsample() is a step where the point cloud is divided in smaller parts represented by their middle points, defined as the cores of the candidate zones. In this step, a voxelization or a random sampling can be implemented for example.

stop() is the stopping criterion. For example, it is based on the scale's value or the size of the reduced zone.

extract_neighborhood() objective is to apply a neighborhood extraction such as a spherical extraction, thus defining subparts of the space around each point given by the previous

subsampling. This neighborhood extraction is per essence applied to the initial measured scene.

get_scaled_fpfh() generate a descriptor using the method presented in Section 3.3.2.

get_closest_core() extract from the set $kern_s$ the descriptor closest to the studied one F_p. The *distance()* used is presented in Section 3.3.5.

ge_clusters() is one of the major tuning points as it's objective is to reduce the computation time while keeping an sufficient efficiency. E.g. a k-means algorithm can be used to clusterize the $n_{FPFH|r}$ descriptors to a set of k nucleus. It is sacrificing estimation accuracy in favour of a reduction in computational complexity.

bests() is a second major tuning point. It defines the way how two FPFHs are compared, and how a good match is defined. The FPFH distance analysis in sec. 3.3.5 can be used to select the correspondences up to a threshold or rank the correspondences over their distances and select the bests ones. With the threshold, good correspondences may be expected. But if the threshold is too high, few or no correspondences can be found. While on the other side, the ranking evaluation may give enough points to continue further but with less correspondences to the object.

new_search_zone() allows to define how the search zone is reduced while processing each consecutive scales. For example, in an experiment that is presented later, the new search zone is defined as presented in Algorithm 3. The presented solution has a particularity: as the new zones' cores are selected as being part of the selected candidates and the new zones as the neighborhoods of the cores in the initial scene, some previously ignored point may be back in the search zone. These behaviour is wanted to be able to get back filtered parts of the object. As the definition of the candidate zones provides pseudo-randoms candidates without taking into account the consistency of the data. Thus, objects may be separated in multiple candidates zones. Small parts of the objects that are filtered because they are of little influence on their candidate zone can then be taken back if they are close enough to a selected zone.

Algorithm 3: Example of *new_search_zone()* evaluation.

Input: Point cloud of the scene $\mathscr{C}$, scale evaluated s_0, next scale s_1 and estimated correspondences *corr*

Output: New search cores $\mathscr{K}_{s_1}$ and new search zone $\mathscr{P}_{s_1}$

1 $\mathscr{P}$ = get_neighborhoods($\mathscr{C}$, *corr*, s_0); `// extract the zone given as possibly containing the object`

2 $\mathscr{K}_{s_1}$ = subsample(P, s_1); `// get the new set of zone cores`

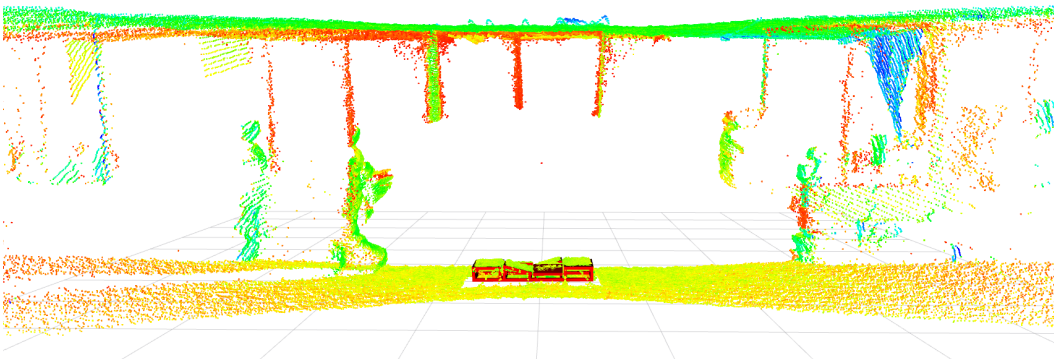
3 $\mathscr{P}_{s_1}$ = get_neighborhoods($\mathscr{C}$, $\mathscr{K}_{s_1}$, s_1); `// get the new zone`

3.7 Experimental results

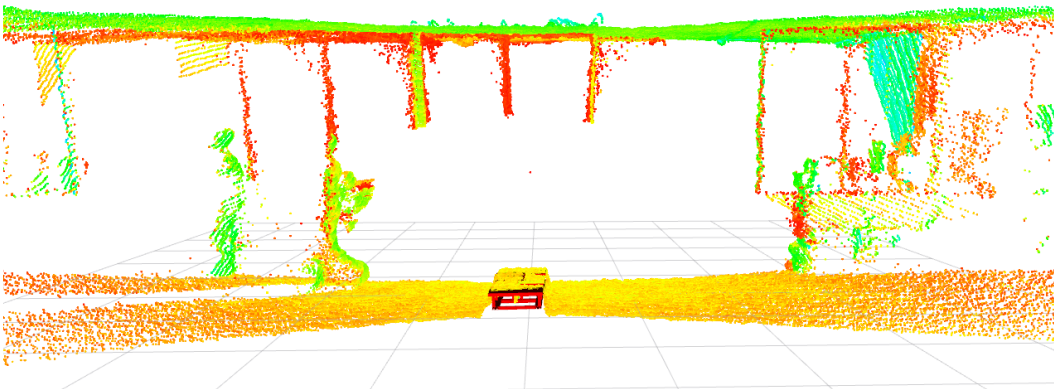
3.7.1 Context

The objective of this work is to localize an object in the environment using LiDAR measurement. Thus the tests are performed over multiples scenes as examples.

Firstly, scenes recorded by PAL-Robotics in their office are used. These scene have the LiDAR tilted, looking at the object on the floor. As the LiDAR has a horizontal FOV of 360°, the data contains also the ceiling and some other objects as robots, desks, etc. In these data, the object is in two different orientations: perpendicular or aligned with the LiDAR's field of view main axis. The test scenes can be viewed in the Figure 3.17, with the object represented.



(a) Aligned with the FOV main axis



(b) Perpendicular to the FOV main axis

Figure 3.17: Visualization of the test scenes for the platforms with the point cloud from the LiDAR colored by the intensity and the object represented in red.

3.7.2 First experiment: localization with fixed radius

In the implementation, we are using PCL version of FPFH. The computed histograms are of 33 bins, 11 bins by features.

3.7.2.1 Process

Before going through the "Scale drop object detection" defined at Section 3.6.3, it was decided to make a first experiment consisting of a single step of the localization with a fixed radius r.

In the scene, the point cloud is downsampled in order to reduce the computation time by reducing the number of FPFH calculated. This downsample is performed by a Voxel grid algorithm with a leaf-size of $r/2$ to completely cover the point cloud with the defined zones. The FPFH descriptor is then computed with a r-radius neighborhood for each point of the downsampled point cloud and compared to the cluster with the distance to the descriptors. The distance is computed as presented in Section 3.3.5.

3.7.2.2 Results

Figure 3.18 shows the result of the localization. The results are classified in "good/green" and "not good/red" by comparing them to a threshold of 1 (green: $\delta_{FPFH,p_i,p_j} \leq 1$, red: $\delta_{FPFH,p_i,p_j} > 1$).

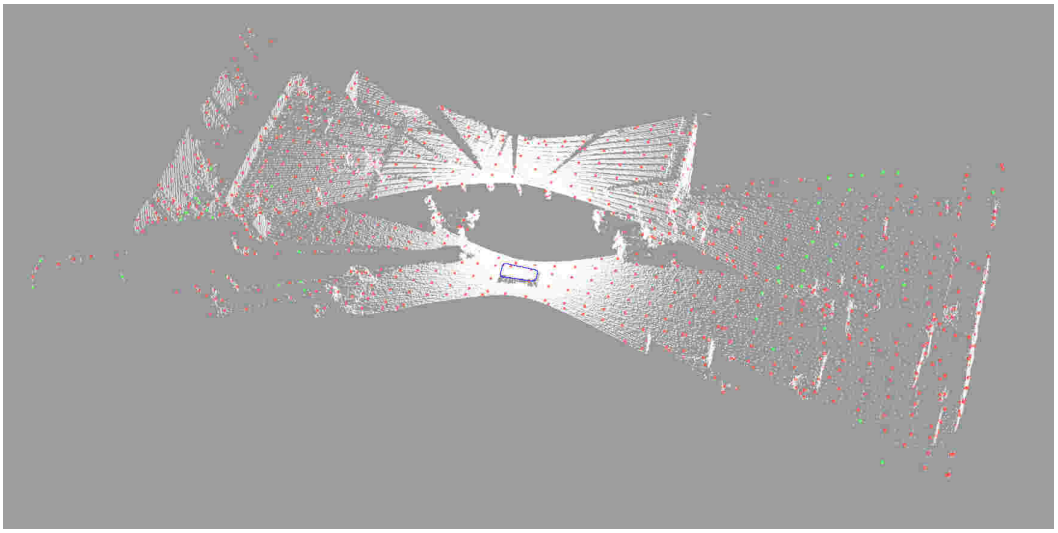


Figure 3.18: Visualisation of the rough localization with the scene points (white), the circled object (blue) and the evaluated points (red and green).

As can be seen in the Figure 3.18, the results are flawed: the points on and near the object are classified as "not good" and the only points considered "good" are in the remote parts of the data. It can also be seen that most of the "green" points are in areas with low density or with very weak geometric features.

These results rise many flags:

- the density of points in the descriptor vicinity is to take in consideration
- the threshold can be wrongly estimated and should be re-estimated
- the model composed of multiple views may present the same weakness as the scene (1st point)

3.7.2.3 Comparison of descriptors

In order to reduce the hypothesis on the failure presented in sec. 3.7.2, a comparison between the FPFH of the scene and the ones of the model was performed.

Scene descriptors As the FPFH near or on the object were not classified as interesting, they were compared to the descriptors of the model. For that, the model was manually aligned on the scene and a descriptor of the scene that is from the object was compared to the descriptors of the closest point of each view in the model. A "good" descriptor was also evaluated for comparison. Figure 3.19 shows the manual alignment result (red), the position of the point analysed from the object in the scene (yellow) and its affected correspondence in the model (blue). Figure 3.20 shows the position of the two points that was analysed in the global view.

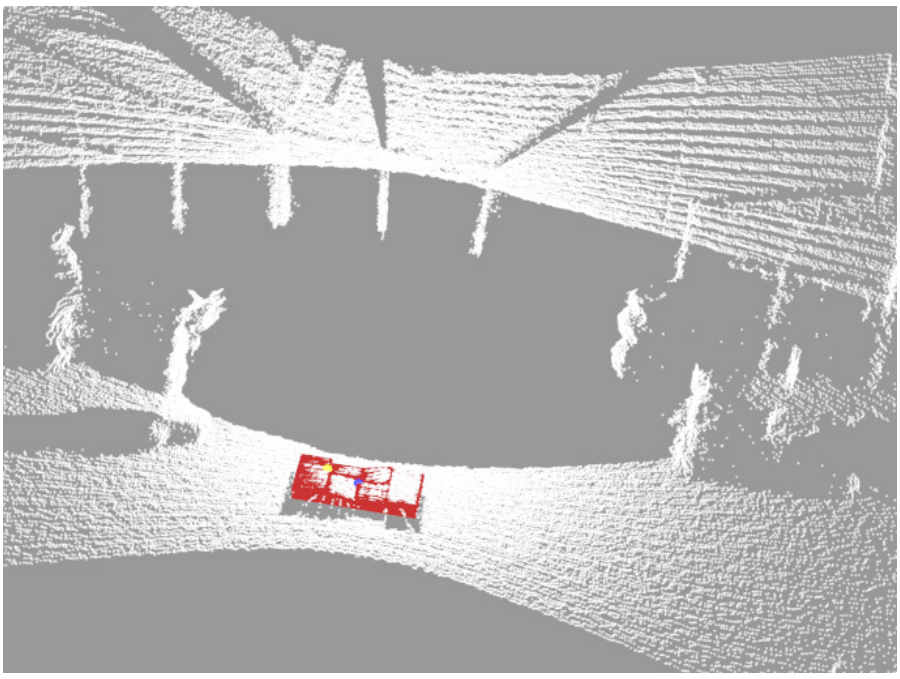


Figure 3.19: Visualisation of the model manually aligned to the scene.

Figure 3.21 shows the FPFH histograms of the inlier point (center one on Figure 3.20) and of the outlier point (right one on Figure 3.20). As shown, the two histograms are really different:

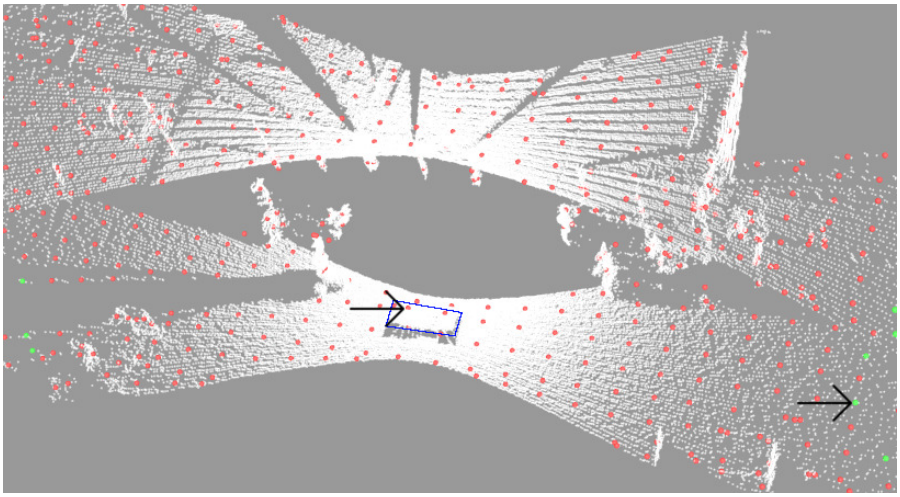


Figure 3.20: Visualisation of the scene with the two analysed points pointed.

- The inlier histogram (blue) shows three gaussian-like distributions, one for each feature,
- The outlier histogram (red) presents a single bin with a value close or equal to 100 for each feature.

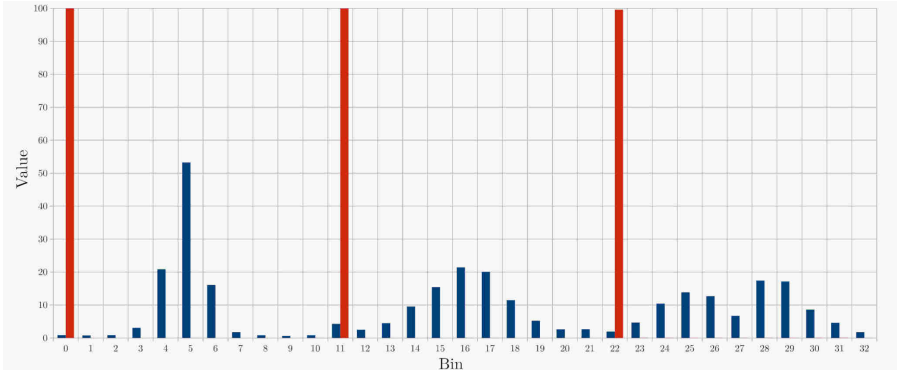


Figure 3.21: FPFH histogram of the inlier (blue) and the outlier (red) studied.
Inlier's peaks' values: bin 5: 53.26, bin 16: 21.44, bin 28: 17.43, bin 29: 17.15.
Outlier's peaks' values: bin 0: 99.97, bin 11: 99.96, bin 22: 99.61.

Object descriptors As the model have currently multiple views of the object, each view was searched for the closest point to the inlier chosen. Figure 3.22 show the distribution of the histograms of the points close to the inlier point.

As shown on figures Figure 3.21 in blue and Figure 3.22, the histograms in the model are closely related to the one on the scene. The next step is to look at the corresponding distances.

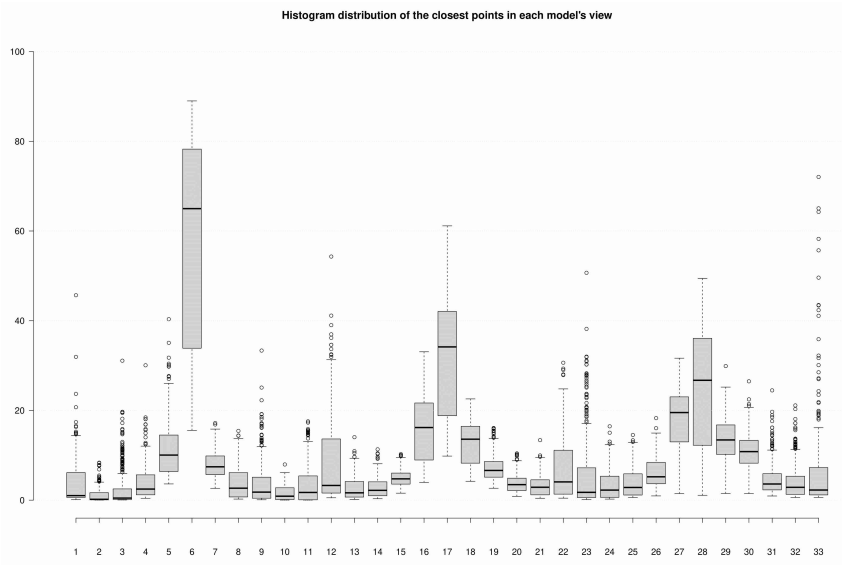


Figure 3.22: Distribution of the FPFH histograms of the closest point in each model-s view.

	Points' distances	**Histograms' squared distances**
Min	1,430E-05	352,317
Max	0,018	9315,321
Median	4,799E-05	2337,543
1st Quartile	3,274E-05	1488,539
3rd Quartile	1,584E-04	3388,805

Table 3.3: Distribution of the distance to the closest point in each view and the square distance to the corresponding histograms.

Table 3.3 resumes the distribution of the closest points in each view of the model and the distance of each corresponding histogram. Thanks to these observations and opposing it to the threshold defined on Section 3.7.2.2, it was decided to look more closely on the histograms corresponding to the outliers with a distance under the threshold of 1. This analyse has driven us to a failure in the model that is presented below.

Failure in the model Following the previous observations, a failure in the model has been found. It was due to Gazebo. To understand where it comes from and how it has affected the model, a reminder of the current model generation explained in Section 3.6 is necessary: the views are generated using a Gazebo simulation of our sensor. Looking closely to the generated views, a specific view is found where Gazebo seems to generate inconsistent points as shown on Figure 3.23 (red square). These points are defined with a y coordinate greater than 5m whereas the object is contained in a sphere of diameter 1.34m.

To resolve this problem, it was decided to remove the view from the model and check

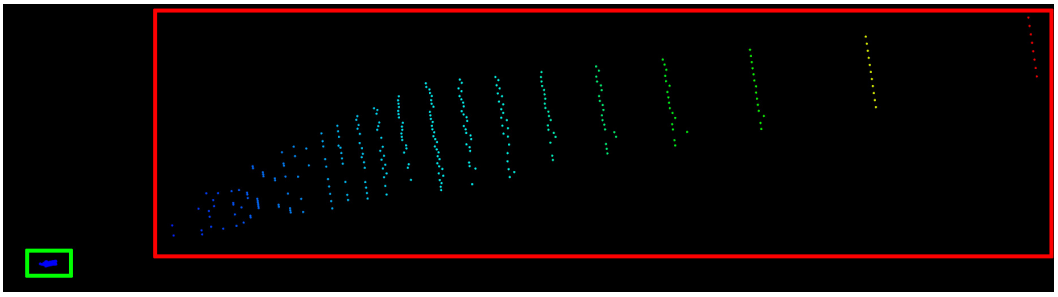


Figure 3.23: Points of the faulty view (green box: points of the object, red box: glitch points).

whether all the points in the other views are part of the original object. This problem is induced by the simulator and it would be interesting to analyse the defect and correct it in order to avoid its recurrence. However, this problem has not been analysed any further in this work.

3.7.3 Scale drop object detection

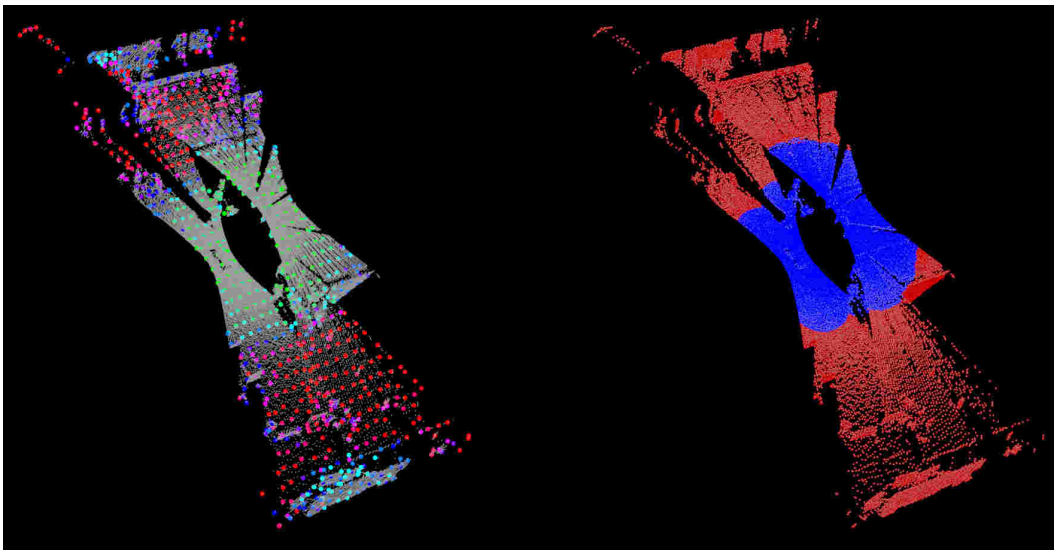
Once the failure in the model resolved, the work was concentrated on the system to localize the object with the multiscale logic: the Scale drop presented in sec. 3.6.3.

As presented previously, this system needs a clustering part in order to reduce the computation complexity. In the next experiment, a k-means system implemented in the Point Cloud Library is used to clusterize for a given scale all the descriptors of each views.

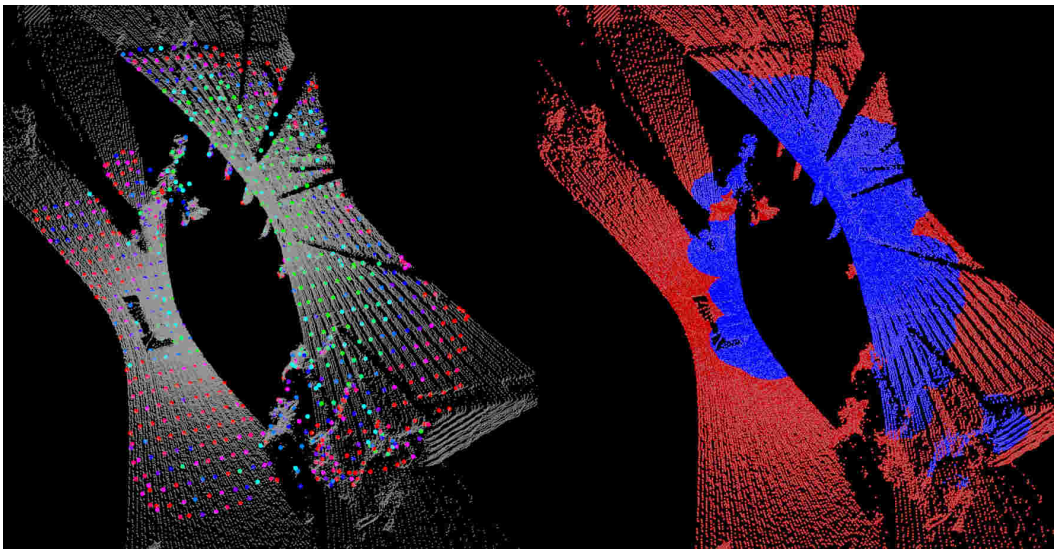
Figures 3.24 and 3.25 shows the result of a scale drop for the Memmo's platforms in the experimental room of PAL-Robotics. Left parts of the figures shows the original point cloud in gray with the candidates considered at each scale colored by their ranking. The green points are the 20% bests correspondences, then each consecutive 10% is respectively in turquoise, light then dark blue, dark violet to light violet and finally red. On the right side, the two zones defined at each scale are shown. The 10% best candidates and their neighborhoods are in blue, they constitute the selected search zone. The red part represent the candidates that have a lower correspondence score, they are therefore the rejected part. The object is located in the middle of the left part of each point clouds. It can be seen that the first scale $s_0 = 1.34$ has already reduced greatly the search zone to concentrate only on two zones. The first one which contain the object and a second one which is composed mainly of angular features such as the pillars nearby. The two next scale drops have reduced again this search zone mainly by removing some purely plane zones. On the scales figures 3.24b and 3.25a, the object searched was partly defined as rejected at the scale 0.67. At the scale 0.335, the object has been reassembled, due to the way the search zone is defined between two scales, as described in Section 3.6.3.

3.7.4 Enhancing the Scale-drop

After the tests presented above, and following the observations made while testing the system that is presented in Section 4.4, the failures of the scale-drop were evaluated in

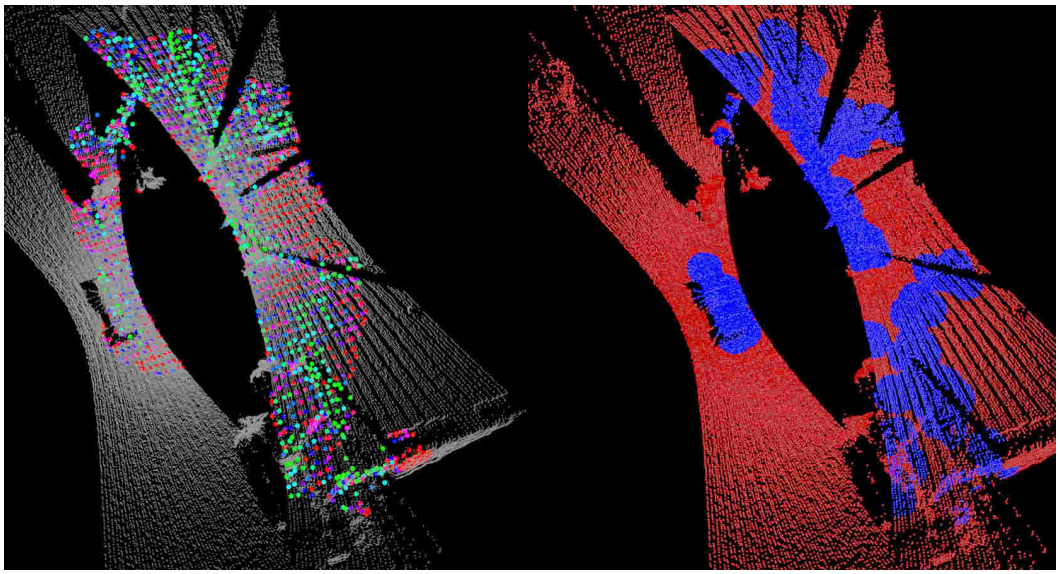


(a) scale 1.34

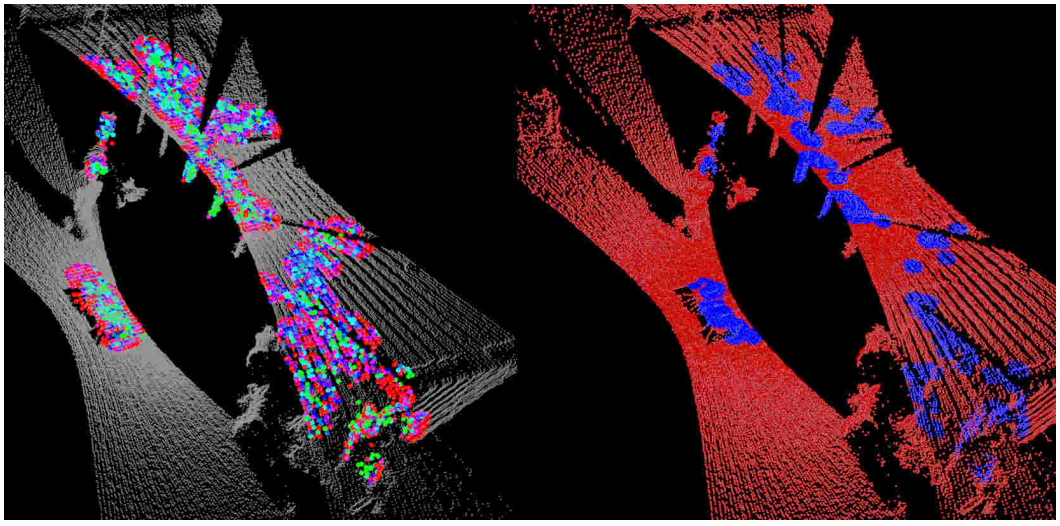


(b) scale 0.67 over scale 1.34 selection

Figure 3.24: Visualisation of a scale drop for the Memmo's platforms, with $k = 100$ clusters and from scale $s_0 = 1.34$ to scale $s_1 = 0.67$.



(a) scale 0.335 over scale 0.67 selection



(b) scale 0.1675 over scale 0.335 selection

Figure 3.25: Visualisation of a scale drop for the Memmo's platforms, with $k = 100$ clusters and from scale $s_2 = 0.335$ to scale $s_3 = 0.1675$.

collaboration with Weikang Zeng during his Master's Internship.

3.7.4.1 Observations on initial results

The results shown in Figure 3.24 and 3.25 are encouraging because they show that the system gives candidates zones containing the searched object. However, this test case can be considered an easy one. Indeed, the object is seen in its entirety from the top, and at a distance similar to the one used in the simulation to build the model. When testing with more challenging scenes, for example where the object is seen from the top but with missing parts, the results are not good, as shown in Figure 3.26. This figure shows the result of the first scale test with each color points being the center of a candidate zone and the color representing the ranking place. The ranking is defined as a percentage over the distance between the candidate descriptor and the closest descriptor in the model. The lower the distance, the more the descriptors match. Thus, 0% corresponds to the candidate with the lower distance and are shown in *green*. 100% are the candidate with the greater distance and are shown in *red*. In the figure, it is noticeable that the candidates on the object are mainly in *blue*, meaning that they are around 40% to 60%.

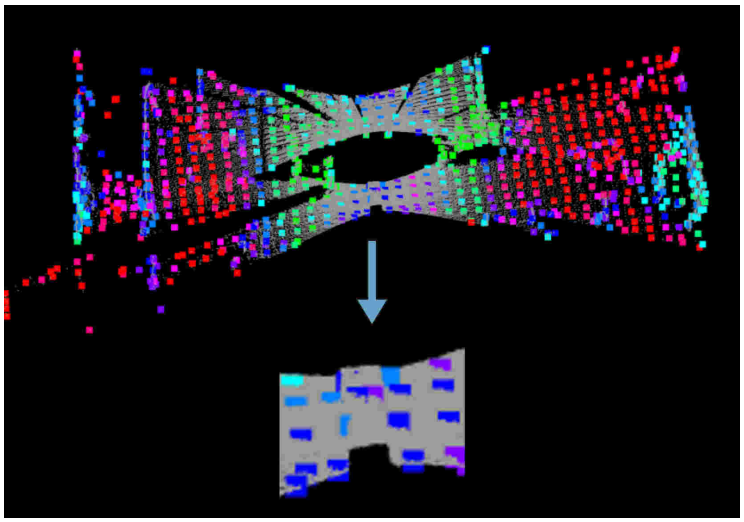


Figure 3.26: Result of the first scale test on the measure with occluded part.

While discussing about the result, it appears that the clustering step may be tuned as it was arbitrarily setup.

3.7.4.2 Optimising the clustering step

The first improvement of the system on which W. Zeng has worked was the clustering step. The clustering step principle is to group the elements of a set into smaller consistent sub-sets that can be further reduced to a single representation for the whole sub-set. The main parameters are the number of clusters to determine and their initial position. Previously, the number of clusters was established arbitrarily by the user and the clusters positions were initialized randomly by the system.

In the state-of-the-art, it exists a method to optimise for a dataset the number of clusters to use. This method is called the *Elbow Method* [Tibs 01]. As inputs, it takes the set of data and the algorithm used to clusterize it. Then, it computes the clustering for a predefined set of number of clusters going from 1 to a determined end $k_{clusters,max}$. At each step, it computes the "Sum of Squared Errors" (SSE), often called *inertia* in implementations, as:

$$\mathscr{E}(\mathscr{D},k) = \sum_i \|\mathscr{D}_i - \mathscr{K}_k(\mathscr{D}_i)\|^2 \tag{3.31}$$

with $\mathscr{D}$ being the data clustered and $\mathscr{K}_k(\mathscr{D}_i)$ the function that provides the corresponding cluster of the given data when clustered with k clusters. The closest the number of clusters is from the number of data in the set, the smallest $\mathscr{E}$ is. Thus, an optimal value of the clustering can be defined as being the inflection point of the result of $\mathscr{E}(\mathscr{D},k)$ with $k = [1 : k_{clusters,max}]$.

One of the limitations of this a method is the computation time. Indeed, this method needs to be computed with a maximal number of clusters $k_{clusters,max}$ high enough in order to determine the inflection point of $\mathscr{E}(\mathscr{D},k)$. Figure 3.27 shows the result of the application of the "Elbow Method" to a set of 1910 descriptors for five values of $k_{clusters,max}$: 5, 10, 20, 50, 100. It shows that, if $k_{clusters,max}$ is too low, the inflection point may be missed. As the clustering step needs to be computed for each k-value, time increases with the size of the set. Moreover, the clustering step time consumption is increasing with the increase of the dimension of the data and the size of the set. As an example of the time consumption of such a method, on a 4 cores, 8 threads Intel Core i5-8400H, it takes 19s to compute the SSE values for $k = [1 : 100]$ with 1910 descriptors and 6min 16s with 130518 descriptors.

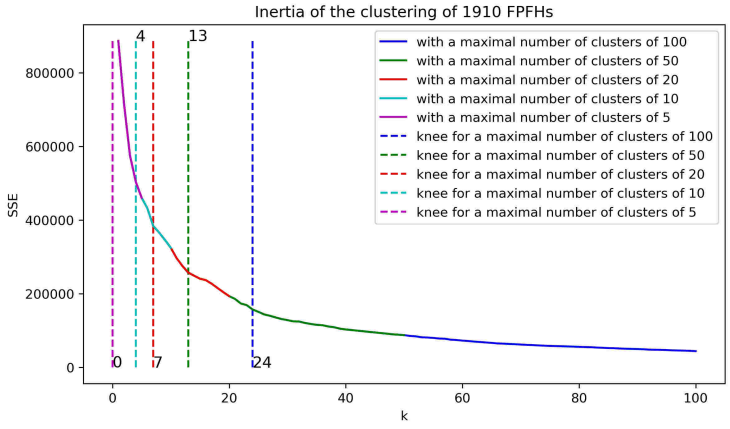


Figure 3.27: Result of the "Elbow Method" on a set of 1910 FPFH descriptors with various values of $k_{clusters,max}$. In the case of $k_{clusters,max} = 5$, no inflection point was found by the *kneed* python implementation.

This method was then applied offline on our model to determine the parameters of the clustering step for the algorithm. As its application was offline, it was decided to not take into account of the computation time, thus, $k_{clusters,max}$ was set to 100.

Moreover, in order to enhance the clustering result, W. Zeng proposed to use the K-

means++ method instead of the K-means algorithm. K-means++ tends to improve the cluster selection of K-means by defining that the clusters should be from the set studied while in K-means, the clusters are defined as part of the space but not specifically from the set.

3.7.4.3 Model clustering for "large" scales

The second improvement way evaluated was for the "large" scales. These scales were defined as the ones where the information can be considered global, meaning that the features in each views are not discriminant enough to differ a view from the rest of the model.

For these scales, it was decided to change the clustering step. Previously, this step was clustering each view separately, thus obtaining a new set of descriptors consisting of k_i descriptors for each view i. Thus, having n views in the model, the clustered set size is $\sum_i k_i$. E.g. for the platforms presented previously, each view at the greater scale of 1.34m have an mean optimal k of 15, it then can be expected to have approximately 6585 clusters with the initial clustering step.

This set presents some redundant clusters because the model has views that can be equivalent due to close poses of the sensor or regional similarity on the object, e.g. due to a symmetry of the object. It was decided to change this step for a global clustering step. Instead of clustering the points for each view, it will now cluster all the descriptors of each view in the model obtained at a given scale. This is expected to reduce the number of clusters from $\sum_i k_i$ to a single $k_s << \sum_i k_i$, k_s computed thanks to the "Elbow Method" presented before. Taking back the example of the platforms at the scale of 1.34m, the new method then have an optimal number of clusters close to 20, reducing greatly the number of clusters.

Reducing this clusters set implies a reduced computation time of the correspondence estimation.

3.7.4.4 Improvements results

These improvements were evaluated on the ranking result of the candidates point on the object for the first scale. Table 3.4 summarize the ranking attained with the original solution, each of the solutions applied alone and both solutions combined.

scene	Method			
	Original	*Elbow Method*	*All In One*	Both
scene_side	$30 \sim 50\%$	$10 \sim 15\%$	$10 \sim 15\%$	$9 \sim 14\%$
scene_front	$40 \sim 60\%$	$25 \sim 40\%$	$25 \sim 37.5\%$	$12.5 \sim 15\%$

Table 3.4: Ranking of the object's candidate for each method on two different scenes.

This improvement in the ranking from 40% to 15% is interesting because the next step is to select the best candidates and it was decided to make the selection over the ranking place with a certain percentage of the candidates selected. However, the original solution was using a really narrow selection threshold of 10%, thus, some candidates on the objects

were left over. W. Zeng proposed to increase this threshold to at least 15% such as to hope keeping the object's candidates but not enhancing widely the data to analyse at the next step.

In order to recapitulate the improvements proposed by W. Zeng on the model, Algorithm 1 can be modified to become Algorithm 4. The first scale is clustered with the method presented in Section 3.7.4.3. The following scales are reduced with the clustering of each view separately.

In the algorithm:

- *elbow_method(D)* is the method applying the Elbow Method to determine the optimal number of clusters
- *clusterize(D, k)* is the clustering step that previously was using the K-means algorithm but can be modified to use the K-means++ algorithm as proposed by Zeng

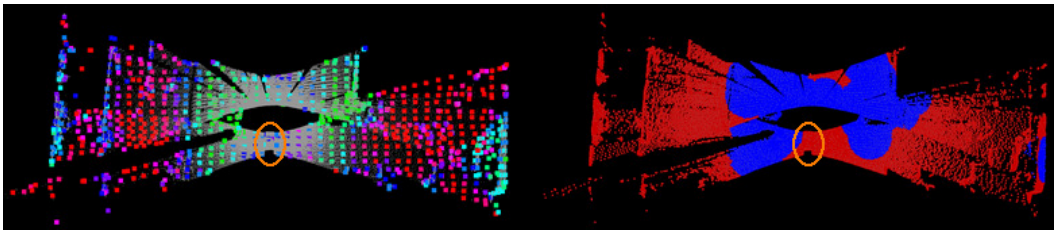
Algorithm 4: Generation of a model enhanced with the propositions of Weikang Zeng

Input: Set of views of the object as point clouds V, Set of scales S
Output: Multi-view and multi-level model

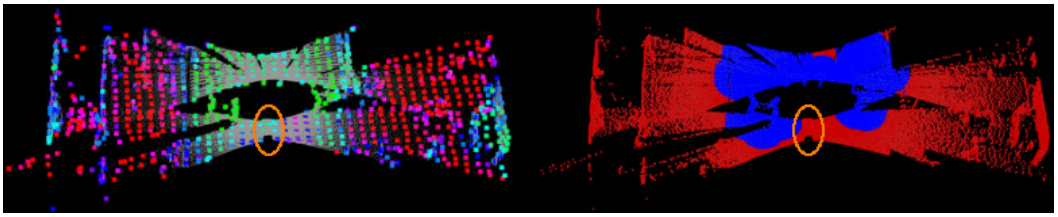
```
/* Computation for the greater scale: clustering all the views in one set. */
```
$D_{s_o} = []$
foreach *view* V_j *in* V **do**
    $D_{V_j,s_0} = \text{FPFH}(V_j, s_0)$
    D_{s_0} += D_{V_j,s_0}
$k_{s_0} = \text{elbow\_method}(D_{s_0})$
$K_{s_0} = \text{clusterize}(D_{s_0}, k_{s_0})$
$\text{save}(s_0, D_{s_0})$
```
/* Computation for the other scales: clustering each view separately. */
```
foreach s_i *in* S **do**
    foreach *view* V_j *in* V **do**
        $D_{V_j,s_i} = \text{FPFH}(V_j, s_i)$
        $k_{V_j,s_i} = \text{elbow\_method}(D_{V_j,s_i})$
        $K_{V_j,s_i} = \text{clusterize}(D_{V_j,s_i}, k_{V_j,s_i})$
        $\text{save}(V_j, s_i, D_{V_j,s_i}, K_{V_j,s_i})$

The last improvement proposed, using K-means++ method instead of the K-means, was only applied to the large scale's clustering step but was planned to be applied to the other scale's ones.

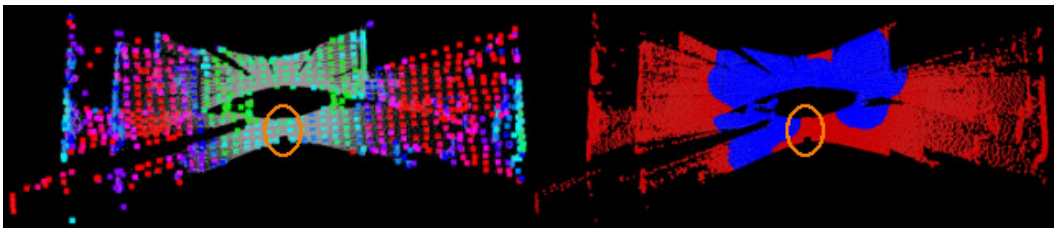
The evolution of the Scale-Drop first scale evaluation with the improvements presented is shown in Figure 3.28. On both figures, the left part shows the candidates considered and the right part the selected candidates. On both parts, the object is circled in orange to help visualize the result improvement. We can see on right parts, that only by combining the three improvements proposed the system will consider the object as a good result.



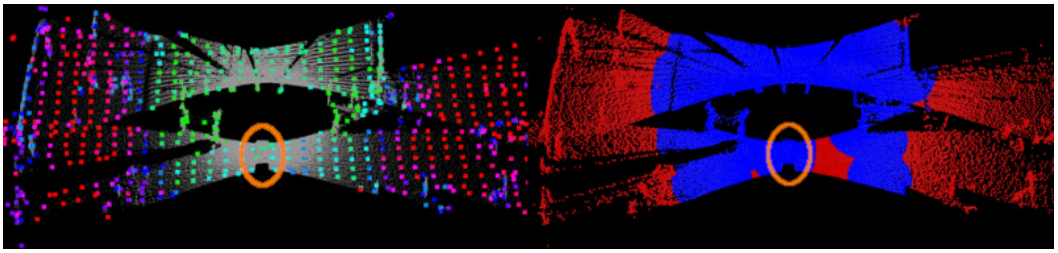
(a) Initial result



(b) Elbow Method



(c) Elbow Method and All In One



(d) Both three improvements

Figure 3.28: Results of the Scale-Drop first scale evaluation with each improvement.

However, as summarized in Table 3.4, we can observe on the left parts the ranking improvements for the candidates close and on the object, going from dark blue corresponding to approximately 50% to a clear light green corresponding to approximately 10 to 20%. Then, elevating the selection threshold from 10% to 15% permits to consider the object for the next steps.

3.7.5 Long distance experimentation

The improved system was also tested on other tests with the object further away from the sensor. In the scenes shown previously, the distance between the sensor and the object is approximately of 1.35m. This distance is similar to the simulated object where the sensor is

at 1.34m of the center of the bottom face of the object. To determine wether the same model can be used to look for the object with a greater distance, measurements were performed where the object is up to 4.5m away from the sensor. As an example, Figure 3.29 shows the visualization of a test with the object viewed lying on the floor from 4.5m. Figure 3.30 and 3.31 show the result of the scale drop for the scales of $[1.34, 0.67, 0.335, 0.1675]$ for this example. In Figure 3.30 and 3.31, the object is circled on the right part of each view and it can be observed that the candidates on the objects keeps a ranking enough to continue the evaluation and not reject the majority of the object. At the last scale evaluated, half of the point measured on the object of interest are kept in the candidate zone.

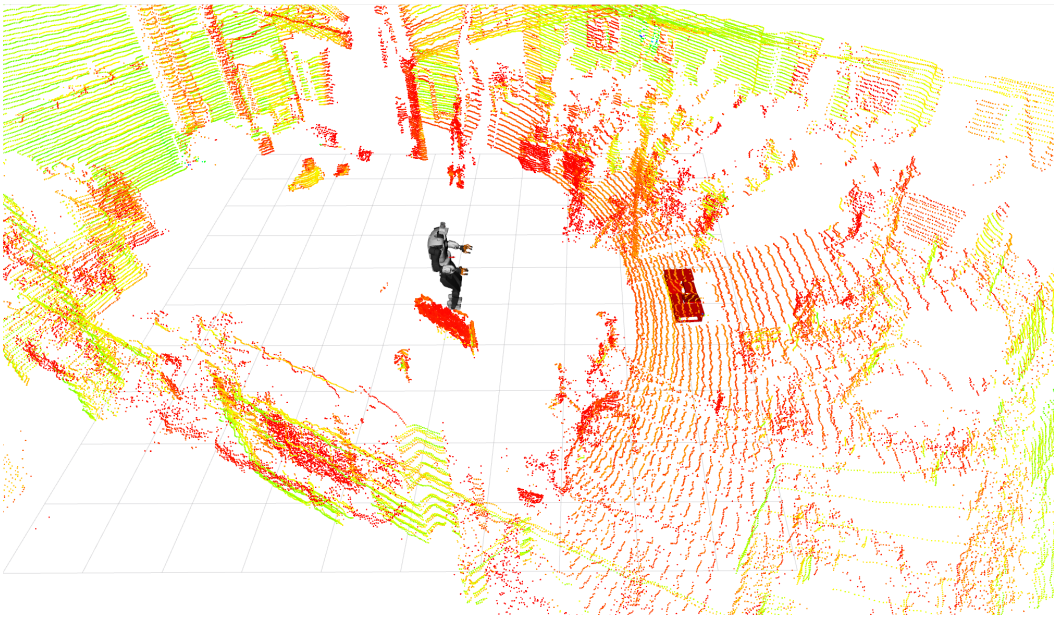
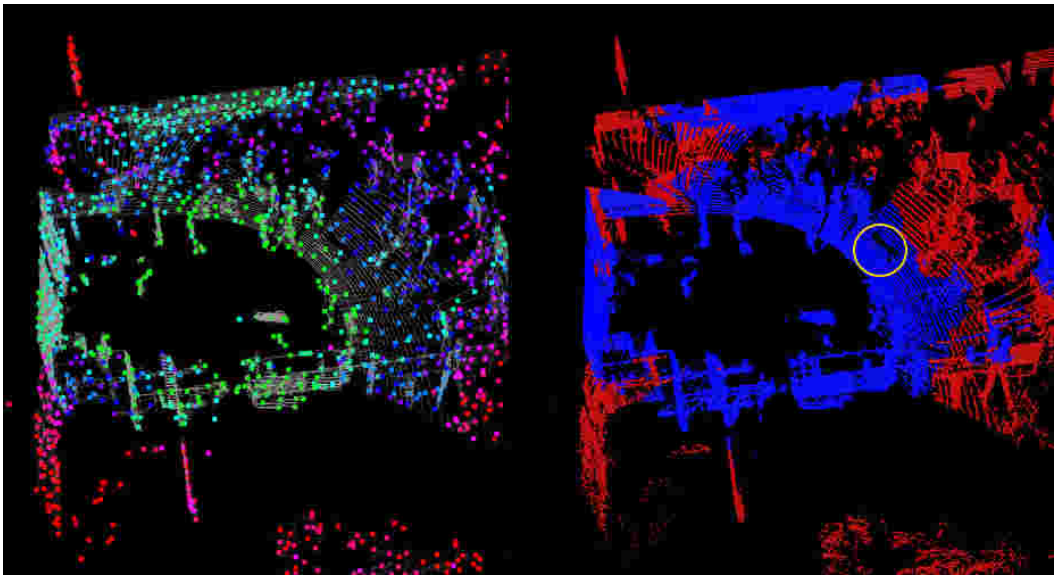


Figure 3.29: Visualization of the long distance scene presented, with the robot and the object represented.

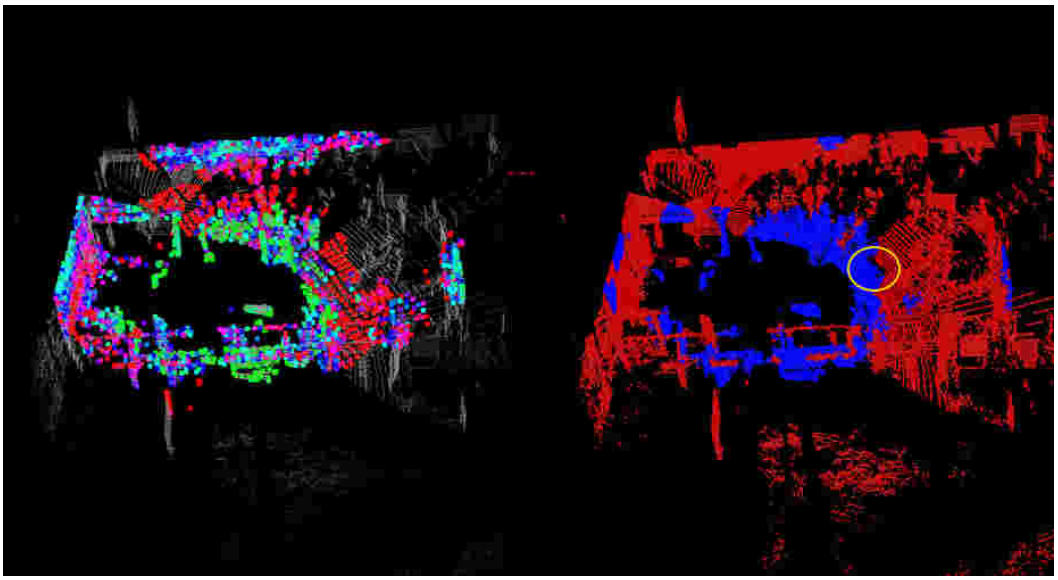
3.8 Conclusion

In this chapter, the development of a system to localize object in LiDAR data trying to overcome the low density of information and occlusion problems was presented. This method uses a state-of-the-art descriptor, a multi-level analysis and a multi-view simulated model. The results are encouraging and shows an ability to determine candidate zones in a measured scene for the object to be localized.

However, more developments are needed to be able to return the estimated pose of the object but bases for this development have been laid. Moreover, the tests were only performed on a single but difficult object: an object composed of a set of tilted plans. The difficulties of this object lies in a lack of geometric information as it is highly similar to a



(a) Scale 1.34



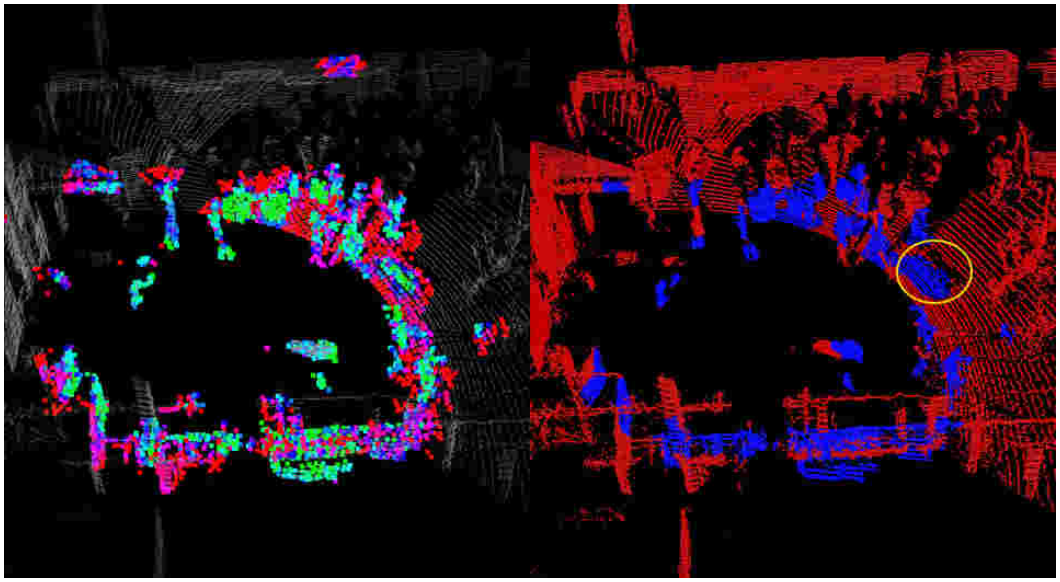
(b) Scale 0.67

Figure 3.30: Results of the Scale-Drop with the object at 4.4m of the sensor.

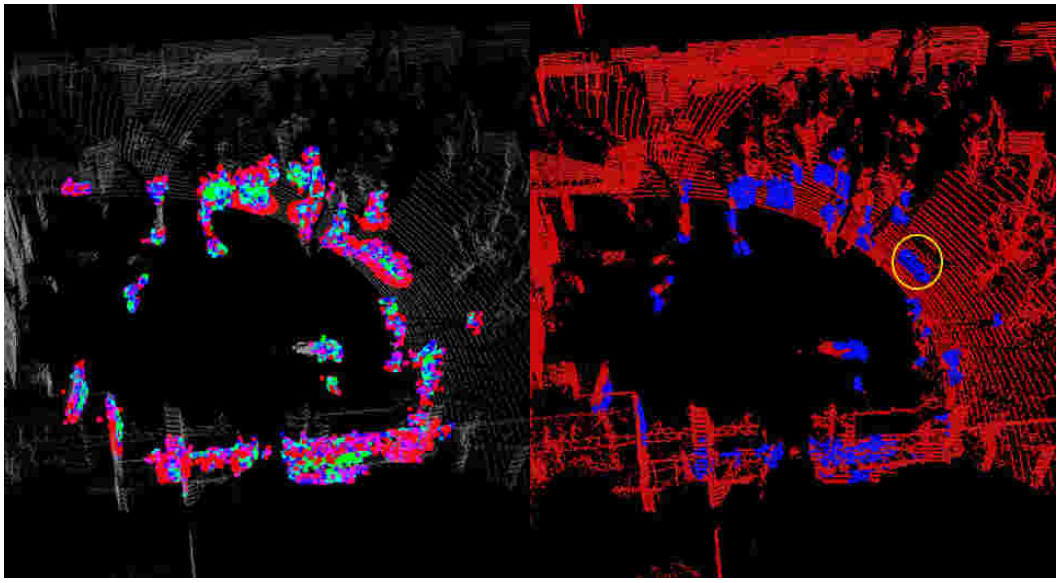
step of 1.23×0.5m with an axial symmetry. We want to test this system over some other objects, such as an aircraft engine mast or other objects to determine the limits of this work according to the object searched and the environment.

The code developed in this part is open-source and available at https://gitlab.laas.fr/tlasguigne/multiscale-fpfh.

In the computer vision domain, a lot of research are working on the use of Deep Learning or Reinforcement Learning solutions. The method studied in this chapter is not using such systems but it seems possible to use it for supervised learning in order to develop a



(a) Scale 0.335



(b) Scale 0.1675

Figure 3.31: Results of the Scale-Drop with the object at 4.4m of the sensor.

network performing object localization in LiDAR data.

In the next chapter, another localization system based on the same state-of-the-art descriptor is discussed, however it will no longer be about localizing an object but about localizing the robot in a known environment.

CHAPTER 4

FPFH-based SLAM initialization

4.1 Introduction

Chapter 2 describes a LiDAR-based localization system implemented on the robot allowing a precise estimation of the robot localization in a known environment. A limitation of this system is the need for an estimate of the robot initial pose. When the visual-inertial odometry or the base estimator is started, each makes an initial assumption about their world origin.

Thus, the pose ${}^{M}p_0$ of the robot in the environment when the localization system is started needs to be estimated in order to initialize the SLAM system by providing the estimation in the known world. In the experiment presented in Chapter 2, an approximation of the Mocap estimate, that was used as ground truth, was used as the initial pose.

This problem of knowing the initial pose of the robot in the environment is called the *kidnapped robot problem*. This problem is equivalent to the place recognition problem that has attracted a great deal of interest in the recent decades.

In this chapter, we will discuss the research that has been performed with Guillaume Gobin during his Master internship under the supervision of Olivier Stasse and myself. The objective is to resolve the place recognition problem in a known indoor environment using measurements from a LiDAR. Later, we will discuss the efficiency of our solution when trying to apply it on the object localization problem presented on Chapter 3.

4.2 Related work

The place recognition problem has been widely studied in the last years. Its formulation is dependent of the environment scale, the sensors used, the application, etc.

There are two common choices as visual sensors: camera and LiDAR. Each sensor has its pros and cons as summarized in [Barr 21]. For example, cameras have a limited Field of View (FoV) whereas nowadays LiDAR can have a 360° FoV on one axis. However, in terms of cost and energy consumption, cameras are often cheaper and less energy-consuming than LiDAR. The target environment needs to be taken into account as cameras are dependent of textures whereas LiDAR mostly relies on reflective materials. Other sensors overcome the dependance to textures of monocular cameras by being able to produce depth information, either with stereo-vision or infrared projector and cameras. However, these depth information are less precise than LiDAR's. With the evolution of technology and particularly MicroElectroMechanical Systems (MEMS), cameras integrate

LiDAR systems such as the IntelRealSense L515 LiDAR Camera [Inte b], enhancing the range and precision of the depth information.

Depending on the sensors used, different features are used. In case of a camera, the system may be using SIFT [Lowe 99] or ORB [Rubl 11] whereas Scan Context [Kim 18] or FPFH [Rusu 09] and others 3D descriptors may be used in case of using a LiDAR.

Other method such as in [Komo 21b] try to combine both worlds instead of using only one of these visual sensors. In this work, features are extracted from the LiDAR using MinkLoc3D [Komo 21a], presented on Section 4.2.2, and from the monocular with a part of ResNet18 [He 16]. These two features are lately aggregated in a single one which is matched to a database.

The LiDAR-based methods may differ in the type of information processed.

4.2.1 Segment-based methods

[Doui 12] presented the concept of segments or how to use the segmentation of a point cloud to reduce its computational cost.

This concept is used by [Dub 17] and [Dub 18] to perform place recognition and SLAM. In both works, the point cloud is segmented and features are extracted for each segment. Then, using these features, the segments are matched to known ones from a map in a learning approach. These matches are then given to a geometric verification system based on RANSAC to evaluate the consistency of each match.

On this same principle, [Tinc 19] applies a similar method but the feature is learnt and the final pose is estimated with PRObabilistic SAmple Consensus (PROSAC) [Chum 05] instead of the usual RANSAC method. The main advantage of their method is the design of the network that is able to run on a single CPU whereas numerous network-based method needs a GPU to be time-efficient. [Rame 20] presented an online LiDAR-based SLAM system combining the Autotuned-ICP (AICP) [Nobi 17] for point cloud matching, and the Efficient Segment Matching (ESM) [Tinc 19] to detect loop-closure.

4.2.2 Geometry-based methods

The main characteristic of LiDAR is the ability to provide a measure of the environment as a 3D point cloud, giving strong geometric information. Whereas methods divide the point cloud into segments, others take the complete point cloud and use its geometric information to compute local and global features.

[Kim 18] presented a new descriptor called Scan Context, a 2D-shaped descriptor used to describe 3D LiDAR scans. This descriptor divide the space into azimuthal and radial bins, similarly to [Belo 02], with each bin being represented in the descriptor by the maximum height of the points in the space division. A way to compute the similarity between the descriptors in order to be robust to the sensor's orientation is also presented in the paper. This descriptor is then used to detect loop-closure.

[Elba 17] presented an algorithm for the registration between a local point cloud and a large-scale point cloud. The system is based on the selection of local subsets, called super-points, which are described with unsupervised auto-encoders. The super-points from the

local measure are matched to the ones of the large-scale point cloud acting as base. These matches are then used to make a coarse registration, later refined by ICP.

[Qi 17a] proposed PointNet, a deep neural network taking raw 3D point clouds and learning local features. The initial work focus on classification and segmentation tasks but it has opened the door of deep neural network to point cloud based systems.

[Uy 18] presented a place recognition solution based on PointNet and NetVLAD [Aran 16]. The first one is used to extract local descriptors, shortened of its maxpool aggregation layer used for its original purpose. NetVLAD is then fed with these descriptors in order to extract a global descriptor.

[Komo 21a] presented an alternative approach called MinkLoc3D. This approach uses a convolutional neural network to compute local descriptors. A generalized-mean (GeM) pooling is then used to produce global descriptors.

The global descriptors obtained are used in a base-query manner. The measurement, called query, is described and compared to the descriptors available in a database.

4.2.3 Intensity-based methods

The geometry-based method may lack the appearance information available in camera-based systems. Most LiDAR provides, in addition to geometric information, intensity information that depends mostly on the texture and material of the object. Figure 4.1 shows an intensity image that can be acquired with the Ouster OS1-64 set at 1024 samples per turn. Thus, works were proposed combining this intensity information with the geometric one.

Figure 4.1: Intensity image reconstructed, the data is from an experiment of Chapter 2, it has been divided in two parts for visibility purposes.

[Guo 18] proposed a new descriptor, called ISHOT, combining the geometric information, represented by a SHOT descriptor [Tomb 10b, Tomb 11], to the intensity. This descriptor is then used in a place recognition algorithm for local descriptor evaluation and mapping.

[Shan 21] proposed to use the vision-based ORB descriptor of [Rubl 11] applied to the LiDAR intensity. The ORB features are then converted into a bag-of-words vector using DBoW [Galv 12] and compared to a database.

4.2.4 Attention-based methods

Point Contextual Attention Network (PCAN) [Zhan 19] is a neural network adding an attention map in the process. The system extract local features using PointNet. Then these features are fed to PCAN that output a per-point attention map, used to tune NetVLAD which is used to aggregate the local features into a global descriptor.

[Xia 20] proposed a self-attention and orientation encoding network (SOE-NET). They integrate an orientation-encoding process into PointNet to extract local features. A self-attention unit influencing the aggregation performed by NetVLAD is applied on the set of features to extract a global descriptor of the input point cloud.

4.3 VLAD-based solution to the kidnapped robot problem

In this section, we discuss the solution proposed by Guillaume Gobin during his Master's internship on how to combine the FPFH descriptors, discussed on Chapter 3 Section 3.3.2, and machine learning techniques used in computer vision. We also discuss the results obtained and some observations made during this work. More details on what is presented can be found in Guillaume's internship report titled "Localization of the TALOS humanoid robot with geometric and visual information combined to learning" [Gobi 21].

4.3.1 Pipeline

The proposed solution is based on the pipeline of PointNetVLAD [Uy 18]. During a first step, it creates a local descriptor and during a second step it creates a global signature. In our case, it was decided to use the FPFH at a point p as the local signature $\mathscr{L}(p)$. Two signatures were tested for the global signature $\mathscr{G}(C_s)$ of a cloud C_s. One based on the computation of a normalised sum of the local descriptors, the second one using the VLAD encoding over the FPFH.

4.3.1.1 FPFH average signature

The first method defined was to group the FPFH computed on each points of the scene in a global one. Guillaume Gobin proposed to define the signature in a bi-channel way composed of:

1. the normalised sum of the descriptors
2. the standard deviation of this normalised sum

Given C_s a point cloud and a normalisation function $Normalize()$, the formulations of the normalised sum and its standard deviation are respectively presented in Eq. 4.1 and Eq. 4.2.

$$\mathscr{S}(C_s) = Normalize\left(\sum_{p \in C_s} \mathscr{L}(p)\right) \tag{4.1}$$

$$\sigma_{\mathscr{S}}(C_s) = \sqrt{Normalize\left(\sum_{p \in C_s} \mathscr{L}(p)^2\right)} \tag{4.2}$$

The global signature is then:

$$\mathscr{G}(C_s) = \begin{bmatrix} \mathscr{S}(C_s) \\ \sigma_{\mathscr{S}}(C_s) \end{bmatrix} \tag{4.3}$$

4.3.1.2 VLAD encoded signature

The VLAD encoding is based on a dictionary of words. To obtain this dictionary, it has been decided to use a k-means algorithm.

Given k centroids, the k-means algorithm is performed over the sets of local descriptors of the different views from the training-base used to build the codebook. Once the dictionary obtained, a view is classified by assigning each local descriptor to a word c_i in our codebook. Then, for each word of the dictionary, the distances to its associated descriptors in the measure is accumulated as d. These couples (c_i, d) for each word in the codebook are our global descriptor.

The word association is performed thanks to a *Nearest Neighbor* algorithm applied on the codebook with the distance defined as an Euclidean distance of FPFH, presented in Section 3.3.5.

The final complete pipeline is presented in Figure 4.2.

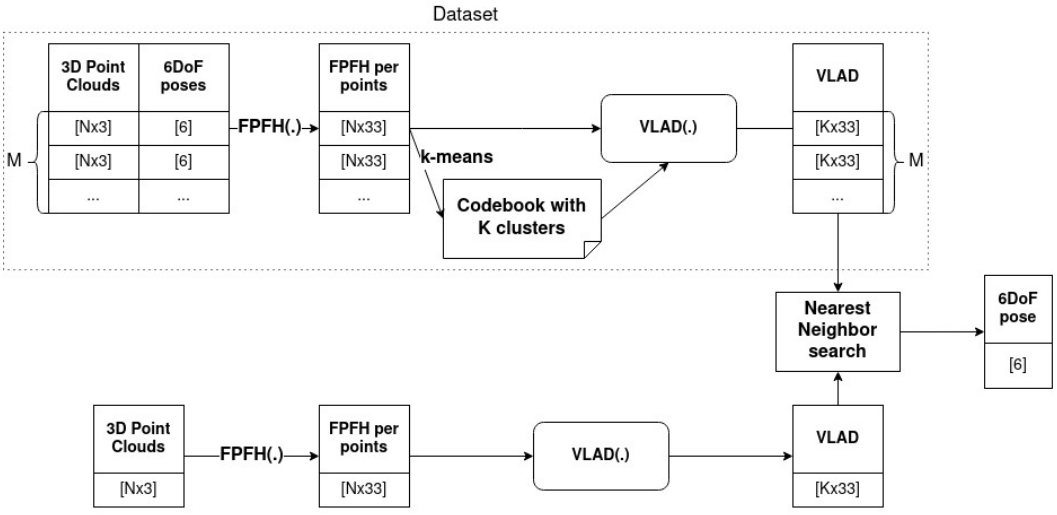
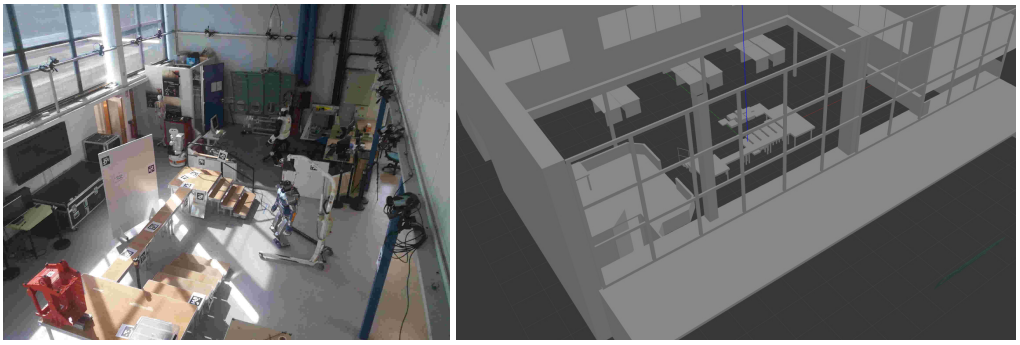


Figure 4.2: Final architecture of the system.

4.3.2 Databases

There are 2 databases that were produced. The first one was produced using a simulated version of the Bauzil Room. A second dataset has been built in a real setup. Figure 4.3

shows the real Bauzil Room in 2021 and a simulated version viewed from different points of view.



(a) Real (b) Simulation

Figure 4.3: The Bauzil room and its simulated version in Gazebo.

4.3.2.1 Simulated sets

For the simulated experiments, the middle platforms with the stairs were removed from the reality and the simulation. The simulation uses a modeled noiseless LiDAR on the architecture of the Ouster OS1-64. However, a noised LiDAR, modeled with the data provided by the fabricant could also be used. The training-base has been generated with the robot placed along a (x,y) grid. The test-base has been built with the robot randomly placed in the room. Both bases are generated with the sensor at a constant height of 1.65m, the approximate heigh of the sensor on the robot when in its half-sitting pose. Figure 4.4 shows examples of the distribution of the poses from the databases. In the figure, the databases sizes have been reduced for visualization purposes. Each smaller axis represent a pose present in the database. The training-base shown is generated over a 8×11m grid graduated every meter. The test-base is generated with 50 random poses from the same 8×11m space.

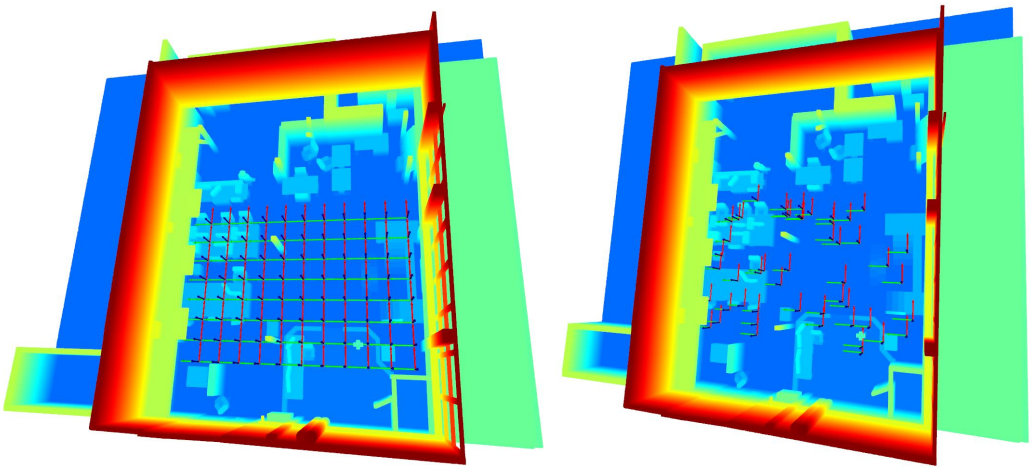
In the generation of the simulated training-bases and test-bases, the orientation is left aside. Indeed, the LiDAR has a 360° horizontal field of view. Moreover, when we start the robot, it is assumed that the LiDAR (x,y) plane is parallel to the world's (x,y) plane.

In the case of a noiseless sensor, the data is really smooth and the FPFH descriptors are highly discriminating as shown in Figure 4.5 by the clustering of the FPFH in 4 groups. In this example, the FPFH were computed with a 50cm radius and a limit of 4000 used neighbors.

4.3.2.2 Reality sets

The robot has been put in different places of the Bauzil room with its poses estimated using the available Motion Capture system. Thus, the zone measured has been limited by the zone covered with the Mocap cameras. Again, the measures for the training-base were taken with an approximate 1m grid on (x,y) axes. Whereas the test-base is taken with

random (x, y) positions. Moreover, the clouds were captured on positions where the robot was standing still for 10s to avoid noise from the robots movements.



(a) training-base (b) Test-base

Figure 4.4: Visualization of the poses generated distribution. The number of poses is reduced for visualization purposes.

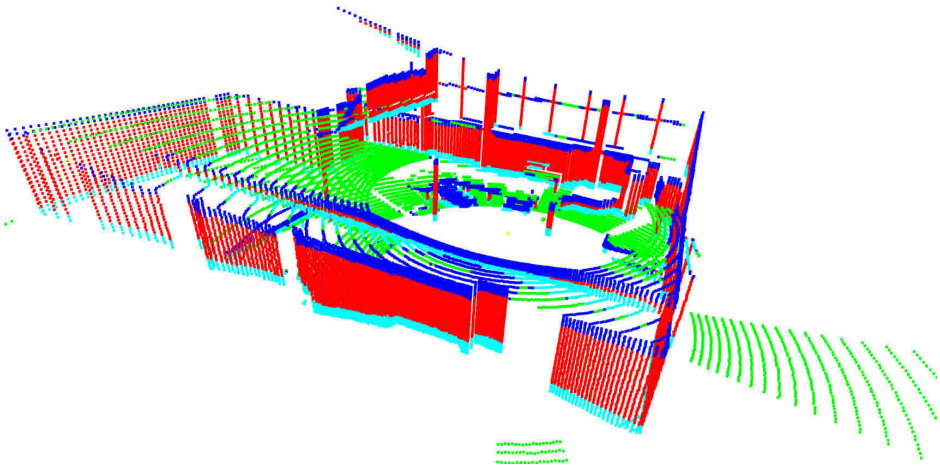


Figure 4.5: Visualization of a simulated cloud clustered in 4 groups.

4.3.3 Results

This solution has been tested on the two generated datasets, each divided in a training-base and a test-base. Each couple consists of a training-base and a test-base. Thanks to these bases, three evaluations were performed, they are summarized in Table 4.1.

To define the results, it is proposed to compare 3 positions, visible in Figure 4.6:

***Ground Truth (GT)* position** the exact position of the robot

***Nearest* position** the nearest position existing in the training-base, considered as the *must give* from the system

***Recognized* position** the position given by the system

A test is considered as correct if the system has given the *Nearest* position as the *Recognized* one.

Experiment	training-base	Test-base
Simulation	*Simulated*	*Simulated*
Reality	*Real*	*Real*
Combined	*Simulated*+*Real*	*Real*

Table 4.1: training-base and test-base for the test performed.

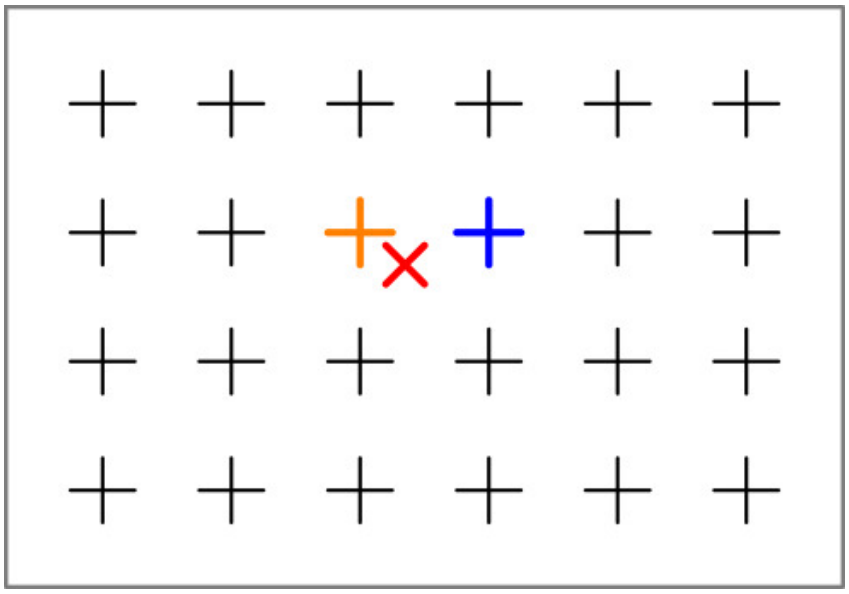


Figure 4.6: Visualization of the 3 proposed positions: the *Ground Truth* in red, the *Nearest* in orange and a potential *Recognized* one in blue, other positions in the training-base in black.

Tables 4.2 and 4.3 resume respectively the percentage of correct matches in the top candidates, according to the previous statement, and some statistics.

The results for each of the three tests are good and promising. But a similarity between the “Reality” and “Combined” tests has been observed. While checking how the latest was resolved, it has been discovered that only the data from the *Real* training-base were used whereas some simulated poses were closest to the test pose. The *Simulated* part did influence the codebook created but not the final pose estimation.

Top candidates considered	Percentage of success
1	76.4%
2	91.4%
3	94.8%
4	95.8%
5	97.0%

Table 4.2: Percentage of correct matches on 500 tests.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000, Number of clusters: 33)

Distance	Mean	Median	Min	Max	RMS	STD
Bet. *GT* and *Recognized*	0.6315	0.4259	0.0142	11.8027	1.2126	1.0352
Bet. *Recognized* and *Nearest*	0.3846	0.0000	0.0000	10.7703	1.1278	1.0601
Bet. *GT* and *Nearest*	0.4112	0.4021	0.0142	1.1023	0.4514	0.1862

Table 4.3: Statistics of the matching errors on 500 tests.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000, Number of clusters: 33)

4.4 Discussion on the application of VLAD to object localization

Once the system previously presented has been working, it was decided to benchmark its efficiency on detecting and localizing an object. Indeed, it's definition is close to the system using a multi-view and multi-level definition presented in Chapter 3. Looking at the definition of inputs and outputs of both methods, Table 4.4 summarize the differences between the solutions.

	Place recognition	Object localization
Model input	Views of the scene	Views of the object
Scene input	View of the scene	View of the scene
output	Pose of the robot	Pose of the object
Method	VLAD and FPFH	Multi-level FPFH and clustering
Method object	Complete scene input	Divided scene input

Table 4.4: Definition of the inputs and outputs of both systems.

On this comparison, keeping away the method, we can see two major differences:

- the way the scene input is used.
- the model which is given.

In the place recognition problem, the scene is used as a single dataset and the model is similar to the scene measured. In the object recognition case, the scene is divided in subparts to be analysed and the model is based on multiple views of the object.

Given this comparison, it is interesting to evaluate whether the place recognition method can be applied to the object recognition and localization problem with LiDAR measurements. This evaluation was performed by directly applying the approach, that leads to

some reflections that are detailed latter. Also, as the systems seems to fail with the combination of real scenes and simulated database, and as the views of the object are simulated, it was decided to use simulated views of the environment. Figure 4.7 present an example of the simulated environment, with the platforms in the middle of the room.

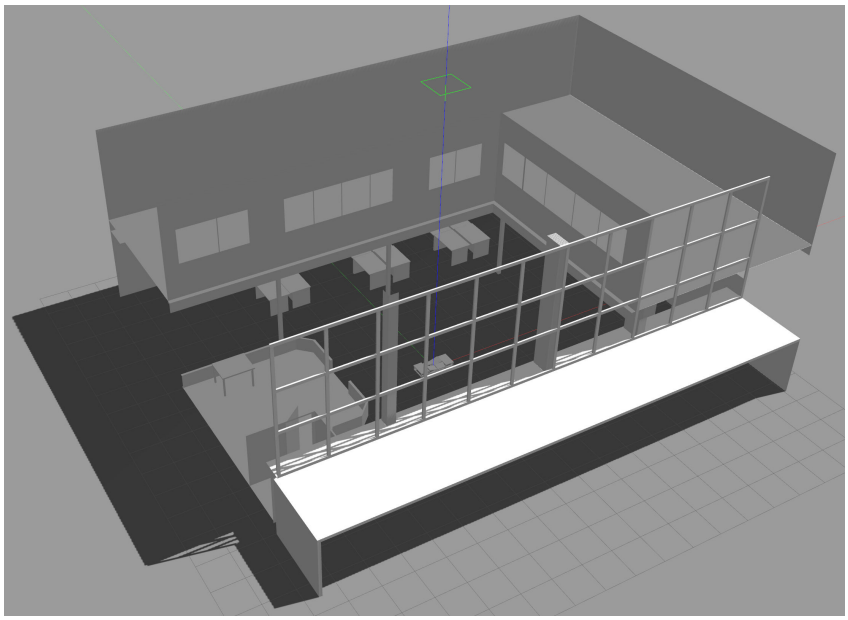


Figure 4.7: Visualization of the simulated Bauzil room with the platforms in its center.

4.4.1 Direct application: a change of model

As one of the main differences in the systems is the model used, it was decided to change the database made of views of the known place by the database of simulated views of the object. The architecture obtained is shown in Figure 4.8.

As for the multi-view and multi-level system presented before, the scene was segmented in *candidate zones* which goes in the VLAD process and were then compared to the database. Another difference of both systems is:

- the place recognition request is the input scene which is a complete LiDAR scan, similar to the database samples
- the object recognition requests are subparts of the input scene to be compared with the database samples

Thus, in the case of the place recognition problem, a single problem is resolved: which view in the database correspond to the input. However, in the case of the object recognition, the best corresponding view in the database has to be identified. But the *candidate zone* representing the object must be determined simultaneously.

As promising as this application looks at firsts, the results were not as expected. Indeed the object was not considered in the 10 best candidates when testing with the real scenes

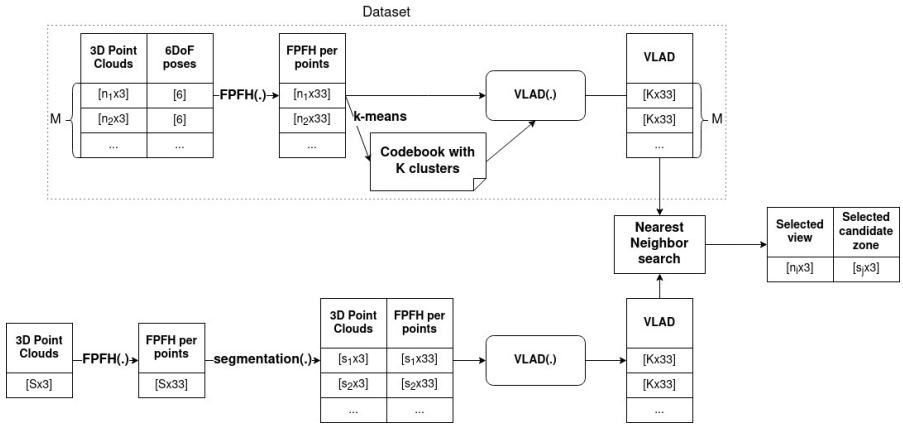
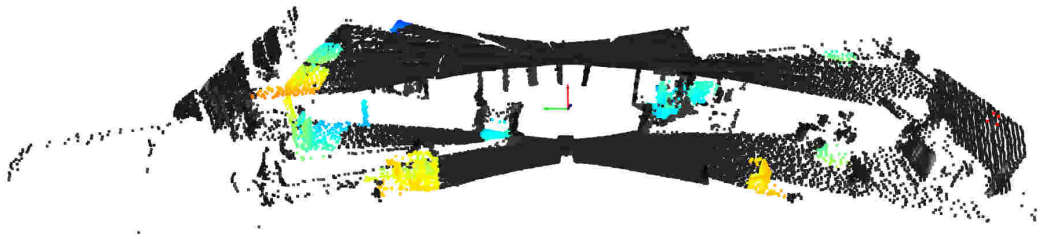
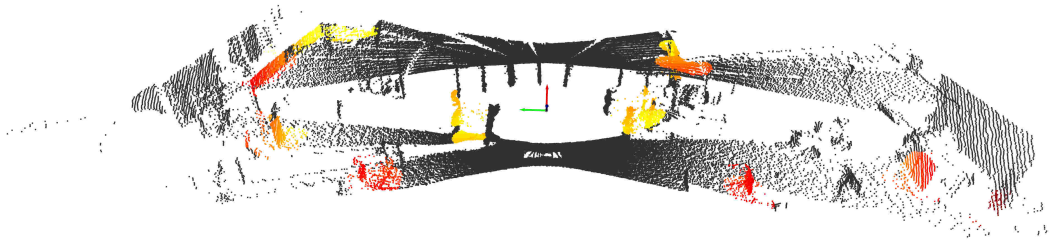


Figure 4.8: Diagram of the application of Guillaume Gobin system to the object localization problem.

presented in Chapter 3. As the system shows difficulties to combine a simulated database and a real test base, as shown previously, the test has been also performed with simulated scenes. Unfortunately, the same results are observed when testing the simulated scenes.



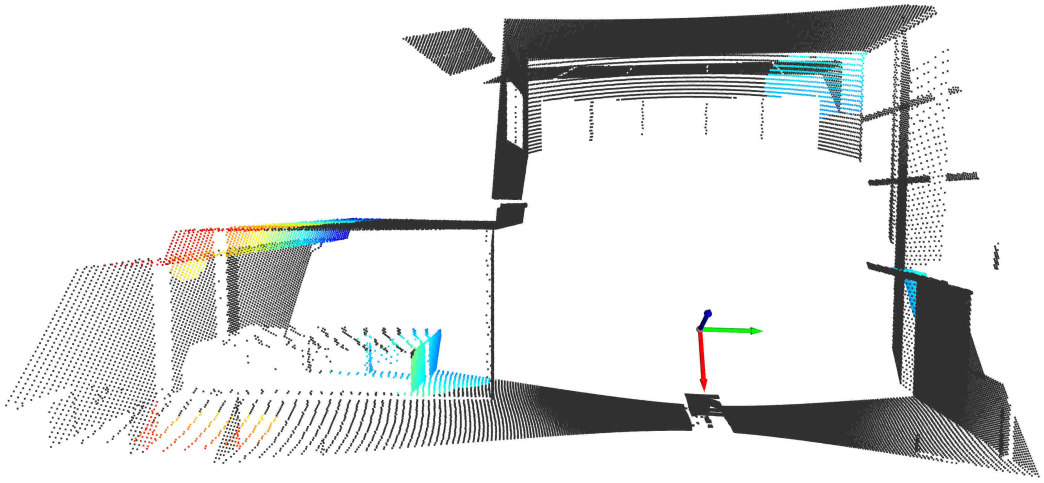
(a) Real scene at PAL Robotics, looking lengthwise



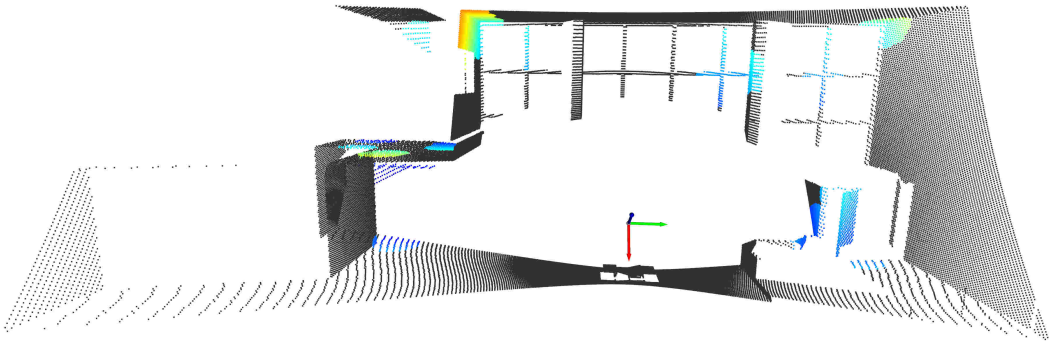
(b) Real scene at PAL Robotics, looking widthwise

Figure 4.9: Visualisation of the object localization in real data, top 50 candidates colored.

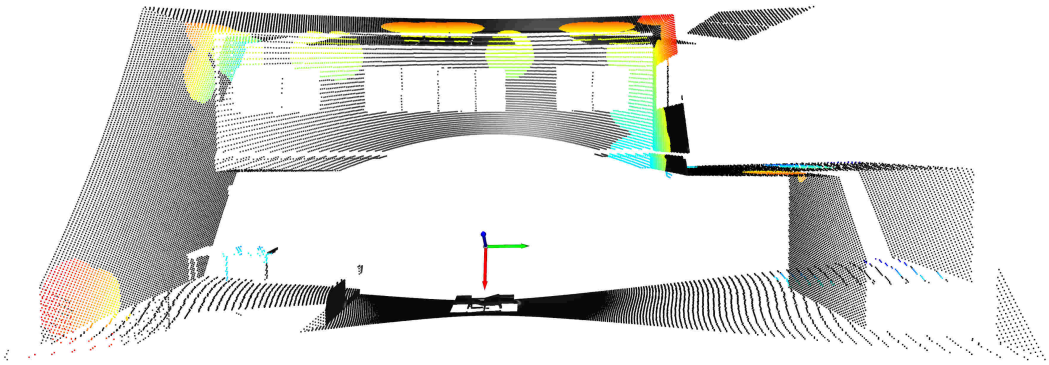
Figures 4.9 and 4.10 show the result of this evaluation. The axis represent the pose of the sensor when the data is taken. These tests were performed with a codebook of 10 words obtained by applying the k-means algorithm to the database features. In the figures, the zone depending of each top candidate is rendered in colors. In the terms of the system, these are the areas in which the object is most likely to be found. However, it can be observed in the figure that the points on the object, in the denser part, are each time rejected.



(a) Simulated scene at LAAS-CNRS, looking lengthwise



(b) Simulated scene at LAAS-CNRS, looking from above



(c) Simulated scene at LAAS-CNRS, looking widhthwise

Figure 4.10: Visualisation of the object localization in simulated data, top 50 candidates colored.

4.4.2 Discussion and perspectives

The results shown by the direct application are not promising but possible improvements are visible. The main aspect of the system is to have a dictionary of features that express the environment.

During the direct application, this dictionary was created using the view of the sensor. Thus, an hypothesis is that using a codebook that represent not only the features present in the object but also the ones that can be encountered in the environment may improve the system. However, a question arises during the discussion: "How can we create this new codebook?". Two solutions are proposed but not studied: create the codebook online from the scene measurement or create a codebook from a database combining the measurement from diverse experiments.

Another hypothesis is on the influence of the environment of the objet. In the simulator, only the object is present for the model generation. Thus, the views are generated without any other feature, such as the floor or a table on which the object is placed. The influence of such detail is not studied here but may be considered.

4.5 Conclusion

In this chapter we presented the work done with G. Gobin to solve the place recognition in a known environment problem using LiDAR data. The proposed method uses state-of-the-art local and global descriptors: FPFH and VLAD. When testing simulated measurements over a simulated database or real measurements over a real database, the results are satisfactory. However a flaw is shown when trying to combine both simulated and real information, either when crossing testbases and databases or when combining both worlds in the testbase and the database.

The method can be improved with the analysis performed with W. Zeng and presented in Chapter 3, Section 3.7.4. Indeed, the "Elbow method" can optimise the clustering step used to create the codebook by defining an optimal number of words based on the dataset. However, the method presented in this chapter was developed before the venue of W. Zeng and the optimisation of the clustering step was empirically looked at.

The method proposed in this chapter has been tested on the object localization problem. This application did not show successful results but has opened up interesting thoughts and perspectives on a way to resolve the object localization problem using the method proposed for place recognition.

CHAPTER 5

Object localization using a Neural Network solution

5.1 Introduction

In the last decade, Neural Network based methods have revolutionized the field of computer vision, object detection and localization.

This chapter starts with a quick recap of what is a neural network and how it is used. Later, a discussion on learning methods applied to object localization is conducted. This discussion is followed by a presentation of the evaluation of two methods applied to the object localization problem using RGB images. After this evaluation, performed with Dorian Baudu during his Master internship, the reader will find a discussion on the perspectives arising from the difficulties encountered.

5.2 Related Works

Object detection, classification and localization are domains of intensive research. These fields are of interest for any system to autonomously interact with it's environment. As example, an autonomous vehicle needs to detect a sufficient part of his environment in order to drive autonomously and safely. It needs to classify the objects around him and determine whether these are mobile obstacles or not. Similarly, other robots, such as complex humanoid-like robots or simpler robotic arms, need to perceive their environment in order to perform a task.

In Chapter 1, a demonstration in which Talos or Tiago are performing deburring tasks on a model of an aircraft pylon is presented. In these demonstrations, the robot needs the core functionality of being able to localize the mast to work on. Moreover, to compensate possible inaccuracies of the robot model or control, it needs to determine the pose of the tool it has to use. Those challenging tasks involve the use of computer vision to perform them.

Localizing an object can be divided into two main phases. Firstly, the environment needs to be analysed in order to determine in which part is the object. This can be resumed as the problem of Object Detection. Secondly, the object pose needs to be refined, often using a known representation, to estimate the pose of the object.

The interests in this field of object detection and localization has lead to an increasing number of publications.

Object detection [Zou 23] make a review of the evolution of object detection based on Deep networks in the last 20 years. The Deep learning-based detectors can be organized in two classes: Two-stages detectors and one-stage detectors. The authors explain the difference between these two categories as ones building a "coarse-to-fine" process while the others perform it "complete in one step".

Two-stages detectors began to appear with R-CNN [Girs 14, Girs 16]. This detector is composed of a first Region proposal system, that extracts object candidates in the image represented by bounding boxes. Then a second CNN took these proposals and extract features for each. These features are used to predict the presence of an object in the proposal and to classify it according to learnt categories. However, R-CNN computes the features for each region candidate, thus resulting in redundant features. This problem as lead to SPPNet [He 15] and later Fast R-CNN [Girs 15]. The principle proposed by SPPNet is to use a "Spatial Pyramid Pooling" layer. This new pooling method allows to compute the features once per image and extract fixed-length representations for each region. Fast R-CNN proposed to make one more step by combining this new layer to an evolution of the training part. Based on the drawbacks observed on the training of R-CNN and SPPNet, that needs to train each sub-part separately, the training phase is reviewed for Fast R-CNN in order to train the detector and bounding box regressor in the same flow. Faster R-CNN [Ren 15] improved again the detection speed by proposing a new Region Proposal Network that shares part of its features with the detection network of Fast R-CNN. However, viewing the features computed in the network as a pyramid going from the image (bottom) to a "global" feature (top), these networks based their object classification on the top layer's features. [Lin 17a] proposed to use the features computed in the deep layers of the CNN to build features at multiple scales by combining these features to the result of the above layer. With this architecture, called Feature Pyramid Network, the authors have shown better and faster results at detecting objects. Those two-stages detectors show high precision in the detection and classification but suffers from their low speed as they go up to 5FPS on a GPU for Faster R-CNN or FPN.

One-stage detectors aim to outperform their peers in terms of speed. YOLO [Redm 16b], standing for You Only Look Once, has been the first of its kind. The authors proposed to apply a single neural network to the full image to perform region proposal, bounding boxes prediction and classification simultaneously. However, while it is faster the two-stage detectors, with its ability to attain 155 FPS, it suffers from a reduced localization accuracy. Subsequent versions of YOLO [Redm 17, Redm 18, Boch 20] took into account their precedent flaws to improve the solution. The detection of objects of diverse sizes is improved in [Liu 16] using a multi-reference and muti-resolution architecture. By performing the detection at different resolutions, the detection accuracy of objects with different sizes was improved. This solution can be compared with the one proposed for FPN as it again uses the result of multiple layers of the network to propose detections. Improvement of the system can also be made by changing the loss used to train the network as shown by RetinaNet [Lin 17b]. In this work, the authors proposed a loss that focus the training of the network on the misclassified examples. Other works have freed themselves from the need to design references of anchor points and boxes for the classification and regression steps. These works evolved to a "Keypoint prediction problem" where they generate heatmap of

the features to detect keypoints. CornerNet [Law 18] sees the problem as detecting the top-left and bottom-right corners of the bounding boxes. Later, ExtremeNet [Zhou 19b] proposed to predict the center point of objects and extreme key points such as the top-most, left-most, bottom-most and right-most points. Thus, the accuracy of the detection can be improved by evaluating the geometry consistency of grouped extreme points to a center point. The same authors proposed to simplify the representation in CenterNet [Zhou 19a] by using only the center point and infer the bounding box from the feature detected.

Object localization In the recent years, the object localization problem has been evaluated during the BOP Challenges [Hoda 20, Sund 23]. The 2019 challenge was led by methods using the depth information to refine the pose estimation. Part of these methods are based on Point Pair Features [Dros 10] such as the 2017 and 2019 winner [Vida 18]. PPF-based methods estimate the pose of an object by matching pairs of 3D points detected in the point cloud with the model. These pairs go through a voting scheme to estimate the final matches and pose.

The BOP Challenge has shown the importance of the training data for object localization. Whereas method based on PPF show to be effectively trained with the only model of the objects, neural networks working on RGB images needs annotated images to train. As taking and annotating real images is a long and fastidious process, the training is commonly performed over generated images. A first solution, often called "render & paste", is to render the objects over random backgrounds, as used in [Kehl 17]. However, [Hoda 19] shows that a NN, in their case Faster R-CNN, performs better when photorealistic images are used instead of the "render & paste" images. Thus, BlenderProc [Denn 20] has been published to synthesize photorealistic images.

Deep Neural Network methods have shown good results in the BOP challenge as the 2020 challenge was led by 5 methods based on deep networks, 3 of which being variants of CosyPose [Labb 20b]. The other two leading method are [Kö 21] and Pix2Pose [Park 19]. [Kö 21] is an hybrid method combining NN and PPF. The method first detects objects and predicts instance masks which are then used to filter the depth information. Each filtered-in mask is then matched to the object following the PPF principle. On the other hand, Pix2Pose predicts 2D-3D correspondences between the image and the model. These predictions are used with the P*n*P-RANSAC algorithm to compute the pose, which is further refined with an ICP algorithm. To refine the final pose of the object, [Labb 20b] also uses a deep neural network from the state-of-the-art, called DeepIM [Li 18]. This DNN is trained to refine iteratively the pose of an object by matching a rendered image of the object to the observed image.

5.3 Understanding the basis of Neural Networks

In the following, a short introduction to Artificial Neural Networks and Convolutional Neural Networks is done.

5.3.1 Artificial Neural Network

Neural networks can be seen as a part of the Machine Learning field. These researches has been inspired by the organic neural networks constituting human's and animal's brain. The principle behind is that complex behaviours can be realised by combining elementary cells, artificial neurons, which perform a simple combination of it's inputs [McCu 43, Rose 58].

These artificial neurons, represented Figure 5.1, are based on a linear combination of the n weighted input x and an activation function f.

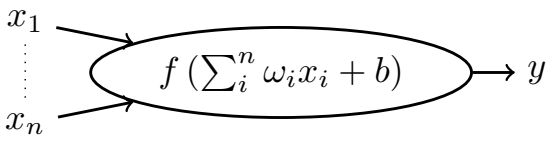


Figure 5.1: Representation of an Artificial neuron.

The output of the artificial neuron is then:

$$y = f\left(\sum_{i=1}^{n} \omega_i x_i + b\right) \tag{5.1}$$

With b being a constant bias of the artificial neuron. Sometimes, this bias is written as the weight ω_0 applied to a constant unitary input.

As simple as these artificial neurons are, more complex tasks and computation are available by combining them as an Artificial Neural Network (ANN) with an entry layer, one or several hidden layers and an output layer, as depicted in Figure 5.2. The middle layers are called hidden due to their inputs and outputs being masked by the activation function f.

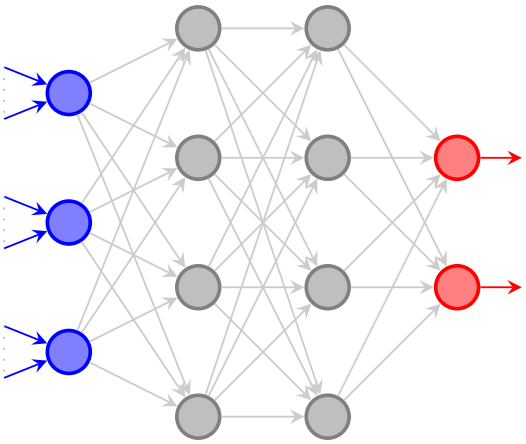


Figure 5.2: Schematization of an Artificial Neural Network with its different layers with it's input layer in blue, hidden layers in gray and output layer in red.

The resulting computation of the ANN is dependent of:

- The number of layers
- The number of artificial neurons per layer
- The weights and activation function of each artificial neurons

The first two items can be regarded as *hyperparameters* of the network, meaning they are constant parameters that define the network. Thus, the learning phase's objective is to adjust the weights and eventually the activation function of the artificial neurons in order to handle better the wanted behaviour. This is done by minimizing the error observed between given samples and the result of the network. For more understanding, it can be expressed as:

$$\min_{W} \sum_{i}^{n} \|Y_i - f(W, I_i)\|^2 \tag{5.2}$$

considering that we are given n training samples consisting of a couple $(in, out) = (I, Y)$, with $f(.)$ the final function of the ANN and W being the weights of the neurons.

5.3.2 Convolutional Neural Network

Convolutional Neural Networks are an evolution of ANN. The principle is to integrate convolutional layers in the layers of a network.

The idea of using a network where each neuron of a layer takes as input a part of the previous layer with a structural representation was used in the "neocognitron"[Fuku 82]. In this work, "neocognitron" is able to differentiate digits. The same idea was then used in LeNet-5 to efficiently recognise digits [LeCu 89] and later ascii characters [LeCu 98].

The convolutional layers apply convolution operation to a set of inputs. This architecture is widely used in the field of computer vision. Indeed, the convolutional layers permits to reduce the data by analysing only regions.

Moreover, the advantage of CNN compared to traditional ANN is the quantity of weights. Indeed, let's take as example a network dealing with images of size 50×50. In this case the input is of size 50×50. For a traditional Artificial Neural Network as a MultiLayer Perceptron (MLP), a neuron of the first layer, fully connected to the input, has $50 \times 50 = 2500$ weights. The quantity of weights for a single neuron can explode the computation time and complexity of the learning phases with the increase of the input size or the size of the network. I.e. for a 100×100 mono-channel image, each neuron needs 10000 weights. In the case of a 3×3 Convolutional Neural Network, a perceptron needs 9 weights to compute the feature. Moreover, weight sharing is often used in convolutional layers. Meaning that each neurons of a single layer uses the same set of weights, called *filter*, reducing again the complexity of the learning phases. The reduction of the number of weights to learn permits to increase the depth of the network and the memory usage.

In computer vision, the input in these networks is a tensor with a shape of $height \times width \times channels$. E.g. when using an RGB image from a 720×480 monocular camera, the input shape is $480 \times 720 \times 3$.

The convolutional layer is tuned by the size of its *kernel*. The kernel represent the quantity of inputs taken and the weight applied to them. Through the convolutional layer, the input is then reduced to a *feature map*.

Figure 5.3 schematizes a Convolutional layer of shape $4 \times 4 \times 1$. This layer uses a 4×4 kernel to transform the image into a feature map. Thus, the receptive field of the neuron is represented in blue and the resulting feature in red.

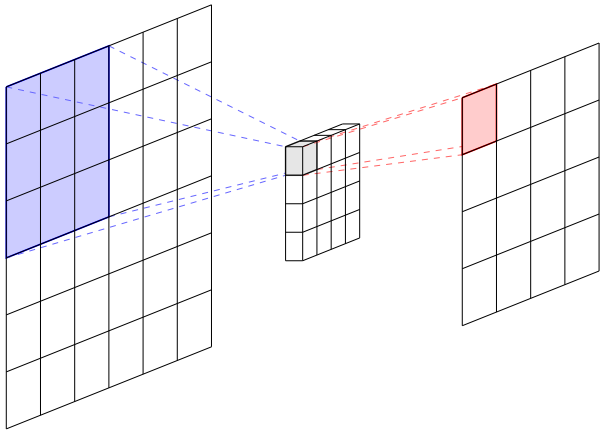


Figure 5.3: Schematization of a $4 \times 4 \times 1$ convolutional layer with a 6×6 input (left) and a 4×4 output feature map (right).

In most CNN, these convolutional layers may be combined with *Pooling layers* or fully connected layers. Fully connected layers are for example used at the end of the network to classify the image. Pooling layers reduces the information by combining the output of a cluster of neurons, e.g. extracting the maximal or average value of the cluster.

Finally, using neural networks in computer vision show a major but laborious step: the learning phase. The learning phase allows to define the weight to use for a specific task. To perform this learning step, the common usage in computer vision is supervised learning, relying on large batch of labelled images with an offline training. Other methods uses self-supervised learning to reduce the quantity of images needed. This kind of learning complete the training with unlabelled data. However, while supervised learning requires the time consuming labelling of the images, the self-supervised learning needs computation capabilities complex to embed on autonomous systems. The computation of self supervised learning may be deported on external computation platforms to reduce the burden on the embed computers.

This section is a short introduction to the bases of Neural Networks to present the complexity of these mechanisms. Due to the large scope of uses of Neural Networks it is not a complete description. For further knowledge, the reader may refer to the specialised literature such as [High 19], [Good 16], [Chol 21] or [Tras 19].

In the present chapter, we study the use of systems using supervised learning to perform an offline training of the networks expected to be embed in the robot. The use of network-based systems generates two phases which can differ considerably in terms of computational complexity. The training phase is computationally expensive and can be handled on

computation clusters. Whereas the execution phase, on the other hand, is intended to be deployed on embedded computers. However, CosyPose, one of the two systems studied in this work, is currently difficult to embed on the robot due to the computing power it requires. The hypothesis of using the network on a decentralised machine using the latest advances in terms of wireless communication has been put forward but will not be studied here.

5.4 Evaluation of CosyPose

CosyPose [Labb 20b] is an approach presented 6D pose estimation of multiple objects using a set of views with unknown viewpoints. The evaluation of this system, as for the evaluation of YOLOv5 has been done with the help of Dorian Baudu for his Master internship.

The objective of this system is to reconstruct the scene observed from multiple viewpoints with unknown poses. This method was been multiple times awarded in the BOP Challenge 2020.
It was awarded as the Overall Best method and the Best Open-Source method.
As the Best Single-Model method because a single model can be trained to detect and estimate the pose of multiple objects.
As the Best RGB-Only method for it's results while using only RGB images, however, it is also able to refine poses with the depth images.
And it was awarded the Best BlenderProc-Trained method for it's use of simulated images for the training.
Moreover, it won a 5th place in the BOP Challenge for it's version with only simulated images in the training phase and without the use of depth refinement.

5.4.1 Global approach

This approach is composed of three main parts, as described in Figure 5.4:

- a single-view single object 6D pose estimation module
- an object candidate matching module
- a global scene refinement module

The first module, detailed later, has for objective to provide object candidates in each views. These candidates are then given to the second module.

The matching module's objective is to remove inconsistent candidates and select candidates that matches across views.

Finally, the third stage uses the matches estimated to refine the 6D poses of the objects and the cameras viewpoints.

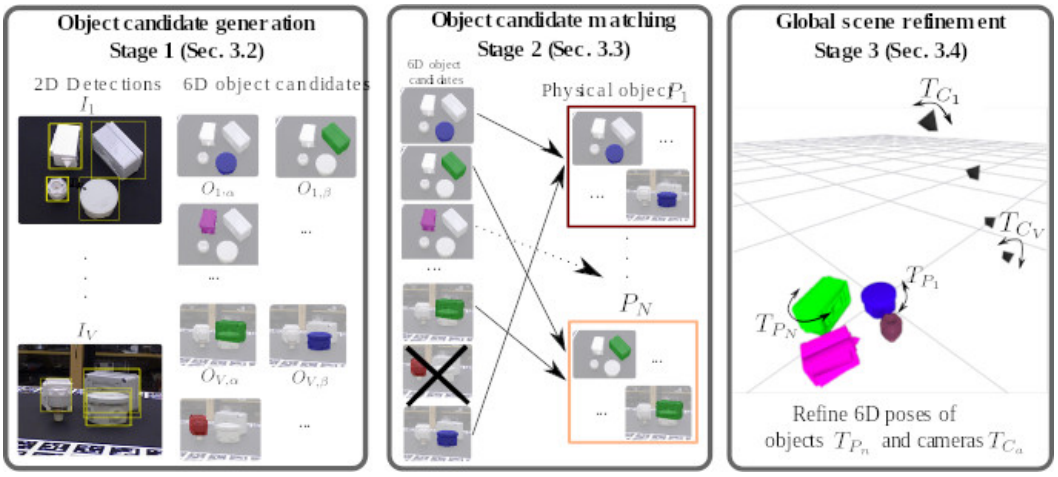


Figure 5.4: Global steps of CosyPose approach, Figure 2 of [Labb 20b].

5.4.2 Single-view 6D pose estimation

In the object localization problem addressed in this chapter, only the single-view single object pose estimation is seen as relevant for an embedded robotic application.

This single-view 6D pose estimation is based on the proposition of DeepIM [Li 18]. However, while the architecture is conserved, multiple changes are done. The network is replaced by a newer based on EfficientNet-B3 [Tan 19]. The quaternion representation is replaced by a continuous rotation parametrization presented in [Zhou 19c], shown to enhance the stability of CNN trainings. The depth and translation prediction are untangled in the loss and the symmetries are handled explicitly with an enumeration of the possibilities. During training, the focal lengths are stated equivalent to the cropped images instead of being fixed to 1 as in DeepIM. The training set is enhanced with images generated from the models with a pre-training on these generated images and a refinement on both the simulated and real images. Lastly, a data augmentation is used on the RGB images in the training phase.

The resulting architecture is shown in Figure 5.5 with its coarse and fine estimators.

5.4.3 Evaluation

In order to evaluate the efficiency of this method to be embedded in our robot, a test was designed. This test plan is in two main parts. First, a model provided by the authors is to be used to determine the efficiency and the potential limits of the system in our scenarii. Then, a training phase is planed over objects of interest, as stairs or the platforms seen in Chapter 3. However, due to difficulties when handling the system that are discussed later, it has been decided to leave this training aside and move on to evaluate a second system: YOLOv5.

Handling with a provided model

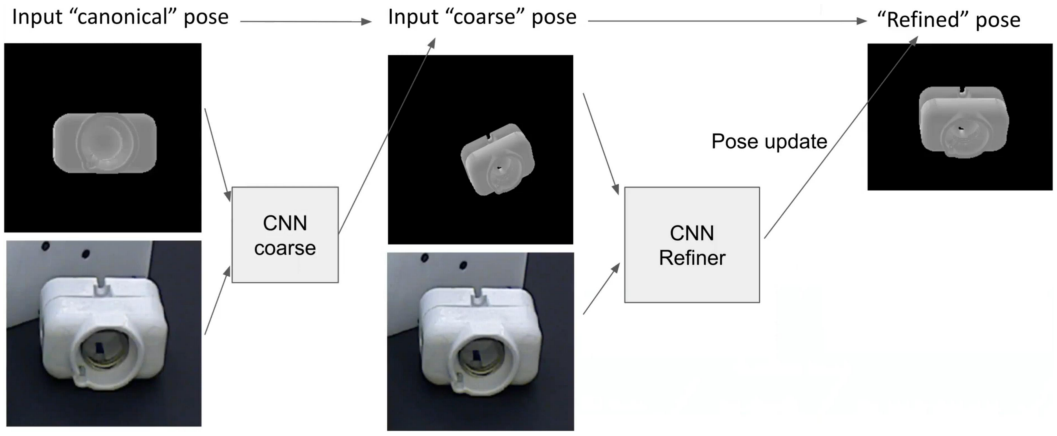


Figure 5.5: Architecture of the single-view single-object 6D pose estimation used in CosyPose, from [Labb 20a].

The first evaluation is the handling of the system with one of the models provided. This evaluation is divided into two parts. Firstly, the tests on a generic dataset, used in the original presentation of the work, are run and compared to the published result. Then the same model is used on data made at our laboratory.

The first test is used to confirm that the system is correctly handled and that there are no problems in the implementation. The T-LESS dataset evaluation is used to perform this handling of the system.

Once these tests on the original T-LESS dataset performed, a dataset is created with the robot and with a handheld camera, using the same objects as in T-LESS and the wooden stairs. Figure 5.6 represent 4 points of view extracted from these measures.

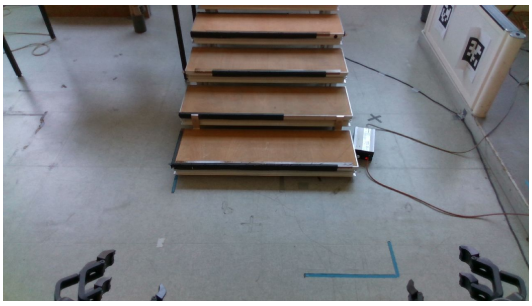
Handling difficulties

While making the first part of the evaluation of the handling of the system, many difficulties occurred. The main difficulties are due to a lack of packaging on the original source code. Indeed, the difference between the platform used to develop by the authors and the computers available introduced problems when trying to deploy CosyPose in more recent computers.

The original source code has been trained on *Jean-Zay*, an HPE SGI 8600 supercomputer from the Institute for Development and Resources in Intensive Scientific Computing (IDRIS). In this work, the objective is to have the training part working on a regional supercomputer called *Olympe* from CALMIP. Moreover, a third supercomputer named *pfcalcul* with less capacities is available in the CNRS-LAAS. This platform was designated as an intermediary support, to verify the system before accessing *Olympe* system. Lastly, a fourth computer, *DeepRAP*, is available for tests. This last platform has been used in a previous work to run CosyPose evaluation. Table 5.1 summarises the capabilities of GPU nodes from the different computers.

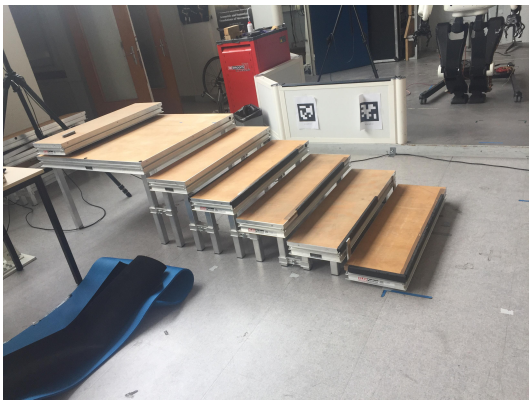
To overcome these difficulties, a Docker container is used. This container has the objective to provide a software environment that is able to run the system reliably on different

(a) “Short” distance with the robot



(b) “Medium” distance with the robot

(c) “Short” distance handheld



(d) “Long” distance handheld

Figure 5.6: Views extracted from the measures used as tests.

computation hardwares. However, the difficulties brought by the change of environment were underestimated. These difficulties are not expanded in this thesis.

A taskforce has been started with the collaboration of the original team of CosyPose and our team to package the system. For the tests presented below, a third computation platform is used. This platform was used for a previous work using CosyPose and was available to try handling the system.

However, the platform was limited in memory and were not able to reproduce the results on the T-LESS dataset as planned. Thus, it was decided to let the reproduction of T-LESS evaluation and continue with the evaluation of single-view object pose estimation with custom images.

Handling results

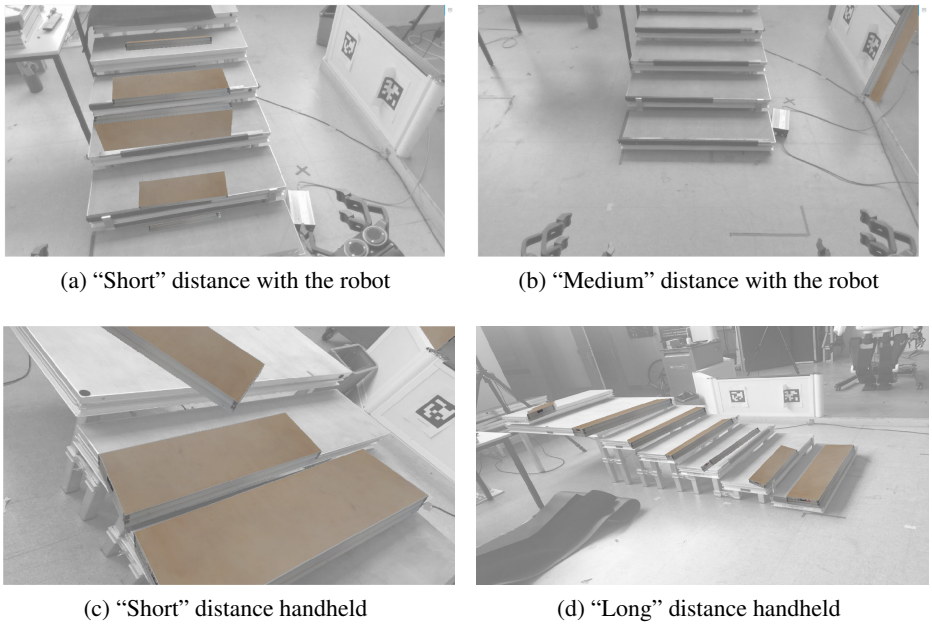
For the tests over the images taken from the robot camera, another trained model is used. This new model is from CosySLAM [Debe 21]. In this work, CosyPose is used to estimate 6D poses of trained objects considered as landmarks in the environment. The network used is trained on T-LESS objects and has been retrained on a set of wooden stairs available at LAAS-CNRS.

This evaluation had encouraging results but also has shown some flaws. Figure 5.7 shows the result provided by CosyPose. The main image has been transformed to a gray-

Name	Nodes	CPUs per node	CPU RAM	GPUs per node	GPU RAM
Jean-Zay (“v100-16g”, “v100-32g”)	612	2 Intel® Cascake Lake 6248 at 2.5GHz	192GB	4 NVIDIA® V100 SXM2	16 – 32GB
Jean-Zay (“gpu_p2”)	31	2 Intel® Cascake Lake 6226 at 2.7GHz	384 – 768GB	8 NVIDIA® V100 SXM2	32GB
Olympe (GPU nodes)	12	2 Intel® Skylake at 2.3GHz	192 – 384GB	4 NVIDIA® V100 SXM2	16 – 32GB
pfcalcul (“dolcino”)	1	Intel® Xeon®Gold 5218R at 2.1GHz	192GB	2 NVIDIA® RTX ™A6000	48GB
DeepRAP	1	Intel® Xeon®Silver 4214R at 2.4GHz	192GB	4 NVIDIA®QUADRO RTX™6000	24GB

Table 5.1: Comparison of the Hardware of the supercomputers.

scale image and the detected object is added with the evaluated transformation. Two observations may be made on these results: the influence of a modified object and the distance of the camera with respect to the object.



(a) “Short” distance with the robot

(b) “Medium” distance with the robot

(c) “Short” distance handheld

(d) “Long” distance handheld

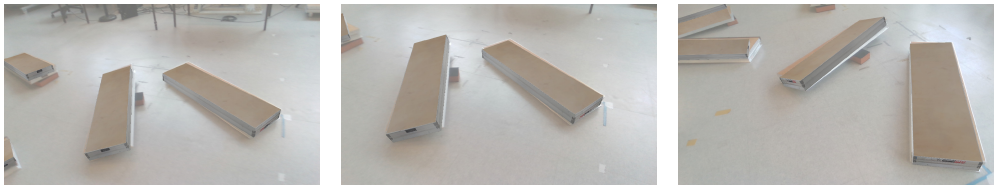
Figure 5.7: Results of the evaluation of the views extracted from the dataset generated.

As shown in the images, a part of the wooden stairs has been modified. A black strip has been glued on the frontal edge. This strip is not present on the initial object representation used to train CosyPose in CosySLAM [Debe 21].

On multiples tests, as shown on Figure 5.7a, 5.7d and 5.7c, the stairs are detected but

the pose estimation does not fit with the observation. However, Figure 5.7b shows that CosyPose may struggle to detect the stairs and instead see other objects as stairs, here the end of a half-wall.

The influence of the modification of the stairs is reinforced by a test reevaluating the data measured in CosySLAM. Figure 5.8 shows views extracted from the measures presented in CosySLAM with the result of the new evaluation. In this tests, we can see that most of the objects are correctly detected and their pose estimated. The results also shows errors in the estimation of object poses. In the top right corner of Figure 5.8c, an object can be observed to be detected but not aligned correctly. These estimation errors are supposed to be due to occlusions hidding most of the object.



(a) (b) (c)

Figure 5.8: Re-evaluation of the measures from CosySLAM.

5.4.4 Discussion and evolution

While the results are encouraging, the difficulties encountered would have implied a lot of technical development. Due to the complexity of integrating the system on other computational environments, it was decided to evaluate another neural network based solution.

Thus, the state-of-the-art was scrutinised for a promising solution that may be efficient when integrated with the robot. Such a solution, while answering to the problematic of estimating the pose of an object, was intended to be computationally frugal. The choice was made for YOLOv5[Redm 16b, Joch 20] and is presented later.

However, CosyPose was not left aside. As mentioned before, a taskforce has been deployed with the original team which developed CosyPose. This taskforce objective is to package CosyPose in a way that it is easily deployed on different computation environment. At the time of writing this manuscript, their work is in the test phase and is promising. They resolved the packaging issue by using a container. And they also addressed the issues that occurred with a used library when running on certain headless environments.

5.5 Evaluation of YOLOv5

YOLO [Redm 16b], for You Only Look Once, is a state-of-the-art network presented in it's first version in 2016. This neural network is used in many applications including object pose estimation for it's fast detection of known objects and it's ability to detect multiple instances of multiple objects in the same iteration. For example, [Kang 20] extends YOLO

to estimate 6D poses of objects. The network was completed the network to estimate 2D projections of the 3D bounding box of objects and then compute the 6D pose. Whereas in [Yang 22], YOLO is used to provide detections. The results of YOLO are then used to obtain cropped images of the detected objects that goes through their other steps including an object center localizer, a translation optimizer, and a rotation estimator and optimizer.

5.5.1 Global approach

The main principle of YOLO is to deal with the objects detection in an image in one shot. Using CNN, this work first divides the image into regions. For each region, a set of confidence bounding boxes are defined. These bounding boxes define the object limits and the probability of having an object. Later, the network classifies each bounding boxes to a learnt class. Thus, as said by one of the initial authors in [Redm 16a], the network does not define that there's an object *A* in this box but that, if there's an object in this box, it is an object *A*.

5.5.2 Evaluation

For a fair evaluation of YOLO compared to CosyPose, it was decided to follow closely the evaluation made on CosyPose. In contrast to CosyPose, YOLO is primarily an object detector. Thus, in this evaluation, only the detection of objects are compared.

Thus, a model was trained on simple images of our wooden stairs. Two phases of learning were done. A "quick" dataset of 100 existing pictures was used first. An "enhanced" dataset of 500 images from different cameras with images showing only the background or ignored objects was made in a second step.

5.5.3 Result with 100 labelled images

This test interest lies in the time needed to do it. Nowadays, developments around YOLO are such that interfaces can be found that takes images, let the user label the images and train the network. Moreover, the installation is easy and compatible with most of the computation platforms available.

A first set of 100 images, extracted from the experiments of [Debe 21] and the previous tests with CosyPose, has been used to train YOLO. This training, as successful as it is shown by the control values, gives bad detection results visible in Figure 5.9. On these results, the detections are, for a majority of them, on the objects. However, for each object, multiple detections appear with a high range of probabilities going from 0.25 up to 0.90.

These problems have been determined to be a lack of training data, as only 100 images were used while at least few thousands are used in traditional trainings. Thus more images were labelled to create a second set.

5.5.4 Result with 500 labelled images

Because of the results obtained with the first set of labelled images, it was decided to put more efforts on these images. The set has been expanded to 500 images using images from

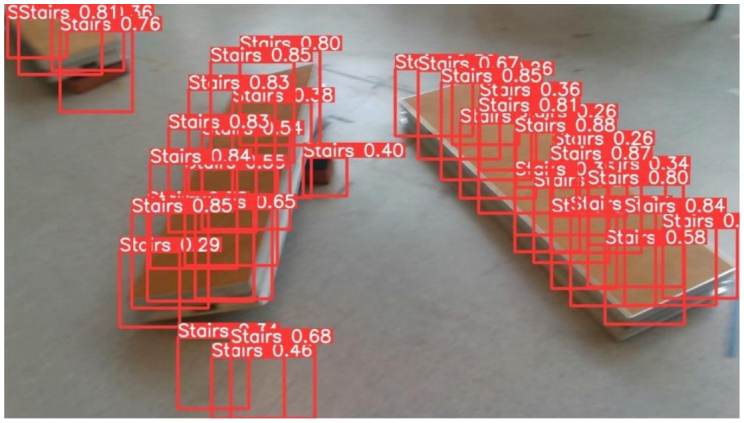
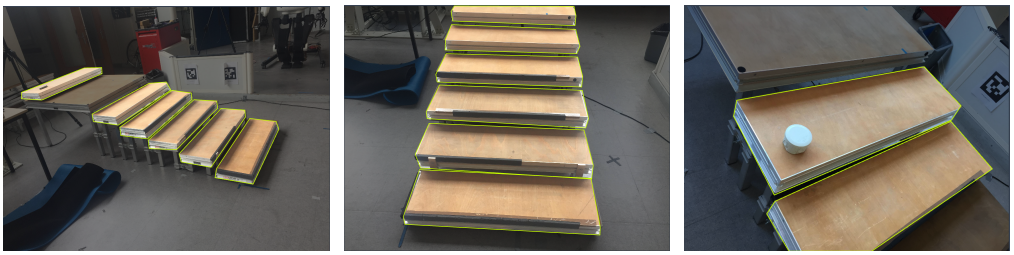
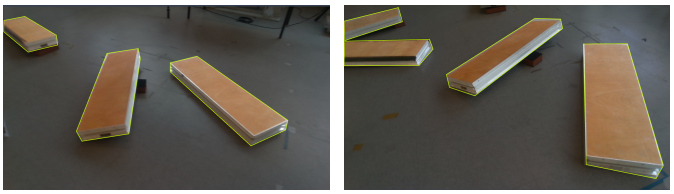


Figure 5.9: Example of the detection of the stairs with the first trained YOLO model.

different cameras. These images contain the wooden stairs with multiple poses and with occlusions. Images without the stairs have also been added to the set. Figure 5.10 shows views from the dataset used to train. In each images, wether from the test set or the training set, the wooden stairs that are at least 60% visible have been framed and labelled. The test images show multiple other objects and views without the stairs, however, only the stairs are for now labelled. The images are extracted from our diverse tests, thus integrating views from different cameras.



(a) (b) (c)



(d) (e)

Figure 5.10: Sample of views from the training dataset, with the object framed.

Compared to the first training, this new step has improved greatly the results. As shown in Figure 5.11, the objects are now detected with probabilities higher than 70%. These results are compared with the results obtained with those using CosyPose. However, as this brick current objective is only detecting an object and does not estimate the object pose, only the detections are compared.

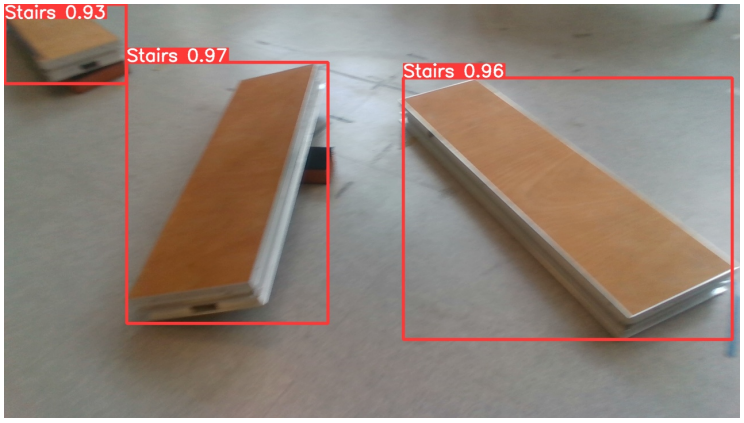


Figure 5.11: Example of the detection of the stairs with the newly trained YOLO model.

Figure 5.12 shows results from both systems. In the middle range case, both systems detect both objects. Differences appears in the others cases.

In case of modified and occluded objects, Five stairs and the last plane are visible. This last plane is longer than a normal stair but can be easily confused due to the texture and shape similarities. As shown on Figure 5.12b, the model of CosyPose used fails to detect all the stairs. It detects a stair in the middle and the last plane as a second stair. On the other hand, YOLO succeed in detecting the 5 stairs. However, YOLO also detects the last plane as a stair and detect a stairs on the floor with low probability.

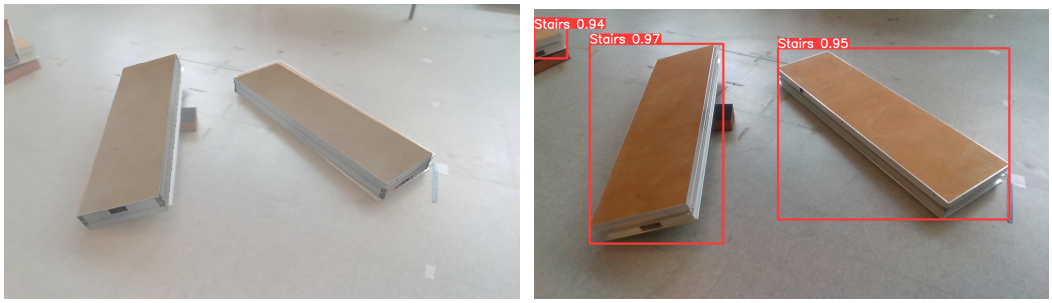
However, when looking at the long distance results, YOLO detects stairs but do not differentiate between the stairs. Indeed, it has provided only two bounding boxes englobing multiple stairs. Of the two bounding boxes, one has a rather high probability of containing stairs whereas the second one, containing the last plane, has a lower probability. On its side, CosyPose at long distance detect successfully the sixth stairs and consider the last plane as a stair too.

5.5.5 Discussion and evolution

The results acquired in these tests are interesting. Indeed, while the dataset is of only hundreds of images with the object, thus not a complete dataset, the results shown are good with the objects detected in most of the cases. This dataset needs to be improved with other objects to evaluate the ability to differentiate the stairs from other objects. Moreover, increasing the datasets with other objects is also interesting to determine multiple objects for robot tasks in one go.

This solution is not yet connected to the object pose estimation. As seen previously, many solutions exists to make this connection and have been tested over existing datasets [Kang 20, Yang 22].

As shown by the tests, the result of YOLO is not dependent of the camera used. Thus, it is conceivable to enclose the images from different cameras to infer or improve the object pose estimation.



(a) Comparison of the results from CosyPose and YOLO in the case of middle range objects.

(b) Comparison of the results from CosyPose and YOLO in the case of messy objects.

(c) Comparison of the results from CosyPose and YOLO in the case of objects at long distance.

Figure 5.12: Comparison of the results from CosyPose (left) and YOLO (right) in the studied cases.

5.6 Conclusion

In this chapter we discussed the use of neural networks to resolve the problem of object localization for a humanoid robot. We first tried to use a state-of-the-art algorithm that has show excellent results in the BOP Challenge 2020. The system was performant but highly difficult to integrate on new computational systems. Thus, it was decided to move to another NN-based solution. As the problem of object localization can be divided in 2 parts, the object detection and the pose estimation, it was decided to first focus on the object detection. Thus, we decided to evaluate the use of YOLO to detect specific low textured object such as wooden stairs. A rapid evaluation of the system shows that such methods are encouraging to localize objects at different distances. Moreover, the tests show that the system is able to perform on different sensors. Thus, it is envisioned to use detections on several sensors in order to better estimate the pose.

In the next chapter, the integration of a new LiDAR-based RGB-D camera and a plane segmentation algorithm is discussed. The objective is to allow the robot to plan its footsteps online.

CHAPTER 6

Plane segmentation integration

6.1 Introduction

For the H2020 MEMory of Motion (MEMMO) project, a plane segmentation algorithm [Fall 19], that was originally developed at MIT for the DARPA Robotics Challenge, has been provided by the Oxford Dynamic Robot Systems Group. This algorithm takes either an elevation map [Fank 14, Fank 18], or a depth measurement, and it extracts plane from it. The extracted planes can be used to perform online contact planning. The University of Edinburgh developed such planner called SL1M[Tonn 20] and applied this pipeline to the ANYmal quadruped robot. In this chapter, the primary steps to perform this same contact planning over data from the environment with the Talos Humanoid robot are presented. For this purpose, an Intel® RealSense™ LiDAR Camera L515 [Inte b] has been added to the robot's waist. The objective for this new camera is to "see" the surfaces in front of the robot. Figure 6.1 shows the robot with the newly installed camera.

Figure 6.1: The Talos robot with the depth camera added to its waist.

The perception part of this problem aims at extracting planes of the environment that can be used for contact planning. Figure 6.2 schematize the proposed pipeline. The work present in this chapter correspond to the part squared in blue. First the calibration of the camera is presented to allows representing the view in the robot space. In a second time, the implementation of a solution from the state-of-the-art is presented with its preliminary results.

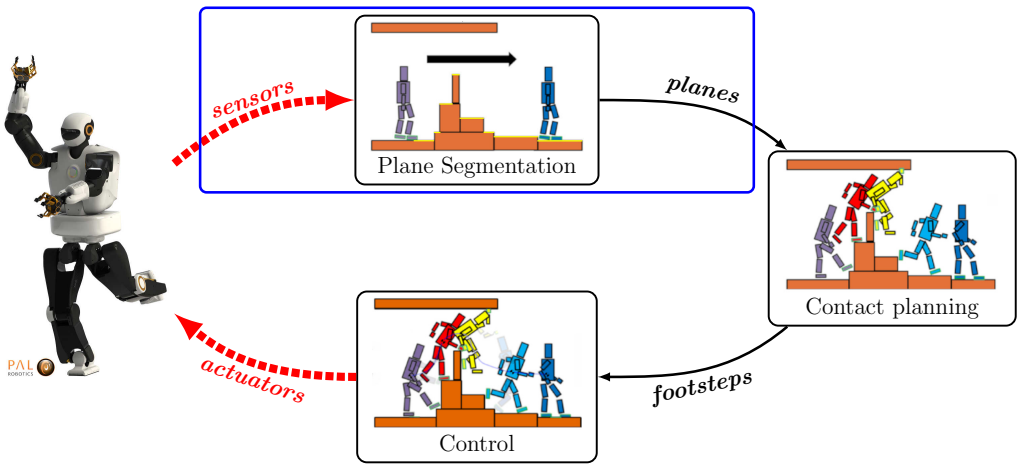


Figure 6.2: From the targeted perception action pipeline, the work presented in this chapter is framed in blue.

6.2 Related works

Contact planning is an important part of the researches conducted by roboticists. In order to behave autonomously and to react to its environment, the legged robot needs to define where it can make contact with the environment. This is necessary for locomotion and can be difficult for a navigation task in a complex environment. Indeed, to move itself in the environment, the robot, either being a quadruped like ANYmal or an humanoid such as Talos, needs to define a sequence of contact for its feet. This sequence of contacts is used to move the robot's body from a starting point to a desired one.

Thus multiple researches have been made to find an optimal solution for this problem.

[Deit 14] presents a method to resolve footstep planning using the Mixed-Integer optimization. It is a mixed-integer problem because the discrete part is to decide wether to make contact with a surface or not. The continuous part is the location of the contact. Their work has shown qualitative results on planning contact phases on uneven terrain while ensuring the reachability of each step and the avoidance of obstacles. However, according to their evaluations the computation time of this method goes up to a minute for a plan involving 10 to 20 footsteps. Thus, [Tonn 20] intends to reduce this computation time by relaxing the contact planning problem using the L1-norm minimization.

The common problematic to the different categories of contact planner is on which data do they plan. Indeed, all these methods needs to know how is the environment and where can the contact be done in order to determine how to perform an asked motion.

[Oß 11] evaluated two state-of-the-art approaches to segment plane in order to climb stairs. The first method is based on the scan-line grouping algorithm and is presented in [Gutm 08]. The second is based on the two point sampling method [Kida 04] to extract planes. With the plane extracted, the authors reconstruct a model of the staircase in order to define how to climb it. The principle to extract planes over a 2.5D height map has also been used in [Fall 15b] to plan footsteps. The data from a stereo camera is filtered

and aggregated in an height map. Then convex regions, large enough to be used as a foothold, are estimated and the motion is planned over with a mixed-integer quadratic optimization. [Kark 16] uses a 2.5D height map to extract planes and edges in the same time. Latter, [Grif 19] uses an equivalent planar representation to plan footsteps but allows the planification of partial footholds to increase the possibilities.

However, in real environment, contact may be done with curved surfaces. Thus, [Kano 18] studied the combination of the modeling of curved patches on local surfaces with a flat contact analysis. The authors show that with this combination applied to visual range data allows to plan contact in order to go through the surface. Latter, [Kano 19] extends this approach by generating a set of parametrized curved patchs. These patches, about the size of the robot's are matched to the perceived environment to define potential contact poins.

6.3 Calibration of the L515 camera in the robot model

In order to use the Intel RealSense L515 LiDAR camera on the robot, its pose on the robot needs to be determined. There are two main ways to calibrate the camera. A first way is using a mire and a measurement while moving the robot to perform a kinematic calibration. For our purpose, a simplest but less precise calibration is performed using an external system in order to determine the usefulness of such a camera in this location. This external calibration is the fastest and let us avoid the current flexibility problem on the robot.

Thus, to perform this calibration, the Motion Capture (Mocap) available in our experimental room is used with the help of markers fixed on specific poses on the robot and the camera. Then, using their know poses in the base frame of the robot p_k^B, the camera base frame p_j^C and their respective poses in the Mocap measurements $p_{\{k,j\}}^W$, the transformations between the robot base frame and the Mocap ${}^W T_B$ and between the camera base frame and the Mocap ${}^W T_C$ can be determined. Lastly, the relation between the camera base frame and the robot base frame ${}^B T_C$ is easily determined with Eq. 6.1.

$$ {}^B T_C = {}^W T_B^{-1} \, {}^W T_C \tag{6.1} $$

6.3.1 Markers positions

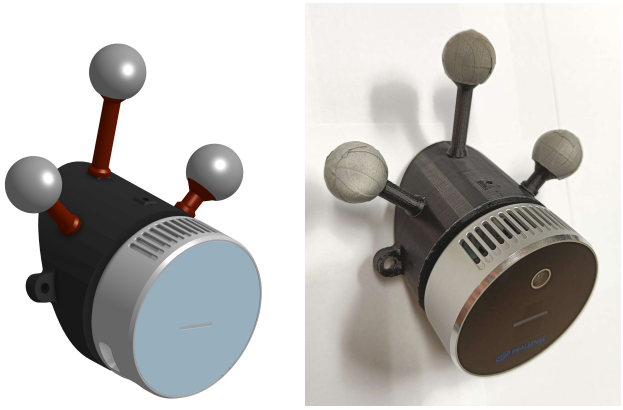
6.3.1.1 Fixed markers on the camera

A camera stand has been designed firstly by PAL-Robotics. Then, it was modified to add fixed markers with easy-to-determine poses. The final stand is shown in Figure 6.3 and has been 3D printed to be mounted on the robot.

The position of the mocap markers in the camera frame are summarized in Table 6.1.

6.3.1.2 Fixed markers on the robot

To get the best result from this calibration, markers on the robot needs to be placed with precise and known poses. As this calibration is performed with the robot standing still,



(a) CAO visualization (b) Printed version

Figure 6.3: Stand used to mount the Intel RealSense L515 on the Talos waist.

id	x (mm)	y (mm)	z (mm)
0	−35	0	69.099
1	−19.990	−45.576	37.404
2	−10	31.055	37.841

Table 6.1: Camera's markers positions in the camera frame.

it was decided to use structural screw threads used to mount the covers of the robot, here the battery cover fasteners. The same sticks as for the camera stand were used, visible in Figure 6.3a and the poses of the threads centers were provided by the robot maker. The position of the markers in the robot base frame are summarized in Table 6.2.

id	x (mm)	y (mm)	z (mm)
0	−39	−228.099	−12
1	−73	−223.099	−123
2	−6	−203.099	−120

Table 6.2: Robot's markers positions in the robot frame.

6.3.2 Results

Once the camera mounted and the Mocap markers in place, a measurement of the markers was taken with the Mocap system. This measurement is given in Table 6.3.

Thus the transformation between the frames and a virtual Mocap frame M can be determined as in Eq. 6.2, with F being either the robot base frame B or the camera frame C, ${}^{M}\tilde{x}_i$ the measured of a marker and ${}^{F}x_i$ its position in its respective frame, given by the forward kinematics.

$$ {}^{M}T_F = argmin \sum_i \left\| {}^{M}\tilde{x}_i - {}^{M}T_F * {}^{F}x_i \right\|, \; {}^{M}T_F \in SE(3) \tag{6.2} $$

Object	id	x (mm)	y (mm)	z (mm)
Camera	0	547.5	−207.2	1003.3
	1	538.1	−254.5	970.4
	2	544.3	−177.2	962.9
Robot	0	361.3	−440.7	1010.6
	1	338.7	−436.7	895.8
	2	417.1	−415.7	906.3

Table 6.3: Measured position of the markers in the Mocap virtual frame.

These transformations is optimised using a $SE(3)$ object expressed as a vector $[x, y, z, qx, qy, qz, qw]$, with its translation part $[x, y, z]$ and its rotation part as a quaternion $q = [qx, qy, qz, qw]$. Due to the use of a quaternion, a second cost must be added to ensure its definition. The final cost is presented in Eq. 6.3

$$ {}^{M}T_{F} = argmin \sum_{i} \left(\left\| {}^{M}\tilde{x}_i - {}^{M}T_{F} * {}^{F}x_i \right\|^2 \right) + \left\| \|q\|^2 - 1 \right\|^2, \; {}^{M}T_{F} \in SE(3) \tag{6.3} $$

The resulting transformations are presented on Table 6.4 while Table 6.5 provides the errors computed given by Eq. 6.4.

	translation			rotation (quaternion)			
	x	y	z	x	y	z	w
${}^{M}T_{B}$	0.408	−0.2137	1.026	−0.0037	−0.0677	−0.0184	0.9975
${}^{M}T_{C}$	0.5434	−0.2092	0.9149	−0.0123	0.3609	0.0064	0.9325

Table 6.4: Transformation expressing the camera base frame C and the robot base frame B into the mocap virtual frame M. (rounded at $1e^{-4}$)

Object	Axis	Mean error	Standard deviation
Camera	x	$1.56e^{-4}$	$1.68e^{-4}$
	y	$-2.5e^{-4}$	$1.94e^{-4}$
	z	$9.9e^{-5}$	$4.8e^{-5}$
Robot	x	$-3.13e^{-4}$	$7.28e^{-4}$
	y	$-2.126e^{-3}$	$2.363e^{-3}$
	z	$2.439e^{-3}$	$3.074e^{-3}$

Table 6.5: Resume of the mean error and standard deviation for each axis on each set of markers. (rounded at $1e^{-6}$)

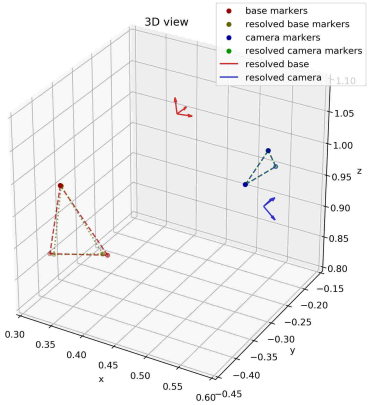
$$ e = {}^{M}\tilde{x}_i - {}^{M}T_{F} \, {}^{F}x_i \tag{6.4} $$

Then Eq. 6.1 can be applied to determine the wanted transformation of the camera base frame in the robot base frame. The resulting transformation of the camera base frame in

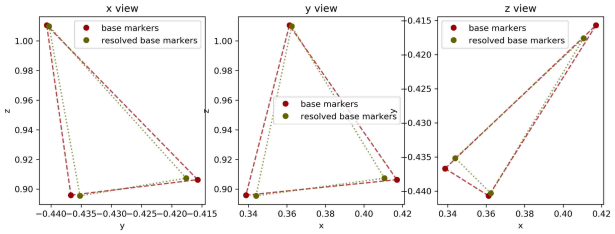
the robot base frame, rounded at $1e^{-6}$, is

$$^{B}T_{C} = [0.118935, 0.009963, -0.128373, -0.014975, 0.422896, 0.025669, 0.905691]$$

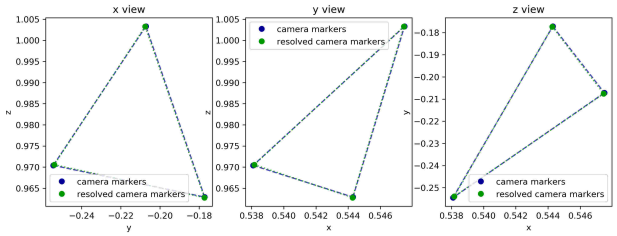
Figure 6.4 shows the visualization of the resolved calibration.



(a) 3D visualization



(b) Planar views of the base markers



(c) Planar views of the camera markers

Figure 6.4: Visualization of the result of the L515 calibration in the Mocap virtual frame. (a) shows a 3D visualization of the calibration, (b) and (c) show the planar views.

Lastly, another RGB-D camera, an Intel RealSense D435i, is placed in the head. Its FOV intersect with the L515's when the robot is standing still. The calibration quality has been roughly estimated by "visually" inspecting the fusion of the two camera point clouds. The calibration results in a good intersection of the measured data of both cameras as shown in Figure 6.5. The middle of the image shows that the two cameras' images are overlapping on the first stair observed. The "smooth" and wide perception is from the Intel RealSense D435i and the "rougher" view that shows the 3 first steps and the front of the steps is from the Intel RealSense L515. There are several ways to improve the fusion of the cameras information. Chapter 5 with the use of the multi-view system of CosyPose is one possibility. A model of both cameras could also be used to improve the precision of the reconstruction. This result has not been more evaluated yet.

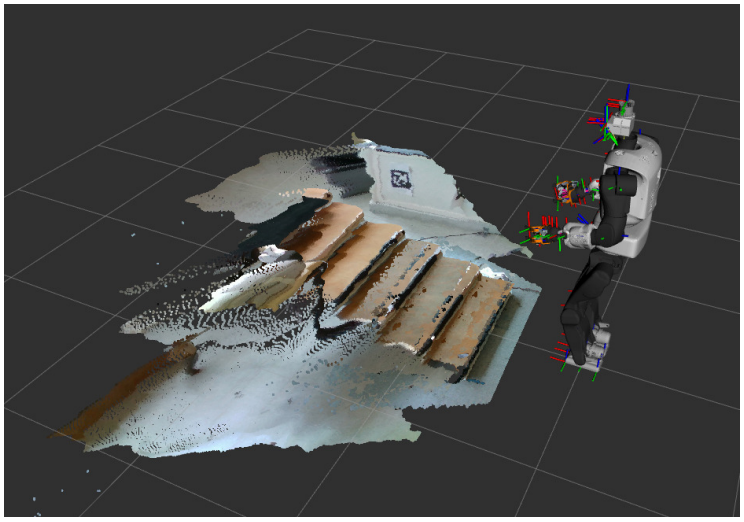


Figure 6.5: Combination of the head's D435i and the waist L515 cameras with the calibration.

6.4 Plane segmentation using an Intel RealSense L515 camera

6.4.1 Setup

The sensor used is an Intel®RealSense™LiDAR Camera L515. It is an RGB-D camera using a flash LiDAR to measure the depth of the scene. Thanks to its solid state LiDAR, it is able to provide a depth image at $30Hz$ with a resolution up to 1024×768. It has a FOV of $70° \times 55°$ for an ideal range of $25cm$ to $9m$.

6.4.2 First implementation on Talos

In order to be implemented, this couple of system needs a base estimator in order to combine successive views of the cameras. On the robot, a base estimator is present which origin is set as the sole frame of a feet at initialization of the estimator. Thus its reference is the ground plane at the start. In this first experiment, this base estimator is used. In further experiments, this estimator could be joined to the LiDAR SLAM presented in Chapter 2 in order to express all the useful information in the world frame of the map.

Parameters are needed to implement the elevation mapping system. For the first experiment, the basic version of these configurations is used, with for example a sensor considered as "perfect", and a mapping of a $2.5 * 2.5m$ around the base.

On the side of the plane segmentation algorithm, some light modifications were applied to adapt the interface to our robot and its sensor.

Once these configurations and modifications are ready, a measurement of the robot walking slowly to go in front of stairs was done. Figure 6.6 shows the robot on its final position in front of the stairs and the result of the combined system with the grid map and the plane segmented. The results shows that the stairs in front of the robots are detected. However, because the parametrization is the simplest, a high level of noise is present in the

elevation map. Due to the noise in the map, the planes are unstable and small planes are detected. This test shows the interest of having a camera looking at the floor. However, it also shows that further optimisation of the different parameters from the map and the plane extractions is needed, to avoid the propagation and amplification of noises, perturbing the final plane detection.

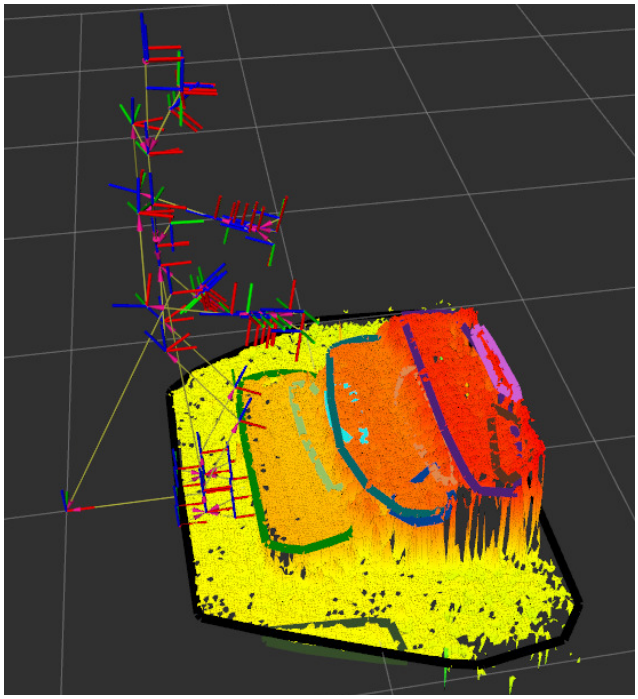


Figure 6.6: Visualization of the final position in front of the stairs and the results given by the elevation mapping and the plane segmentation.

6.5 Discussion and Perspectives

In this chapter, the preliminary work to link the contact planning to real data extracted from the environment was presented. This system objective is to reproduce experiments made at the University of Edinburgh on their ANYmal robot with the Talos robot. To reach this objective, the addition of a new sensor on the robot was needed to improve the perception of the floor. The choice was made to use an Intel®RealSense™LiDAR Camera L515, a LiDAR-based RGB-D camera.

Thus, this camera has been added to the robot and a kinematic calibration has been performed to express the measures in the robot space. This calibration has been made using an external system. Results from such a calibration are fast but can be inaccurate. This calibration can be improved by using a dynamic calibration with known motions of the robot.

Then, the sensor has been linked to the plane segmentation algorithm maintained by Oxford Robotics Institute. This interface was made with minimal parametrization. The resulting planes obtained are encouraging but show noise and unnecessary overlaps. These flaws can be resolved by tuning the interface with a better understandings of the parameters and their use.

These observations leave us with a last perspective to this part. Indeed, time has been spent by an engineer on the link between the acquired planes and the contact planer. However, this link has not yet been integrated on the robot for online contact planning on real time measures of the environment. The team is working on linking each part together to enable the online footstep planning on our robot.

CHAPTER 7

Conclusion

7.1 Contributions

Throughout this thesis, practical solutions were investigated to address the problems of environmental perception for humanoid robots in aeronautic factories. Chapter 1 presented the industrial partnership between Airbus and the Gepetto team and the specificities of working in aeronautic factories. The humanoid robot TALOS used along this thesis was also presented. These solutions are mainly using LiDAR data. There are several reasons for that, the first one being the wide and low texture environment due to the industrial context. The objective to this thesis is to propose new solutions or use existing ones to improve the autonomy of humanoid robots in an industrial environment for locomotion and manipulation.

Chapter 2 presents the integration of a state-of-the-art LiDAR-based localization system initialized with a Visual Inertial Odometry. The system enables to localize the robot in an indoor environment based on LiDAR data with an Iterative Closest Point algorithm. As it needs to have a lower-level base estimation to initialise each new run of the ICP, we proposed to use a VIO available with a sensor, the T265 camera, that is embedded in the robot. This system is able to localize over a pre-built map as well as to build its map "on the fly". We tested the system accuracy during a locomotion task and compared its results to a Motion Capture system. In the experiment, the robot had to reach multiple pre-defined positions in the map, as it may need when working autonomously. The localization system shows an average precision of $2\,cm$ on each axis in translation and of $0.024\,\text{rad}$ around the z axis with respect to the motion capture system. This experiment shows that the ICP algorithm initialised by a off-the-shelves VIO system is able to localize accurately the robot in its environment. This integration and tests led to a conference paper in ICAR 2021 [Lasg 21].

Chapter 3 discussed the use of geometric descriptors to localize objects from LiDAR data. This work is motivated by the industrial context of this work, presented in Chapter 1. This context requires the robot to be able to locate itself in large aeronautic factories and to be able to find objects of interest for the tasks it has to perform. These objects can either be tools at short distance or its workbench or work area at longer distance. Thus, we discussed the possibility to use LiDAR measurements and geometric descriptors to localize these objects. This localization problem is highly dependent of the density of informations available and of occlusions. We proposed to use a state-of-the-art descriptor, the Fast Point Feature Histogram, in a multi-level analysis with a multi-view model. We proposed to use this descriptor in a two steps algorithm. First, the generation and use of a multi-view and multi-level model is designed taking into account the sensors specificities. Secondly, the

use of compact data representation methods is discussed to manage the memory complexity of the model. For the model generation, considering the 3D model to be known, a simulation is performed to extract multiple sensor views. The simulation uses the sensor model as a reference, allowing to reduce the difficulty of the combination of simulated and real data. These views are used to generate a database for a feature matching algorithm. Then, the scene is divided in segments at a large scale. The FPFH descriptor is computed for each segment and compared to the database to be filtered. The best correspondences are then studied at a smaller scale. This process iterates from the division to the filtering steps until we reach a defined scale. The evaluation of this methodology shows the ability to reduce a large FOV LiDAR measurement to candidates zones with a challenging object. However, this methodology needs to be tested on different types of objects in order to have a better understanding of its capabilities and limitations. The next step is to use these candidates zones to estimate the pose of the object.

The Chapter 4 is motivated by the observation of a limitation of the system presented in Chapter 1. Indeed, localization systems over a known environment may need an initial guess of the robot pose and orientation. This is a well known problem in the community, called "Place Recognition" or "Kidnapped Robot problem". We proposed to resolve this problem using the same descriptor as used in Chapter 3 and LiDAR data. Our proposed solution combines the local descriptor FPFH to a global VLAD descriptor. This system builds a database composed of known poses of the robot in the environment and the global descriptor measured. Then for a new LiDAR measurement, the local and global descriptors are computed and compared to the database in a Nearest-Neighborhood fashion to provide an estimate of the pose of the robot. This system has been evaluated with data acquired in simulation and in reality. However, a limitation of the system has been shown when combining both kind of data. Indeed, while the system is satisfactory when using a database and a testbase of the same kind, being Simulated or Real, when the bases are crossed, the system fails to localize the test measurements. Moreover, when combining measurements from the Simulation and the Reality in the database and in the testbase, for a measurement of the testbase, the systems uses only the measurement of the same kind in the database.

Chapter 5 describes a tentative to solve the object localization problem with promising Deep Neural Network solutions. Indeed, the computer vision community tries to resolve this problem in multiple ways. The use of the RGB data by a neural network is discussed. First, we tried to use CosyPose [Labb 20b], the network which won the BOP Challenge 2020 [Hoda 20]. However, while it shows great results, its implementation was difficult to manipulate on different computation systems. Thus, we decided to switch to another strategy. As the object localization problem can be resolved first as an object detection problem then as a pose estimation, we decided to start a progressive step. After a review of the existing systems, we decided to start by evaluating the integration of YOLO [Redm 16b]. This integration as shown encouraging results with a low level of difficulty in the training and handling.

Last but not least, Chapter 6 presents an evolution of the capabilities of the robot. In order to allow our robot to be more autonomous in its locomotion, we worked at making it able to plan it's footsteps. Thus we added a LiDAR-based RGB-D camera on the waist of the robot and calibrate it. The calibration was firstly performed with an external Mo-

tion Capture system. Then, we integrated a system provided by our partner in the project Memmo to extract plans from the depth measurements. This is done by integrating successive depth measurement into an Elevation Map. The plane segmentation system takes the 2.5D map and provides convex hulls of the detected surfaces. These convex hulls are to be used by a contact planner in order to define possible footholds for the robot.

7.2 Perspectives

This thesis discussed perception solutions to realize reactive object manipulations and locomotion in aeronautic factories with a humanoid robot. We concentrated on the use of LiDAR information in the context of this PhD. In this thesis we presented solution to localize the robot, objects of interest and surfaces for foothold planning. However, these solutions can be improved.

The solution discussed in the Chapter 2 is currently closed source. Recent development are now offering an open-source equivalent in [ETH 21]. As the team's motivation is to have open-source systems for reproducibility, it is interesting to evaluate these solutions and compare them to our deployed solution.

The place recognition solution discussed in Chapter 4 shows encouraging results. This work was led before the last work presented in the previous chapter. It could be made more effective by applying the assessments and developments presented. Moreover, the solution still suffers from the Simulation to Reality gap. The literature shows that this problem is an active area of research, specifically in deep learning methods. Indeed, these methods needs a lot of data to train but real data took a long time to acquire and label. Thus, simulation is seen as a way to acquire large batch of labelled data easily. This is called self-supervised learning.

The object localization based on LiDAR data and FPFH descriptors presented in Chapter 3 is still in its early stage. First of all, the current work need to be linked with a final pose estimation system in order to complete the localization. Moreover, we have tested it on a challenging object we possess at LAAS-CNRS that represent multiple stepping platforms. We want to test it over multiple objects and scene in order to better understand its limitations and improve its capabilities. Furthermore, as explained before, acquiring and labelling data is a fastidious task. We suppose that efficient systems that are not based on neural network can be used to autonomously label data. Thus it is imaginable to use this system to generate labels on acquired data for training purposes.

This work also presented preliminary works performed on the use of deep-learnt solutions. The first solution discussed, CosyPose, presented difficulties to implement on new computation platforms. However, since we made its evaluation, a taskforce has worked on the packaging of CosyPose to make it easier to deploy. Their work is available at [Gauc 22, Agim 22] and is still a work in progress. Moreover, the authors of CosyPose worked on a new network, called MegaPose [Labb 22]. Their work is able to perform pose estimation for novel untrained objects when provided with their meshes. On the other side, YOLO as shown accurate and fast detections. Thus, we consider coupling the detections of

several cameras embed by the robot to allow an accurate estimation of object poses. This coupling can be envisaged either with YOLO or by reverting the multi-view optimisation of CosyPose, as the pose between the cameras are known on the robot.

Lastly, except for the localization system that has been embedded and tested on the humanoid robot Talos, the other solutions discussed have yet to be deployed. In the case of the plane segmentation presented in Chapter 6, the deployment and integration with the footstep planning is in progress. Moreover, the results shown in this work are obtained with few tuning steps of the elevation map algorithm. We hope to be able to efficiently localize available surfaces for contact planning with the L515 camera. Furthermore, the elevation map algorithm is able to fuse multiple sources. We can suppose that fusing the L515 "close" data to the Ouster LiDAR "global" data can improve the accuracy of the environment precision as well as extend the depth of the detection.

Bibliography

[Agil 23] Agility Robotics. *ProMat 2023 Recap*. https://www.youtube.com/watch?v=RVQ68Iagnb0, 2023. (Cited in page 14.)

[Agim 22] Agimus-Project. *HappyPose*. GitHub, https://github.com/agimus-project/happypose, 2022. (Cited in page 115.)

[Airb 19] Airbus. *Airbus inaugurates new A320 structure assembly line in Hamburg*. https://www.airbus.com/en/newsroom/press-releases/2019-10-airbus-inaugurates-new-a320-structure-assembly-line-in-hamburg, 2019. (Cited in pages 1 and 2.)

[Aldo 12] A. Aldoma, F. Tombari, R. B. Rusu and M. Vincze. *OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. In A. Pinz, T. Pock, H. Bischof and F. Leberl, editors, Pattern Recognition, pages 113–122, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (Cited in page 32.)

[Aran 16] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla and J. Sivic. *NetVLAD: CNN Architecture for Weakly Supervised Place Recognition*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 5297–5307, 2016. (Cited in page 73.)

[Barr 21] T. Barros, R. Pereira, L. Garrote, C. Premebida and U. J. Nunes. *Place recognition survey: An update on deep learning approaches*. arXiv preprint arXiv:2106.10458, 2021. (Cited in page 71.)

[Belo 02] S. Belongie, J. Malik and J. Puzicha. *Shape matching and object recognition using shape contexts*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 24, no. 4, pages 509–522, 2002. (Cited in pages 32 and 72.)

[Besl 92] P. J. Besl and N. D. McKay. *A method for registration of 3-D shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 14, no. 2, pages 239–256, 1992. (Cited in page 21.)

[Blac 19] J. Blackman. *"Yes, the driller has to drill, but it also has to compute", and other Airbus rules for Indutry 4.0*. https://www.rcrwireless.com/20190124/fundamentals/the-driller-has-to-drill-but, 2019. (Cited in page 1.)

[Boch 20] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao. *Yolov4: Optimal speed and accuracy of object detection*. arXiv preprint arXiv:2004.10934, 2020. (Cited in page 86.)

[Chol 21] F. Chollet. Deep learning with python. Manning Publications, 2021. (Cited in page 90.)

[Chum 05] O. Chum and J. Matas. *Matching with PROSAC - progressive sample consensus*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 220–226 vol. 1, 2005. (Cited in page 72.)

[Dant 22] E. L. Dantec, M. Naveau, N. Mansard, P. Fernbach, N. Villa, G. Saurel, O. Stasse and M. Taïx. *Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot*. In IEEE-RAS Int. Conf. on Humanoid Robotics (Humanoids), Ginowan, Japan, November 2022. (Cited in page 11.)

[DARP] DARPA. *Defense Advanced Research Projects Agency Robotics Challenge (DRC)*. `https://www.darpa.mil/program/darpa-robotics-challenge`. (Cited in page 5.)

[Davi 07] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse. *MonoSLAM: Real-Time Single Camera SLAM*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 29, no. 6, pages 1052–1067, 2007. (Cited in page 19.)

[Debe 21] C. Debeunne, M. Fourmy, Y. Labbé, P.-A. Léziart, G. Saurel, J. Solà and N. Mansard. CosySlam: investigating object-level SLAM for detecting locomotion surfaces. working paper or preprint, September 2021. (Cited in pages 10, 94, 95, and 97.)

[Deit 14] R. Deits and R. Tedrake. *Footstep planning on uneven terrain with mixed-integer convex optimization*. In IEEE-RAS Int. Conf. on Humanoid Robotics (ICHR), pages 279–286. IEEE, 2014. (Cited in page 104.)

[Dell 99] F. Dellaert, D. Fox, W. Burgard and S. Thrun. *Monte Carlo Localization for Mobile Robots*. In Int. Conf. on Robotics and Automation (ICRA), volume 2, pages 1322–1328, 1999. (Cited in page 20.)

[Dell 17] F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Foundations and Trends in Robotics, vol. 6, no. 1-2, 2017. (Cited in page 20.)

[Denn 20] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam and A. Lodhi. *Blenderproc: Reducing the reality gap with photorealistic rendering*. In Robotics: Science and Systems (RSS), 2020. (Cited in page 87.)

[Doui 12] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. De Deuge, S. Hugosson, M. Hallström and T. Bailey. *Scan segments matching for pairwise 3D alignment*. In Int. Conf. on Robotics and Automation (ICRA), pages 3033–3040, May 2012. ISSN: 1050-4729. (Cited in page 72.)

[Dros 10] B. Drost, M. Ulrich, N. Navab and S. Ilic. *Model globally, match locally: Efficient and robust 3D object recognition*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 998–1005, 2010. (Cited in page 87.)

[Dub 17] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart and C. Cadena. *SegMatch: Segment based loop-closure for 3D point clouds*. Int. Conf. on Robotics and Automation (ICRA), pages 5266–5272, May 2017. arXiv: 1609.07720. (Cited in page 72.)

[Dub 18] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart and C. Cadena. *SegMap: 3D Segment Mapping using Data-Driven Descriptors*. Robotics: Science and Systems (RSS), June 2018. arXiv: 1804.09557. (Cited in page 72.)

[E.Jo 99] A. E.Johnson and M. Hebert. *Using spin images for efficient object recognition in cluttered 3D scenes*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 21, no. 5, pages 433–449, 1999. (Cited in page 32.)

[Elba 17] G. Elbaz, T. Avraham and A. Fischer. *3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 2472–2481. IEEE, July 2017. (Cited in page 72.)

[ETH 21] Robotic Systems Lab Legged Robotics at ETH Zürich. *icp_localization - Localization using ICP in a known map*. GitHub, `https://github.com/leggedrobotics/icp_localization`, 2021. (Cited in page 115.)

[Fall 15a] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. Pérez-D'Arpino, R. Deits, M. Dicicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller. *An Architecture for Online Affordance-based Perception and Whole-body Planning*. Jour. of Field Robotics, vol. 32, no. 2, pages 229–254, 2015. (Cited in pages 13 and 20.)

[Fall 15b] M. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald and R. Tedrake. *Continuous humanoid locomotion over uneven terrain using stereo fusion*. In IEEE-RAS Int. Conf. on Humanoid Robotics (ICHR), pages 881–888, 2015. (Cited in page 104.)

[Fall 19] M. Fallon and M. Antone. *Plane Seg – Robustly and Efficiently Extracting Contact Regions from Depth Data*. `https://github.com/ori-drs/plane_seg`, 2019. (Cited in page 103.)

[Fank 14] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter and R. Siegwart. *Robot-Centric Elevation Mapping with Uncertainty Estimates*. In Int. Conf. on Climbing and Walking Robots (CLAWAR), 2014. (Cited in page 103.)

[Fank 18] P. Fankhauser, M. Bloesch and M. Hutter. *Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization*. IEEE Robotics and Automation Letters (RAL), vol. 3, no. 4, pages 3019–3026, 2018. (Cited in page 103.)

[Fern 20] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier and M. Taïx. *C-CROC: Continuous and Convex Resolution of Centroidal dynamic trajectories for legged robots*

in multi-contact scenarios. IEEE Transactions on Robotics (T-RO), pages 1–16, 2020. (Cited in page 11.)

[Fisc 81] M. A. Fischler and R. 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, June 1981. (Cited in page 13.)

[Flay 17] R. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue and O. Stasse. *Experimental evaluation of simple estimators for humanoid robots*. In IEEE-RAS Int. Conf. on Humanoid Robotics (Humanoids), pages 889–895, 2017. (Cited in page 9.)

[Four 22] M. Fourmy. *State estimation and localization of legged robots : a tightly-coupled approach based on a-posteriori maximization*. Theses, INSA de Toulouse, March 2022. (Cited in page 10.)

[From 04] A. Frome, D. Huber, R. Kolluri, T. Bülow and J. Malik. *Recognizing objects in range data using regional point descriptors*. In European Conference on Computer Vision (ECCV), pages 224–237. Springer, 2004. (Cited in page 32.)

[Fuku 82] K. Fukushima and S. Miyake. *Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition*. In Competition and cooperation in neural nets, pages 267–285. Springer, 1982. (Cited in page 89.)

[Galv 12] D. Galvez-López and J. D. Tardos. *Bags of Binary Words for Fast Place Recognition in Image Sequences*. IEEE Transactions on Robotics (T-RO), vol. 28, no. 5, pages 1188–1197, October 2012. (Cited in page 73.)

[Garr 14] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas and M. J. Marín-Jiménez. *Automatic generation and detection of highly reliable fiducial markers under occlusion*. Pattern Recognition, vol. 47, no. 6, pages 2280–2292, 2014. (Cited in page 3.)

[Gate 19] D. Gates. *Boeing abandons its failed fuselage robots on the 777X, handing the job back to machinists*. `https://www.seattletimes.com/business/boeing-aerospace/boeing-abandons-its-failed-fuselage-robots-on-the-777x-handing-the-job-back-to-machinists`, 2019. (Cited in pages 1 and 2.)

[Gauc 22] D. Gauchard and E. Maître. *gputainer*. GitHub, `https://github.com/d-a-v/gputainer`, 2022. (Cited in page 115.)

[Girs 14] R. Girshick, J. Donahue, T. Darrell and J. Malik. *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), June 2014. (Cited in page 86.)

[Girs 15] R. Girshick. *Fast R-CNN*. In IEEE Int. Conf. on Computer Vision (ICCV), December 2015. (Cited in page 86.)

[Girs 16] R. Girshick, J. Donahue, T. Darrell and J. Malik. *Region-Based Convolutional Networks for Accurate Object Detection and Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 38, no. 1, pages 142–158, 2016. (Cited in page 86.)

[Gobi 21] G. Gobin. Localisation d'un robot humanoïde TALOS par information géométrique et visuelle combinée à l'apprentissage. Master's thesis, Toulouse 3 Paul Sabatier, August 2021. (Cited in page 74.)

[Good 16] I. Goodfellow, Y. Bengio and A. Courville. Deep learning. MIT Press, 2016. `http://www.deeplearningbook.org`. (Cited in page 90.)

[Grif 19] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee and J. Pratt. *Footstep Planning for Autonomous Walking Over Rough Terrain*. In 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), pages 9–16, 2019. (Cited in page 105.)

[Guo 18] J. Guo, P. V. K. Borges, C. Park and A. Gawel. *Local Descriptor for Robust Place Recognition using LiDAR Intensity*. arXiv:1811.12646 [cs], November 2018. arXiv: 1811.12646. (Cited in page 73.)

[Gutm 08] J.-S. Gutmann, M. Fukuchi and M. Fujita. *3D perception and environment map generation for humanoid robot navigation*. Int. Jour. of Robotics Research (IJRR), vol. 27, no. 10, pages 1117–1134, 2008. (Cited in page 104.)

[Hack 16] T. Hackel, J. Wegner and K. Schindler. *Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density*. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Annals), vol. III-3, pages 177–184, 06 2016. (Cited in page 49.)

[He 15] K. He, X. Zhang, S. Ren and J. Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 37, no. 9, pages 1904–1916, 2015. (Cited in page 86.)

[He 16] K. He, X. Zhang, S. Ren and J. Sun. *Deep residual learning for image recognition*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. (Cited in page 72.)

[High 19] C. F. Higham and D. J. Higham. *Deep Learning: An Introduction for Applied Mathematicians*. SIAM Review, vol. 61, no. 4, pages 860–891, 2019. (Cited in page 90.)

[Hoda 19] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha and B. Guenter. *Photorealistic image synthesis for object instance detection*. In IEEE Int. Conf. on Image Processing and Robotics (ICIP), pages 66–70. IEEE, 2019. (Cited in page 87.)

[Hoda 20] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother and J. Matas. *BOP challenge 2020 on 6D object localization*. In European Conference on Computer Vision (ECCV), pages 577–594. Springer, 2020. (Cited in pages 87 and 114.)

[Horn 10] A. Hornung, K. M. Wurm and M. Bennewitz. *Humanoid robot localization in complex indoor environments*. In IEEE Int. Conf. on Intelligent Robots and Systems (IROS), pages 1690–1695, 2010. (Cited in page 20.)

[Hurs 19] J. Hurst. *Building Robots That Can Go Where We Go*. IEEE Spectrum: Technology, Engineering, and Science News, `https://spectrum.ieee.org/building-robots-that-can-go-where-we-go`, 2019. (Cited in page 14.)

[Inte a] Intel. *Intel® RealSense™ Depth Camera D435i*. `https://www.intelrealsense.com/depth-camera-d435i/`. (Cited in page 15.)

[Inte b] Intel. *Intel® RealSense™ LiDAR Camera L515*. `https://www.intelrealsense.com/lidar-camera-l515/`. (Cited in pages 72 and 103.)

[Inte c] Intel. *Intel® RealSense™ Tracking Camera T265*. `https://www.intelrealsense.com/tracking-camera-t265/`. (Cited in pages 10 and 15.)

[Joch 20] G. Jocher. *YOLOv5 by Ultralytics*, May 2020. (Cited in page 96.)

[John 17] M. Johnson, B. Shrewsbury, S. Bertrand, D. Calvert, T. Wu, D. Duran, D. Stephen, N. Mertins, J. Carff, W. Rifenburgh, J. Smith, C. Schmidt-Wetekam, D. Faconti, A. Graber-Tilton, N. Eyssette, T. Meier, I. Kalkov, T. Craig, N. Payton, S. McCrory, G. Wiedebach, B. Layton, P. D. Neuhaus and J. E. Pratt. *Team IHMC's Lessons Learned from the DARPA Robotics Challenge: Finding Data in the Rubble*. Jour. of Field Robotics, vol. 34, no. 2, pages 241–261, 2017. (Cited in page 13.)

[Kang 20] J. Kang, W. Liu, W. Tu and L. Yang. *YOLO-6D+: Single Shot 6D Pose Estimation Using Privileged Silhouette Information*. In IEEE Int. Conf. on Image Processing and Robotics (ICIP), pages 1–6, 2020. (Cited in pages 96 and 99.)

[Kano 18] D. Kanoulas, A. Stumpf, V. S. Raghavan, C. Zhou, A. Toumpa, O. Von Stryk, D. G. Caldwell and N. G. Tsagarakis. *Footstep Planning in Rough Terrain for Bipedal Robots Using Curved Contact Patches*. In Int. Conf. on Robotics and Automation (ICRA), pages 4662–4669, 2018. (Cited in page 105.)

[Kano 19] D. Kanoulas, N. G. Tsagarakis and M. Vona. *Curved patch mapping and tracking for irregular terrain modeling: Application to bipedal robot foot placement*. Robotics and Autonomous Systems, vol. 119, pages 13–30, 2019. (Cited in page 105.)

[Kark 16] P. Karkowski and M. Bennewitz. *Real-time footstep planning using a geometric approach*. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1782–1787, 2016. (Cited in page 105.)

[Kehl 17] W. Kehl, F. Manhardt, F. Tombari, S. Ilic and N. Navab. *Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again*. In IEEE Int. Conf. on Computer Vision (ICCV), pages 1521–1529, 2017. (Cited in page 87.)

[Kida 04] Y. Kida, S. Kagami, T. Nakata, M. Kouchi and H. Mizoguchi. *Human finding and body property estimation by using floor segmentation and 3D labelling*. In IEEE Int. Conf. on Systems, Man and Cybernetics (SMC), volume 3, pages 2924–2929 vol.3, 2004. (Cited in page 104.)

[Kim 18] G. Kim and A. Kim. *Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map*. In IEEE Int. Conf. on Intelligent Robots and Systems (IROS), pages 4802–4809, October 2018. (Cited in page 72.)

[Kö 21] R. König and B. Drost. *A hybrid approach for 6DoF pose estimation*. In European Conference on Computer Vision (ECCV), pages 700–706. Springer, 2021. (Cited in page 87.)

[Komo 21a] J. Komorowski. *MinkLoc3D: Point Cloud Based Large-Scale Place Recognition*. In IEEE Winter Conf. on Applications of Computer Vision (WACV), pages 1789–1798, 2021. (Cited in pages 72 and 73.)

[Komo 21b] J. Komorowski, M. Wysoczańska and T. Trzcinski. *MinkLoc++: Lidar and Monocular Image Fusion for Place Recognition*. In Int. Joint Conf. on Neural Networks (IJCNN), 2021. (Cited in page 72.)

[Labb 20a] Y. Labbé. *CosyPose ECCV 2020 - 10 min presentation*. `https://www.youtube.com/watch?v=MNH_Ez7bcP0`, 2020. (Cited in page 93.)

[Labb 20b] Y. Labbé, J. Carpentier, M. Aubry and J. Sivic. *CosyPose: Consistent multi-view multi-object 6D pose estimation*. In European Conference on Computer Vision (ECCV), 2020. (Cited in pages 87, 91, 92, and 114.)

[Labb 22] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox and J. Sivic. *MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare*. In Conference on Robot Learning (CoRL), 2022. (Cited in page 115.)

[Lami 22] F. Lamiraux. *Accurate Pointing of holes in a party by localization using vision.* `https://peertube.laas.fr/w/hYitAMe96DP9iPuBjSYCc8`, 2022. (Cited in pages 7 and 9.)

[Lasg 21] T. Lasguignes, I. Maroger, M. Fallon, M. Ramezani, L. Marchionni, O. Stasse, N. Mansard and B. Watier. *ICP Localization and Walking Experiments on a TALOS Humanoid Robot*. In Int. Conf. on Advanced Robotics (ICAR), pages 800–805, 2021. (Cited in pages 3, 17, 19, and 113.)

[Law 18] H. Law and J. Deng. *Cornernet: Detecting objects as paired keypoints*. In European Conference on Computer Vision (ECCV), pages 734–750, 2018. (Cited in page 86.)

[LeCu 89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. *Backpropagation applied to handwritten zip code recognition*. Neural computation, vol. 1, no. 4, pages 541–551, 1989. (Cited in page 89.)

[LeCu 98] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. *Gradient-based learning applied to document recognition*. Proc. of the IEEE, vol. 86, no. 11, pages 2278–2324, 1998. (Cited in page 89.)

[Li 18] Y. Li, G. Wang, X. Ji, Y. Xiang and D. Fox. *DeepIM: Deep iterative matching for 6d pose estimation*. In European Conference on Computer Vision (ECCV), pages 683–698, 2018. (Cited in pages 87 and 92.)

[Li 21] W. Li, H. Cheng and X. Zhang. *Efficient 3d object recognition from cluttered point cloud*. Sensors, vol. 21, no. 17, page 5850, 2021. (Cited in page 32.)

[Lim 17] J. Lim, I. Lee, I. Shim, H. Jung, H. M. Joe, H. Bae, O. Sim, J. Oh, T. Jung, S. Shin, K. Joo, M. Kim, K. Lee, Y. Bok, D.-G. Choi, B. Cho, S. Kim, J. Heo, I. Kim, J. Lee, I. S. Kwon and J.-H. Oh. *Robot System of DRC-HUBO+ and Control Strategy of Team KAIST in DARPA Robotics Challenge Finals*. Jour. of Field Robotics, vol. 34, no. 4, pages 802–829, 2017. (Cited in pages 12 and 13.)

[Lin 17a] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie. *Feature Pyramid Networks for Object Detection*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), July 2017. (Cited in page 86.)

[Lin 17b] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár. *Focal loss for dense object detection*. In IEEE Int. Conf. on Computer Vision (ICCV), pages 2980–2988, 2017. (Cited in page 86.)

[Liu 16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg. *Ssd: Single shot multibox detector*. In European Conference on Computer Vision (ECCV), pages 21–37. Springer, 2016. (Cited in page 86.)

[Lowe 99] D.G. Lowe. *Object recognition from local scale-invariant features*. In IEEE Int. Conf. on Computer Vision (ICCV), volume 2, pages 1150–1157 vol.2, 1999. (Cited in page 72.)

[Matu 15] D. Maturana and S. Scherer. *VoxNet: A 3D Convolutional Neural Network for real-time object recognition*. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 922–928, 2015. (Cited in page 33.)

[McCu 43] W. S. McCulloch and W. Pitts. *A logical calculus of the ideas immanent in nervous activity*. The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pages 115–133, 1943. (Cited in page 88.)

[Mira 20] J. Mirabel, A. Nicolin, F. Lamiraux, O. Stasse and S. Boria. Integrating path planning and visual servoing in manipulation tasks. working paper or preprint, February 2020. (Cited in page 3.)

[Mira 21] J. Mirabel, F. Lamiraux, T. L. Ha, A. Nicolin, O. Stasse and S. Boria. *Performing manufacturing tasks with a mobile manipulator: from motion planning to sensor based motion control*. In Int. Conf. on Automation Science and Engineering (CASE), pages 159–164, 2021. (Cited in page 3.)

[Mur- 15] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós. *ORB-SLAM: a Versatile and Accurate Monocular SLAM System*. IEEE Transactions on Robotics (T-RO), vol. 31, no. 5, pages 1147–1163, 2015. (Cited in page 10.)

[Newc 11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges and A. Fitzgibbon. *KinectFusion: Real-time dense surface mapping and tracking*. In 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pages 127–136, 2011. (Cited in page 14.)

[Nico 20] A. Nicolin, J. Mirabel, S. Boria, O. Stasse and F. Lamiraux. *Agimus: a new framework for mapping manipulation motion plans to sequences of hierarchical task-based controllers*. In IEEE/SICE Int. Symp. on System Integration (SII), 2020. (Cited in page 3.)

[Nico 22] A. Nicolin. *Planning of visual servoing tasks for robotics*. Theses, INSA de Toulouse, February 2022. (Cited in page 3.)

[Nobi 17] S. Nobili, R. Scona, M. Caravagna and M. Fallon. *Overlap-based ICP tuning for robust localization of a humanoid robot*. In Int. Conf. on Robotics and Automation (ICRA), pages 4721–4728, 2017. (Cited in pages 20, 21, 22, and 72.)

[Osad 02] R. Osada, T. Funkhouser, B. Chazelle and D. Dobkin. *Shape Distributions*. ACM Transactions on Graphics (ToG), vol. 21, no. 4, page 807–832, oct 2002. (Cited in page 32.)

[Oß 11] S. Oßwald, J.-S. Gutmann, A. Hornung and M. Bennewitz. *From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids*. In IEEE-RAS Int. Conf. on Humanoid Robotics (ICHR), pages 93–98, 2011. (Cited in page 104.)

[Oß 12] S. Oßwald, A. Hornung and M. Bennewitz. *Improved proposals for highly accurate localization using range and vision data*. In IEEE Int. Conf. on Intelligent Robots and Systems (IROS), pages 229–254, 2012. (Cited in page 20.)

[Oust] Ouster. *High-resolution OS1 LiDAR sensor: robotics, trucking, mapping*. `https://ouster.com/products/scanning-lidar/os1-sensor/`. (Cited in page 16.)

[Park 19] K. Park, T. Patten and M. Vincze. *Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation*. In IEEE Int. Conf. on Computer Vision (ICCV), pages 7668–7677, 2019. (Cited in page 87.)

[Pome 13] F. Pomerleau, F. Colas, R. Siegwart and S. Magnenat. *Comparing ICP Variants on Real-World Data Sets*. Autonomous Robots, no. 3, page 133–148, February 2013. (Cited in pages 20 and 21.)

[Qi 17a] C. R. Qi, H. Su, K. Mo and L. J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 652–660, 2017. (Cited in pages 33 and 73.)

[Qi 17b] C. R. Qi, L. Yi, H. Su and L. J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. arXiv preprint arXiv:1706.02413, 2017. (Cited in page 33.)

[Rame 20] M. Ramezani, G. Tinchev, E. Iuganov and M. Fallon. *Online LiDAR-SLAM for Legged Robots with Robust Registration and Deep-Learned Loop Closure*. In Int. Conf. on Robotics and Automation (ICRA), pages 4158–4164, 2020. (Cited in pages 21, 22, and 72.)

[Ramu 21] N. Ramuzat, G. Buondonno, S. Boria and O. Stasse. *Comparison of position and torque whole-body control schemes on the humanoid robot Talos*. In Int. Conf. on Advanced Robotics (ICAR), pages 785–792. IEEE, 2021. (Cited in page 3.)

[Ramu 22a] N. Ramuzat. *Robotics Force/Torque Control for Manufacturing Operations*. Theses, INSA de Toulouse, February 2022. (Cited in page 3.)

[Ramu 22b] N. Ramuzat, S. Boria and O. Stasse. *Passive Inverse Dynamics Control using a Global Energy Tank for Torque-Controlled Humanoid Robots in Multi-Contact*. IEEE Robotics and Automation Letters (RAL), vol. 7, no. 2, pages 2787–2794, 2022. (Cited in page 3.)

[Ramu 22c] N. Ramuzat, O. Stasse and S. Boria. *Benchmarking Whole Body controllers on the TALOS Humanoid Robot*. Frontiers in Robotics and AI, vol. 9, page 826491, March 2022. (Cited in page 3.)

[Redm 16a] J. Redmon. *You Only Look Once: Unified, Real-Time Object Detection*. `https://www.youtube.com/watch?v=NM6lrxy0bxs`, 2016. (Cited in page 97.)

[Redm 16b] J. Redmon, S. Divvala, R. Girshick and A. Farhadi. *You only look once: Unified, real-time object detection*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2016. (Cited in pages 86, 96, and 114.)

[Redm 17] J. Redmon and A. Farhadi. *YOLO9000: better, faster, stronger*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 7263–7271, 2017. (Cited in page 86.)

[Redm 18] J. Redmon and A. Farhadi. *Yolov3: An incremental improvement*. arXiv preprint arXiv:1804.02767, 2018. (Cited in page 86.)

[Ren 15] S. Ren, K. He, R. Girshick and J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015. (Cited in page 86.)

[Rose 58] F. Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological Review, vol. 65, no. 6, page 386, 1958. (Cited in page 88.)

[Rubl 11] E. Rublee, V. Rabaud, K. Konolige and G. Bradski. *ORB: An efficient alternative to SIFT or SURF*. In IEEE Int. Conf. on Computer Vision (ICCV), pages 2564–2571, 2011. (Cited in pages 72 and 73.)

[Rusu 08a] R. B. Rusu, N. Blodow, Z. C. Marton and M. Beetz. *Aligning point cloud views using persistent feature histograms*. In IEEE Int. Conf. on Intelligent Robots and Systems (IROS), pages 3384–3391, 2008. (Cited in pages 31, 32, 33, 34, and 35.)

[Rusu 08b] R. B. Rusu, Z. C. Marton, N. Blodow and M. Beetz. *Learning informative point classes for the acquisition of object model maps*. In Int. Conf. on Control, Automation, Robotics and Vision (ICARCV), pages 643–650, 2008. (Cited in pages 33 and 35.)

[Rusu 09] R. B. Rusu, N. Blodow and M. Beetz. *Fast Point Feature Histograms (FPFH) for 3D registration*. In Int. Conf. on Robotics and Automation (ICRA), pages 3212–3217, 2009. (Cited in pages 33, 34, 35, 36, and 72.)

[Scar 11] D. Scaramuzza and F. Fraundorfer. *Visual Odometry [Tutorial]*. IEEE Robotics & Automation Magazine, vol. 18, no. 4, pages 80–92, 2011. (Cited in page 9.)

[Seiw 21] P. Seiwald, S.-C. Wu, F. Sygulla, T. F. C. Berninger, N.-S. Staufenberg, M. F. Sattler, N. Neuburger, D. Rixen and F. Tombari. *LOLA v1.1 – An Upgrade in Hardware and Software Design for Dynamic Multi-Contact Locomotion*. In IEEE-RAS Int. Conf. on Humanoid Robotics (Humanoids), pages 9–16, 2021. (Cited in page 13.)

[Shan 21] T. Shan, B. Englot, F. Duarte, C. Ratti and D. Rus. *Robust Place Recognition using an Imaging Lidar*. arXiv:2103.02111 [cs], March 2021. arXiv: 2103.02111. (Cited in page 73.)

[Spez 20] R. Spezialetti, S. Salti, L. Di Stefano and F. Tombari. 3d local descriptors—from handcrafted to learned, pages 319–352. Springer International Publishing, Cham, 2020. (Cited in pages 32 and 33.)

[Stan 21] P. Stancelova, E. Sikudova and Z. Cernekova. *3D Feature Detector-Descriptor Pair Evaluation on Point Clouds*. In European Signal Processing Conference (EUSIPCO), pages 590–594, 2021. (Cited in page 32.)

[Stas 17] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome and F. Ferro. *TALOS: A new humanoid research platform targeted for industrial applications*. In IEEE-RAS Int. Conf. on Humanoid Robotics (ICHR), 2017. (Cited in pages 14 and 19.)

[Su 15] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller. *Multi-view convolutional neural networks for 3d shape recognition*. In IEEE Int. Conf. on Computer Vision (ICCV), pages 945–953, 2015. (Cited in page 33.)

[Sund 23] M. Sundermeyer, T. Hodan, Y. Labbe, G. Wang, E. Brachmann, B. Drost, C. Rother and J. Matas. *Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects*. arXiv preprint arXiv:2302.13075, 2023. (Cited in page 87.)

[Tan 19] M. Tan and Q. Le. *Efficientnet: Rethinking model scaling for convolutional neural networks*. In Int. Conf. on Machine Learning (ICML), pages 6105–6114. PMLR, 2019. (Cited in page 92.)

[Tang 19] A. Tanguy, D. De Simone, A. I. Comport, G. Oriolo and A. Kheddar. *Closed-loop MPC with Dense Visual SLAM - Stability through Reactive Stepping*. In Int. Conf. on Robotics and Automation (ICRA), pages 1397–1403, 2019. (Cited in page 20.)

[Thom 18] H. Thomas, J.-E. Deschaud, B. Marcotegui, F. Goulette and Y. L. Gall. *Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods*. In Int. Conf. on 3D Vision (3DV), Verone, Italy, September 2018. 3DV2018. (Cited in page 49.)

[Tibs 01] R. Tibshirani, G. Walther and T. Hastie. *Estimating the number of clusters in a data set via the gap statistic*. Jour. of the Royal Statistical Society: Series B (Statistical Methodology), vol. 63, no. 2, pages 411–423, 2001. (Cited in page 63.)

[Tinc 19] G. Tinchev, A. Penate-Sanchez and M. Fallon. *Learning to See the Wood for the Trees: Deep Laser Localization in Urban and Natural Environments on a CPU*. IEEE Robotics and Automation Letters (RAL), vol. 4, no. 2, pages 1327–1334, April 2019. (Cited in page 72.)

[Tomb 10a] F. Tombari, S. Salti and L. Di Stefano. *Unique shape context for 3D data description*. In Proceedings of the ACM workshop on 3D object retrieval, pages 57–62, 2010. (Cited in page 33.)

[Tomb 10b] F. Tombari, S. Salti and L. Di Stefano. *Unique signatures of histograms for local surface description*. In European Conference on Computer Vision (ECCV), pages 356–369. Springer, 2010. (Cited in pages 33 and 73.)

[Tomb 11] F. Tombari, S. Salti and L. Di Stefano. *A combined texture-shape descriptor for enhanced 3D feature matching*. In 2011 18th IEEE international conference on image processing, pages 809–812. IEEE, 2011. (Cited in pages 33 and 73.)

[Tonn 20] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx and A. Del Prete. *SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain*. In Int. Conf. on Robotics and Automation (ICRA), pages 6604–6610, 2020. (Cited in pages 103 and 104.)

[Tras 19] A. W. Trask. Grokking deep learning. Manning Publications, 2019. (Cited in page 90.)

[Trev 13] A. J. B. Trevor, S. Gedikli, R. B. Rusu and H. I. Christensen. *Efficient organized point cloud segmentation with connected components*. Semantic Perception Mapping and Exploration (SPME), pages pp–1, 2013. (Cited in page 13.)

[Tsag 17] N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V.G. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, L. Natale, E. Mingo Hoffman, H. Dallali, N. Kashiri, J. Malzahn, J. Lee, P. Kryczka, D. Kanoulas, M. Garabini, M. Catalano, M. Ferrati, V. Varricchio, L. Pallottino, C. Pavan, A. Bicchi, A. Settimi, A. Rocchi and A. Ajoudani. *WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments*. Jour. of Field Robotics, no. 7, page 1225–1259, 2017. (Cited in pages 13 and 20.)

[Univ] Universal Robots. *The UR10e*. `https://www.universal-robots.com/products/ur10-robot/`. (Cited in page 7.)

[Uy 18] M. A. Uy and G. H. Lee. *PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), May 2018. arXiv: 1804.03492. (Cited in pages 73 and 74.)

[Vida 18] J. Vidal, C.-Y. Lin, X. Lladó and R. Martí. *A Method for 6D Pose Estimation of Free-Form Rigid Objects Using Point Pair Features on Range Data*. Sensors, vol. 18, no. 8, 2018. (Cited in page 87.)

[Wahr 19] D. Wahrmann, A.-C. Hildebrandt, T. Bates, R. Wittmann, F. Sygulla, P. Seiwald and D. Rixen. *Vision-based 3d modeling of unknown dynamic environments for real-time humanoid navigation*. Int. Jour. of Humanoid Robotics (IJHR), vol. 16, no. 01, page 1950002, 2019. (Cited in page 13.)

[Wu 20] S.-C. Wu, K. Tateno, N. Navab and F. Tombari. *SCFusion: Real-time Incremental Scene Reconstruction with Semantic Completion*. In Int. Conf. on 3D Vision (3DV), pages 801–810, 2020. (Cited in page 13.)

[Xia 20] Y. Xia, Y. Xu, S. Li, R. Wang, J. Du, D. Cremers and U. Stilla. *SOE-Net: A Self-Attention and Orientation Encoding Network for Point Cloud based Place Recognition*. arXiv:2011.12430 [cs], November 2020. arXiv: 2011.12430. (Cited in page 74.)

[Yang 20] H. Yang, J. Shi and L. Carlone. *Teaser: Fast and certifiable point cloud registration*. IEEE Transactions on Robotics (T-RO), vol. 37, no. 2, pages 314–333, 2020. (Cited in page 32.)

[Yang 22] J. Yang, W. Xue, S. Ghavidel and S. L. Waslander. *6D Pose Estimation for Textureless Objects on RGB Frames using Multi-View Optimization*, 2022. (Cited in pages 96 and 99.)

[Zhan 19] W. Zhang and C. Xiao. *PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 12436–12445, 2019. (Cited in page 74.)

[Zhou 16] Q.-Y. Zhou, J. Park and V. Koltun. *Fast global registration*. In European Conference on Computer Vision (ECCV), pages 766–782. Springer, 2016. (Cited in page 32.)

[Zhou 19a] X. Zhou, D. Wang and P. Krähenbühl. *Objects as points*. arXiv preprint arXiv:1904.07850, 2019. (Cited in page 87.)

[Zhou 19b] X. Zhou, J. Zhuo and P. Krähenbühl. *Bottom-up object detection by grouping extreme and center points*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 850–859, 2019. (Cited in page 87.)

[Zhou 19c] Y. Zhou, C. Barnes, J. Lu, J. Yang and H. Li. *On the continuity of rotation representations in neural networks*. In IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR), pages 5745–5753, 2019. (Cited in page 92.)

[Zou 23] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye. *Object Detection in 20 Years: A Survey*. Proceedings of the IEEE, vol. 111, no. 3, pages 257–276, 2023. (Cited in page 86.)

Abstract:

This thesis applies to the context of the aeronautical industry. It is conducted in the context of two projects. The first is Robotics For the Future of Aircraft Manufacturing (ROB4FAM). It is a joint laboratory between Airbus Operations and the Gepetto team of LAAS-CNRS. It aims to study the reactive generation of robotic motion for drilling and deburring tasks for the aeronautical industry. The second is the European project H2020 Memory of Motion (Memmo), coordinated by the Gepetto team and in which Airbus Operations is also implied. This project has as objective to develop methods to generate reactive and complex movements independently of the robot architecture. It is based on an extended perception of the environment and a preliminary learning of possible robot motions. Historically, the Gepetto team works on humanoids robots because they present a scientific challenge that requires to develop new concepts. The perception of the environment can be broken down into four main areas: knowing where our tools are, knowing where we are, knowing how much we moved and knowing where we are going. This thesis aims to study the solutions that can be integrated into a humanoid robot in order to solve the problems mentioned above. This work is based on the humanoid robot Talos to perceive the environment using data from its LiDAR. It first shows that it is possible to accurately estimate the position of the robot in its environment. This is achieved by using LiDAR data and by integrating a system developed by the University of Oxford and provided as part of the Memmo project. Secondly, the localization of large objects at long distances, as found in the aeronautical industry, is studied. This localization is looked at using 3D data and geometric descriptors like the Fast Point Feature Histogram. This study is extended to neural networks trained to localize objects carrying little textual information through the use of state-of-the-art methods. In addition, the solution of the kidnapped robot problem is explored using LiDAR information, geometric descriptors and a dictionary mechanism. This problem consists of recognising the surrounding environment at the initialization of the robot localization. Finally, a ground plane detection system is integrated to allow the robot to plan its steps online.

Keywords: Perception, LiDAR, Object localization, SLAM, Place recognition, Plane detection, Humanoid robot

Résumé :

Cette thèse s'applique au contexte de l'industrie aéronautique. Elle a pour cadre deux projets. Le premier est Robotics For the Future of Aircraft Manufacturing (ROB4FAM). Il s'agit un laboratoire joint entre Airbus Operations et l'équipe Gepetto du LAAS-CNRS. Il a pour but d'étudier la génération réactives de mouvements robotiques pour des tâches de perçage et d'ébavurage destinées à l'industrie aéronautique. Le second est le projet Européen H2020 Memory of Motion (Memmo) coordonné par l'équipe Gepetto et dont Airbus Operations est également partenaire. Ce projet a pour but de développer des méthodes pour générer des mouvements réactifs et complexes indépendamment de l'architecture du robot. Il se base pour cela sur une perception étendue de l'environnement et un apprentissage préliminaire des possibles mouvements du robot. Historiquement l'équipe Gepetto travaille sur les robots humanoïdes car ils représentent un challenge scientifique nécessitant de développer de nouveaux concepts. La perception de l'environnement quand à elle peut se découper en 4 grands axes: savoir où sont nos outils, savoir où on est, savoir où on va, et savoir où on met les pieds. Cette thèse vise à étudier les solutions qui peuvent être intégrés dans un robot humanoïde afin de résoudre les problématiques mentionnées. Ce travail s'appuie sur le robot humanoïde Talos pour percevoir l'environnement en utilisant des données issues de son LiDAR. Ces travaux montrent premièrement qu'il est possible d'estimer précisément la position du robot dans son environnement. Ceci est obtenu en utilisant des données LiDAR et par l'intégration d'un système développé par l'Université d'Oxford et fourni dans le cadre du projet Memmo. Ensuite, la localisation d'objets volumineux à longue distance, tels que trouvable dans l'industrie aéronautique, est étudiée. Cette localisation est réfléchie en utilisant les données 3D et des descripteurs géométriques de type Fast Point Feature Histogram. L'étude est étendue aux réseaux de neurones entraînés pour localiser des objets portant peu d'informations textuelles à travers l'utilisation de méthodes issues de l'état de l'art. De plus, la résolution du problème du robot kidnappé est explorée grâce aux informations du LiDAR, de descripteurs géométriques et un mécanisme de dictionnaire. Ce problème se résume à reconnaître l'environnement qui nous entoure à l'initialisation de la localisation du robot. Enfin, un système détectant les plans au sol est intégré pour rendre le robot capable de planifier ses pas en ligne.

Mots clés : Perception, LiDAR, Localisation d'objet, SLAM, Reconnaissance de place, Détection de surface, Robot humanoïde