Localisation d'un robot humanoïde TALOS par information géométrique et visuelle combinée à l'apprentissage

Guillaume Gobin

Rapport de Stage

Localisation d'un robot humanoïde TALOS par information géométrique et visuelle combinée à l'apprentissage

par

Guillaume GOBIN

Equipe GEPETTO du Département Robotique

LAAS - CNRS

&

Master Informatique - Intelligence Artificielle et Reconnaissance des Formes

UNIVERSITÉ TOULOUSE 3 - PAUL SABATIER

Rapport de stage de fin d'études du Master Informatique - IARF, effectué au sein de l'équipe Gepetto (pour l'étude de Mouvement des Systèmes Anthromorphes) du département de Robotique du Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS.

20 Août 2021, Toulouse

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidée lors de la rédaction de mon rapport.

Avant tout, j'adresse mes remerciements à mon maître de stage Olivier STASSE pour ses nombreux conseils, ses explications et son aide apportée lors de l'exécution de mon stage et de la rédaction de ce rapport.

Je tiens à remercier Thibaud LASGUIGNES pour sa bienveillance, pour son aide et pour avoir répondu à mes interrogations lors de mon stage.

Je remercie ma tutrice de stage Sandrine MOUYSSET pour ces conseils et explications qui m'ont été utiles lors de ce stage.

Je remercie l'équipe Gepetto du LAAS, pour m'avoir accueilli, pour leur bienveillance.

Je tiens aussi à remercier ma famille pour son aide apportée à la rédaction de mon rapport.

Résumé

La reconnaissance de lieu est importante dans les applications en robotique, notamment pour détecter les lieux déjà visités, pour détecter des objets afin de les manipuler ou d'éviter des obstacles [1], ou pour localiser un robot dans un environnement connu. La reconnaissance de lieu intervient dans de nombreux sous-problèmes, comme dans le SLAM [2] [3] [4] (pour Simultaneous Localization and Mapping) qui est un procédé qui consiste pour un robot mobile à construire une carte de l'environnement qu'il perçoit et de s'y localiser en même temps. On trouve aussi le problème du robot kidnappé, qui consiste à devoir déterminer la localisation du robot au démarrage dans un environnement connu, sans *a priori* sur sa position et orientation. Durant ce stage, l'objectif a été de déterminer une solution basée sur l'apprentissage pour la localisation d'un robot TALOS dans le contexte du problème du robot kidnappé, cela afin d'améliorer un futur procédé SLAM [5] [6]. Ce rapport propose une formulation du problème de localisation, ainsi que différentes méthodes.

Abstract

The place recognition problem is important for robotics applications, such as detected a loop-closure, to perform objects detection in order to manipulate them or avoid obstacles, to localise a robot in a known map. The place recognition problem proceed in some sub tasks such as the Simultaneous Localization and Mapping (SLAM) [2] [3] [4] process which consist for a mobile robot to build a map of an environment and be able to localize in this map, at the same time. Moreover, the kidnapped robot problem, where the goal is to estimate the robot initial pose in a known map and without any prior on his position and orientation. In this intership, the purpose is to find a learning-based method for TALOS robot localisation in a kidnapped robot problem context. In order to improve a future SLAM approach. This report propose a mathematical problem formulation and several methods.

Table des matières

Page

1 Introduction **5**

1.1 Présentation du laboratoire 6

1.2 Présentation du Master IARF 10

2 Contexte **12**

2.1 Description du problème 12

2.2 Formulation du problème 14

2.3 Revues des méthodologies 17

3 Solutions proposées **24**

3.1 Gestion des données 24

3.2 Description de la stratégie proposée 26

4 Résultats **32**

4.1 Résultats de la première estimation de position avec les FPFH moyennés 32

4.2 Résultats de la première estimation de position avec VLAD-FPFH 33

4.3 Résultats de l'estimation finale de l'état avec VLAD-FPFH 34

5 Discussion et Conclusion **36**

5.1 Discussion sur les problèmes rencontrés 36

5.2 Conclusion 36

Références **40**

Liste des tableaux

4.1 Résultats de la première estimation de la position des FPFH moyennés. Pourcentages de bonnes reconnaissances sur 500 positions avec un *radius* de 0.5, un *maxnn* de 400 et une portée de 20 mètres 33

4.2 Résultats de la première estimation de la position des FPFH moyennés. Statistiques sur les erreurs de reconnaissances sur 500 positions. Avec un *radius* de 0.5, un *maxnn* de 400 et une portée de 20 mètres 33

4.3 Résultats de la première estimation de la position des FPFH moyennés avec $\mathcal{J}'$. Pourcentages de bonnes reconnaissances avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33) 34

4.4 Résultats de la première estimation de la position des FPFH moyennés avec $\mathcal{J}'$. Statistiques sur les erreurs de reconnaissances sur 500 positions avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33) 34

4.5 Résultats de la première estimation de la position avec VLAD-FPFH. Pourcentages de bonnes reconnaissances avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33) 35

4.6 Résultats de la première estimation de la position avec VLAD-FPFH. Statistiques sur les erreurs de reconnaissances sur 500 positions avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33) 35

4.7 Résultats de l'estimation finale avec VLAD-FPFH. Paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33) . . . 35

1 Introduction

L'équipe Gepetto étudie le mouvement des systèmes anthropomorphes. Les outils mathématiques développés sont testés sur le robot Pyrène (visible sur la figure 1.2), un robot humanoïde série TALOS [7] bipède doté de 3 capteurs visuels (un LiDAR, une caméra de profondeur, et une caméra stéréovision). L'objectif de ce stage est de donner la possibilité à Pyrène de se localiser au démarrage sans avoir aucun *a priori* sur sa position et son orientation, pour répondre au problème du robot kidnappé. La perception principale, dont le robot se servira, sera celle du LiDAR. Durant le stage, le robot devra se localiser dans la salle Gérard Bauzil (voir figure 1.1) - lieu des expérimentations robotique du LAAS. La localisation est une étape essentielle pour permettre au robot d'avoir une autonomie lui permettant de se déplacer entre deux points, de manipuler des objets, et d'éviter des obstacles sans l'intervention humaine. Une littérature très riche existe sur le sujet, les papiers pertinents seront présentés dans ce rapport. De plus, pour des raisons de clarté, des formulations et descriptions du problème seront données. Finalement, le rapport montrera les méthodes utilisées, la manière dont elles ont été implémentées, et les résultats obtenus.

FIGURE 1.1 – Lieu des expérimentations de la salle Gérard Bauzil

1.1 Présentation du laboratoire

Dans cette partie, une présentation de l'équipe Gepetto, l'environnement de travail et des outils utilisés sera faite.

1.1.1 Le LAAS, le laboratoire d'analyse et d'architecture des systèmes

Le LAAS est une unité du centre national de la recherche scientifique (CNRS) et rattachée à l'institut des sciences de l'ingénierie et des systèmes (INSIS) et à l'Institut des sciences de l'information et de leurs interactions (INS2I).

Le laboratoire a identifié 5 axes de travaux basés sur les 4 champs disciplinaires qui constituent la spécialité historique du laboratoire (informatique, robotique, automatique et micro et nano systèmes) :

— l'énergie

— la santé et l'environnement

— l'industrie du futur

— l'espace

— le transport et mobilités

Parmis ces 4 disciplines de recherches se répartissent 8 départements scientifiques dans lesquels sont répartis 22 équipes de recherches. Le département de ROB (pour robotique) coordonne les travaux de 3 équipes parmi lesquelles se trouve l'équipe Gepetto dans laquelle mon stage a été effectué.

1.1.2 Gepetto pour l'étude de mouvement des systèmes anthropomorphes

Gepetto est une équipe créée en 2006, d'une quarantaine de personnes composées de chercheurs, post-doctorants, doctorants et de stagiaires. Son activité de recherche est centrée sur l'analyse et la génération de mouvement des systèmes anthropomorphes. Gepetto suit une approche interdisciplinaire axée sur deux objets de recherche : le robot humanoïde et l'homme. Ces activités de recherches se divisent en plusieurs thématiques :

— Planification et commande de mouvements robotiques

— Mouvement humain

— Nouveau actionneurs et muscles artificiels

Ces travaux sont intégrés sur des plate-formes logicielles et expérimentales comme notamment le robot Pyrène, un robot bipède de série TALOS (Figure 1.2).

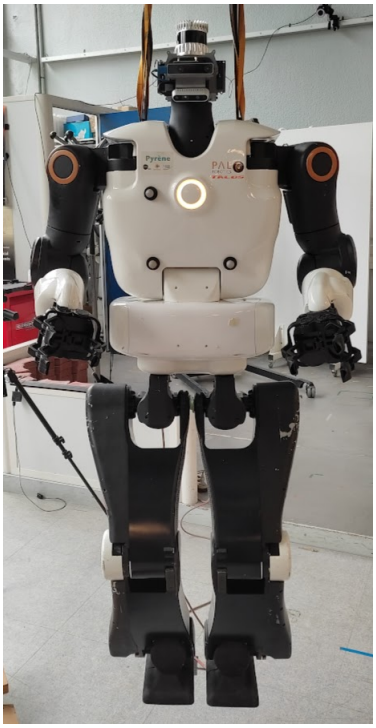

FIGURE 1.2 – Robot Pyrène suspendu dans la salle Gérard Bauzil

1.1.3 Pyrène, le premier robot de la série TALOS

Pyrène est un robot humanoïde d'1 mètre et 75 centimètres (voir sur la figure 1.3), de série TALOS [7] [8], qui est construit et fourni par PAL Robotics - une entreprise siégeant à Barcelone en Espagne et développant des robots comme Tiago et Solo.

L'équipe de Gepetto a modifié la tête du robot pour lui apporter de nouveaux capteurs visuels (voir figure 1.4).

On y trouve un LiDAR, une caméra stéréovision et une caméra de profondeur. Le LiDAR pour "Light Detection And Ranging" est une technique de détection et d'estimation de la distance par la lumière. Autrement dit, le LiDAR va produire des données de la forme d'un nuage de points 3D qui sera les résultats des mesures obtenues autour du capteur. Cette technique permet de s'affranchir

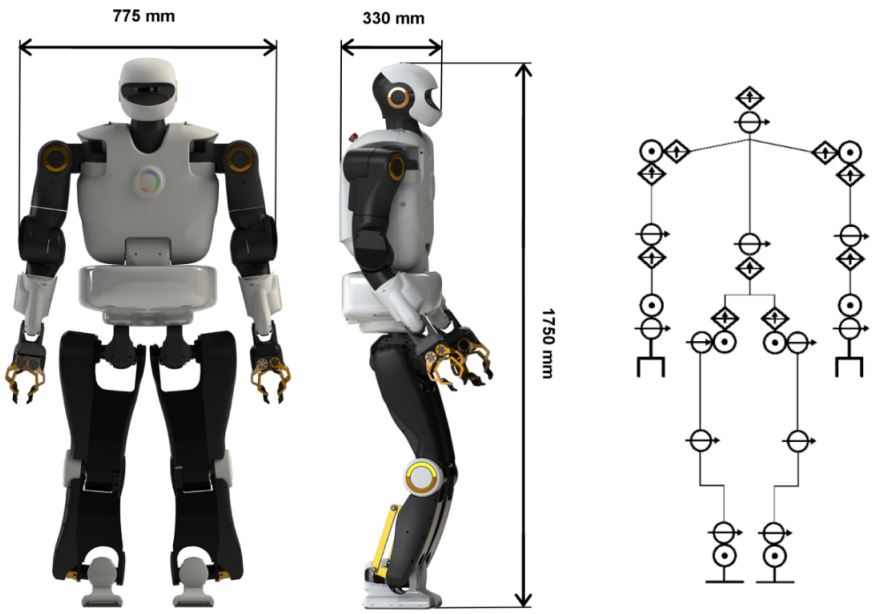

FIGURE 1.3 – Mesures de TALOS en millimètre et schéma de son architecture

des contraintes d'illuminations qu'on retrouve sur des techniques basées sur la caméra. L'intérêt d'utiliser la caméra serait d'apporter une information supplémentaire, comme l'information de la texture et de l'apparence.

Néanmoins, cette information de texture et d'apparence, dans le contexte du projet de Gepetto, est quasi inexistante dû à l'environnement industriel dans lequel évoluera le robot. En effet, les travaux sur Pyrènes s'inscrivent dans une collaboration avec Airbus, dénommé ROB4FAM, et qui a pour conséquence d'avoir des exigences industrielles.

1.1.4 ROB4FAM, une collaboration LAAS-Airbus

ROB4FAM pour *Robots For the Future of Aircraft Manufacturing* est une collaboration entre le LAAS et Airbus visant à développer des technologies et un savoir faire autour de la robotique.

Notamment en permettant à des robots de travailler de manière autonome. Pour mener à bien cet objectif, ROB4FAM divise ses activités de recherches en 4 axes :

— La planification réactive

— Le contrôle en couple

— Le SLAM

— L'équilibre et la marche

FIGURE 1.4 – Tête de Pyrène (on y voit le LiDAR, la caméra stéréovision et la caméra de profondeur - respectivement de haut en bas.)

Mon stage s'intègre dans la partie SLAM où la localisation au démarrage permettra au robot de se localiser par la suite grâce au SLAM [5] [6]. Gepetto collabore avec les équipes de l'université d'Oxford en utilisant un algorithme ICP (pour *Iterative Closest Point*) [8] qu'ils ont développé, nommés AICP (pour *Auto-tuned ICP*) [9]. Cet algorithme s'intègre dans la méthode de localisation et nécessite de connaître la position initiale du robot au démarrage à une précision de 50 centimètres. L'ICP est un algorithme qui a pour but de comparer deux nuages de points 3D et d'en trouver la transformation 3D qui les relie, cela en minimisant itérativement la distance entre ces points. Ainsi, en ayant connaissance de la position du robot avec le nuage de points acquis au démarrage, il est possible, avec la transformation obtenue, de trouver la position suivante ayant acquis le deuxième nuage de points. À noter, que le problème qui vise à trouver la transformation entre deux nuages de points 3D s'appelle le problème de la *registration* 3D.

Ce contexte industriel dans lequel nous place le projet de ROB4FAM implique que l'on connaît l'environnement dans lequel le robot évolue, et justifie l'utilisation principale du LiDAR plutôt que des caméras.

1.1.5 Outils utilisés

Durant ce stage, une modélisation 3D de la carte était à disposition (voir la figure 1.5). De plus qu'une simulation du capteur LiDAR utilisée.

Ces modélisations et simulation peuvent être utilisées grâce à deux logiciels, Gazebo et ROS. Gazebo est un simulateur 3D. Quant à ROS, c'est un *middleware* qui met à disposition des outils informatiques pour le développement de logiciel en robotique. Ainsi, grâce à ROS on peut manipuler et consulter des informations sur Gazebo. Cela permet d'automatiser des simulations et par exemple construire des *datasets*.

Les programmes et scripts développés ont été majoritairement écrits en Python à l'aide de *Visual Code*. De plus, on trouve aussi un variant de XML et du BASH pour d'autres scripts utilisés et développés.

1.2 Présentation du Master IARF

Ce stage s'inscrit dans le programme du Master Informatique en Intelligence Artificielle et Reconnaissances des Formes (IARF) de l'Université Paul Sabatier de Toulouse [10] [11]. Ce Master en informatique de deux années forme ces étudiants aux domaines des systèmes informatiques, de vision par ordinateur, du traitement de la parole, de la robotique, de raisonnement et de la décision par ordinateur.

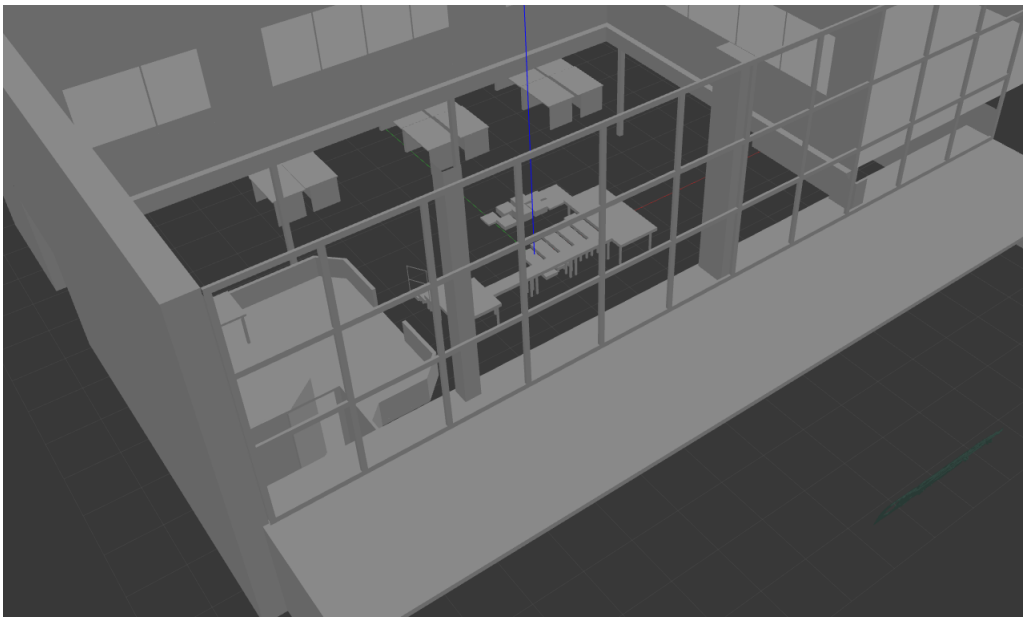

FIGURE 1.5 – Modélisation 3D de la salle Gérard Bauzil

2 Contexte

Cette partie permet de se familiariser avec le problème de localisation, et d'en apprendre plus sur les domaines qui le traitent. On effectuera une description mathématique du problème, ainsi que des formulations algébriques et probabilistes. Pour finir, on exposera quelques travaux trouvés dans la littérature qui nous permettrons d'en savoir plus sur le sujet et de mieux comprendre les techniques utilisées pendant le stage.

2.1 Description du problème

Nous cherchons à localiser le robot Pyrène dans un environnement intérieur, à savoir la salle Bauzil du LAAS, à l'aide de différents capteurs visuels, comme un LiDAR et une caméra stéréovision. On convient que la hanche du robot est son repère de base. Comme le modèle utilisé par Fallon [12] et Ramenazi et al. [13], le robot est considéré comme un système mécanique flottant. Ainsi, nous définissons $T_{\mathcal{B},\mathcal{W}} \in SE(3)$ comme la matrice de transformation de la position et de l'orientation de la hanche dans le repère monde :

$$T_{\mathcal{B},\mathcal{W}} = \begin{bmatrix} R_{\mathcal{B},\mathcal{W}} & P_{\mathcal{B},\mathcal{W}} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

avec SE est le groupe Spécial Euclidien en 3D. Avec $R_{\mathcal{B},\mathcal{W}} \in SO(3)$ et $P_{\mathcal{B},\mathcal{W}} \in \mathbb{R}^3$ sont respectivement la matrice de rotation et la matrice de translation.
Étant donné que Pyrène a un LiDAR et une caméra stéréovision au niveau de la tête, on définit $T_{\mathcal{L},\mathcal{B}} \in SE(3)$ et $T_{\mathcal{C},\mathcal{B}} \in SE(3)$, respectivement les matrices de transformations du LiDAR et de la caméra dans le repère base.

$$T_{\mathcal{L},\mathcal{B}} = \begin{bmatrix} R_{\mathcal{L},\mathcal{B}} & P_{\mathcal{L},\mathcal{B}} \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

$$T_{\mathcal{C},\mathcal{B}} = \begin{bmatrix} R_{\mathcal{C},\mathcal{B}} & P_{\mathcal{C},\mathcal{B}} \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

en utilisant la transformation $T_{\mathcal{L},\mathcal{C}}$ étant donné la position et la matrice de rotation $T_{\mathcal{C},\mathcal{B}}$ de la caméra dans le repère base, la position et la rotation $T_{\mathcal{L},\mathcal{B}}$ du LiDAR sont données par :

$$T_{\mathcal{L},\mathcal{B}} = T_{\mathcal{C},\mathcal{B}} \oplus T_{\mathcal{L},\mathcal{C}} \tag{2.4}$$

où l'opérateur $\oplus$ est utilisé de la même manière que Jo et al. [14] et signifie l'opérateur de composition de position comme le défini Smith et al [15].

Ainsi, le vecteur d'état s au temps k, s_k à estimer, est défini comme ci-dessous :

$$s_k = \begin{bmatrix} R^{(k)}_{\mathcal{B},\mathcal{W}} & P^{(k)}_{\mathcal{B},\mathcal{W}} \end{bmatrix} \tag{2.5}$$

De même que Fallon [12], on peut estimer la vélocité angulaire et linéaire du repère base pour définir le vecteur de contrôle u :

$$u_{k+1} = \begin{bmatrix} v^{(k+1)}_{\mathcal{B},\mathcal{B}} & w^{(k+1)}_{\mathcal{B},\mathcal{B}} \end{bmatrix} \tag{2.6}$$

avec $v_{\mathcal{B},\mathcal{B}} \in \mathbb{R}^3$ et $w_{\mathcal{B},\mathcal{B}} \in \mathbb{R}^3$ signifiant respectivement les vitesses linéaires et angulaires estimées au niveau de la hanche du robot. Ainsi, on peut définir un ensemble de vecteurs d'état s et un ensemble de vecteurs de contrôle u :

$$S_{0:k} = \left\{ s_0,\ s_1,\ \ldots,\ s_k \right\}_{k=Nlocation} \tag{2.7}$$

$$U_{1:k} = \left\{ u_1,\ u_2,\ \ldots,\ u_k \right\}_{k=Ncontrol} \tag{2.8}$$

où $S_{0:k}$ est l'historique des vecteurs d'état du robot et $U_{1:k}$ est l'historique des vecteurs de contrôle du robot. Dans notre cas, il n'y a qu'une seule localisation concernée, donc on fixe k à 0. De plus, $u_0 = \emptyset$ puisque le robot n'a pas de vecteur de contrôle au temps 0.

En considérant nos deux capteurs, LiDAR et caméra, nous avons deux ensembles d'observations. Donc, nous définissons Z_k l'ensemble des échantillons d'observations du LiDAR au temps k

et P_k l'ensemble des échantillons d'observations de la caméra au temps k.

$$Z_k = \Big\{ z_0,\ z_1,\ \ldots,\ z_n \Big\}_{n=NlSample} \tag{2.9}$$

$$P_k = \Big\{ p_0,\ p_1,\ \ldots,\ p_n \Big\}_{n=NcSample} \tag{2.10}$$

avec $z_i \in \mathbb{R}^3$ signifiant le i^{eme} échantillon LiDAR, et $p_i \in \mathbb{R}^3$ signifiant le i^{eme} échantillon caméra. Ainsi, on définit l'historique des observations $O_{1:k}$:

$$O_{1:k} = \Big\{ (Z_1, P_1),\ (Z_2, P_2),\ \ldots,\ (Z_k, P_k) \Big\}_{k=Nobs} \tag{2.11}$$

$$O_{1:k} = \Big\{ o_0,\ o_1,\ \ldots,\ o_k \Big\}_{k=Nobs} \tag{2.12}$$

avec $o_i = (Z_i, P_i)$ la i^{eme} observation.

Finalement, étant donné que nous avons une modélisation de la salle Bauzil et que l'environnement est connu, nous pouvons définir un ensemble m :

$$m = \Big\{ m_0, m_1, \ldots, m_n \Big\}_{n=Nlandmarks} \tag{2.13}$$

avec $m_i \in \mathbb{R}^3$ et signifiant le i^{eme} point de repère de la carte. Un *landmark* [2] [4] [3] est un point de repère qui est une propriété lié à une position donné. La carte m est l'ensemble des points de repère possible que l'on peut détecter.

2.2 Formulation du problème

Dans cette partie, des formulations mathématiques seront proposé avec une formulation algébrique dans un premier temps et une formulation probabiliste dans un second temps.

2.2.1 Formulation algébrique

Dans le contexte du robot kidnappé, donc sans *a priori* sur l'état du robot, on cherche à estimer cet état à la première itération - au démarrage. On va formaliser le problème d'estimation du vecteur d'état s. Avant tout, on précise que le robot a quelques *a priori* sur son état, notamment dû à sa taille

et à son orientation dépendant de sa structure mécanique. On cherche alors à localiser la hanche du robot dans le repère monde. Le robot reste à une hauteur statique, nous pouvons donc simplifier le problème en fixant la position du repère base sur l'axe z à une hauteur de H_{pelvis} au temps $k = 0$:

$$P^{(0)}_{\mathcal{B},\mathcal{W}} = \begin{bmatrix} x \\ y \\ H_{pelvis} \end{bmatrix} \tag{2.14}$$

De plus, la rotation du robot est nulle autour de l'axe des x, à cause de sa structure mécanique. On précise alors $R^{(0)}_{\mathcal{B},\mathcal{W}}$:

$$R^{(0)}_{\mathcal{B},\mathcal{W}} = Rz^{(0)}_{\mathcal{B},\mathcal{W}}(\phi).Ry^{(0)}_{\mathcal{B},\mathcal{W}}(0).Rx^{(0)}_{\mathcal{B},\mathcal{W}}(0) \tag{2.15}$$

avec $Rz^{(0)}_{\mathcal{B},\mathcal{W}}$, $Ry^{(0)}_{\mathcal{B},\mathcal{W}}$, et $Rx^{(0)}_{\mathcal{B},\mathcal{W}}$ signifiant respectivement les matrices de rotations sur l'axe x (*roll*), sur l'axe y (*pitch*) et sur l'axe z (*yaw*) de la base dans le repère monde. Et ϕ l'angle autour de l'axe z. À noter que les angles autour de l'axe x et autour de l'axe y sont fixés à 0.
Finalement, une simplification du vecteur d'état s_k au temps k à estimer est faite :

$$s_k = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \tag{2.16}$$

À noter que, en fonction du contexte, s_k peut être dans le repère LiDAR ou le repère caméra. La formulation du problème est la suivante :

$$\tilde{s_k} = \mathcal{F}(\tilde{O}_{1:k}, \tilde{U}_{1:k}, \tilde{S}_{0:k-1}) \tag{2.17}$$

étant donné l'historique des observations estimées $\tilde{O}_{1:k}$, l'historique des vecteurs de contrôle $\tilde{U}_{1:k}$, et l'historique des précédentes localisations estimées $\tilde{S}_{0:k-1}$, on estime le vecteur d'état s_k du temps k à l'aide d'une fonction $\mathcal{F}$ que l'on cherche.

Sachant que nous sommes dans le contexte du robot kidnappé, les ensembles $\tilde{U}_{1:k}$ et $\tilde{S}_{0:k-1}$ ne sont pas disponibles. De plus que, étant au démarrage, il n'y a que la première observation de disponible o_0.

Autrement dit, pour une vérité de terrain s_k^*, on reconstruit l'observation o_k avec la fonction f :

$$o_k = f(s_k^*) \tag{2.18}$$

où f est la fonction qui nous donne les observations o_k étant donné une position s_k^*. Il est généralement facile de modéliser f, qui est définie par :

$$\tilde{o_k} = \tilde{f}(s_k^*) \tag{2.19}$$

avec $\tilde{f}$ étant la modélisation de la fonction f, qui nous donne une estimation $\tilde{o_k}$ de l'observation étant donné une position s_k^*.

Or, nous pouvons remarquer que $f^{-1} = \mathcal{F}$, et qui à partir d'une observation o_k estime un vecteur d'état $\hat{s_k}$.

$$\hat{s_k} = \mathcal{F}(\tilde{o_k}) \tag{2.20}$$

avec $\hat{s_k} \approx s_k^*$. La difficulté du problème vient du fait qu'inverser f est souvent impossible ou trop difficile. Il est néanmoins possible de construire une approximation de $\mathcal{F}$ en utilisant $\tilde{f}$.

Il se trouve que nous avons une modélisation 3D de la salle et une simulation du capteur. On peut donc construire $\tilde{f}$ et $\tilde{o_k}$ à partir d'une simulation lancé grâce à Gazebo et ROS.

2.2.2 Formulation probabiliste

Il est aussi intéressant de formuler le problème de manière probabiliste, puisqu'il s'agit avant tout d'un problème d'estimation. Ainsi, on définit le modèle d'observation qui décrit la probabilité de produire une observation o_k étant donné un état s_k et une carte connue m :

$$p(o_k|s_k, m) \tag{2.21}$$

Supposons que l'observation o_k est soumise à un bruit Gaussien :

$$o_k = h(s_k, m) + \beta \tag{2.22}$$

avec $h(.)$ la fonction de prédiction de mesure et β provient d'une densité Gausienne à moyenne nulle. Ainsi, on suppose que le modèle d'observation est :

$$p(o_k|s_k,m) = \mathcal{G}(o_k; h(s_k,m), R) \tag{2.23}$$

avec R signifiant la matrice $(n \times n)$ de covariance des mesures et avec la moyenne $h(a_k,m) \in \mathbb{R}^n$, avec n étant le nombre d'échantillons d'une observation. Et $\mathcal{G}$ la fonction gaussienne.

À un temps k, l'observation o_k et la carte m, on effectue une inférence probabiliste, comme donnée ci-dessous :

$$p(s_k|o_k,m) \tag{2.24}$$

et étant donné le logarithme négatif de vraisemblance *NLL* (pour *Negative Log Likelihood*) :

$$NLL(s_k; o_k, m) = -ln\ p(o_k|s_k,m) \tag{2.25}$$

on tente d'estimer l'état s_k^{MLE} du robot, sachant que nous n'avons aucun *a priori* sur $P(s_k)$:

$$s_k^{MLE} = \underset{s_k}{argmax}\ p(o_k|s_k,m) = \underset{s_k}{argmin}\ NLL(s_k; o_k, m) \tag{2.26}$$

2.3 Revues des méthodologies

Le problème de reconnaissance de lieu est très important en robotique. De nombreuses méthodes ont été développées depuis des dizaines d'années. Une revue partielle de ces méthodes va être présentée dans cette partie. Ces méthodes peuvent être différenciées de différentes manières, elles dépendent notamment du type de capteur, du type de robot, et du type de scène. Par exemple, on trouve des méthodes basées sur la caméra, et des méthodes basées sur le LiDAR. L'une va utiliser des connaissances en vision 2D par ordinateur, tandis que l'autre va utiliser des méthodes de perception 3D. On trouve des méthodes différentes en fonction du type de scènes (extérieure ou intérieure).

Dans notre cas, on souhaite trouver la localisation d'un robot en intérieur, dans une salle. Il est équipé d'un LiDAR et de deux caméras. Il sera préférable d'utiliser le LiDAR du robot qui présente un gros avantage comparé à la caméra. En effet, le LiDAR s'émancipe des contraintes

d'illuminations et une caméra perd de son utilité dans un milieu industriel à cause du manque d'information sur l'apparence.

2.3.1 La notion de descripteurs

Un descripteur est une propriété spéciale d'une image dans son ensemble ou d'un objet dans l'image. Il peut s'agir d'une propriété locale ou d'une caractéristique globale de l'image. Ils permettent de caractériser, et de décrire une image pour pouvoir la comparer avec d'autres. On peut aussi dire qu'un descripteur est une représentation qui peut permettre d'identifier un point 3D particulier dans 2 mesures différentes (o_k, o_l). Initialement un point 3D peut être identifié par son apparence extraite de (P_k et de P_l) ou de sa géométrie (Z_k ou Z_l). Le descripteur permet de résoudre un problème discret : associer deux mesures ensemble - par exemple dans Z_k et Z_l, un point (x_k, y_k, z_k) et un point (x_l, y_l, z_l) correspondent au même point 3D m_i dans le monde réel. (Voir le schéma figure 2.1

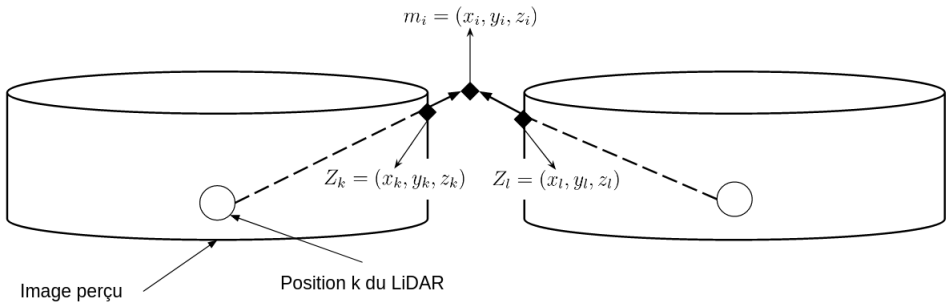

FIGURE 2.1 – Schéma illustrant l'association d'un point 3D réel m_i aux perception LiDAR Z_k et Z_l aux positions k et l.

Une méthode pour faire de la localisation à l'aide des descripteurs est de résoudre le problème (à la manière d'une base de données - clé, valeur, requête) avec un ensemble d'images (clé) associé à une position (valeur), et avec une image acquise (requête). Où les images sont représenté par des descripteurs.

Il y a plusieurs types de descripteurs :

— les descripteurs basés sur des points clés sont calculés autour des singularités d'une image. Il y a potentiellement un grand nombre de descripteurs calculés pour une seule image dépendant du nombre de points d'intérêts.

— les descripteurs basés sur les histogrammes sont calculés en tout point de l'image. Il n'y a donc qu'un seul histogramme par image.

Parmi les descripteurs basés sur des points clés, on peut citer ORB [16], BRIEF [17], ou SIFT [18]. Dans ce cas, le nombre de points clés peut changer et la quantité peut être trop importante en fonction de l'image. Il est donc parfois difficile de comparer des images entre elles (dû au trop grand nombre de descripteurs). C'est pourquoi Jégou et al. [19] propose en 2010 l'*encoding* VLAD qui peut être vu comme un descripteur global de l'image, autrement dit comme une représentation encodée de l'image.

VLAD - Vector of Locally Aggregated Descriptors

La représentation VLAD proposée par Jegou et al. [19] et améliorée par Arandjelovic et Zisserman [20] a son importance dans ce stage, pour cette raison, on explique brièvement son principe. VLAD pour *Vector of Locally Aggregated Descriptors* est comparable aux BoF pour *Bag of Features* (aussi connu sous le nom de *Bag of Words* ou *Bag of Visual Words*).

On a un ensemble C de mots appelés *codebook*. Ces mots sont des descripteurs types. Ce *codebook* est le dictionnaire de tous les descripteurs types qui permettent de décrire une image. Il est souvent appris avec l'algorithme des k-moyennes (aussi connu sous le nom de *k-means*) et où un mot peut être vu comme le centre d'un *cluster*.

Le principe du BoF est de compter le nombre d'apparitions dans l'image de chaque mot. Ainsi étant donné un *codebook* de K mots, on associe à chaque descripteur d de dimension D de l'image son plus proche voisin c_i du *codebook* :

$$c_i = NN_C(d) \tag{2.27}$$

avec $NN_C(.)$ la fonction des plus proches voisins (*Nearest Neighbor*) appliquée sur le *codebook*.

Finalement, on calcule les occurrences des associations pour chaque mot. Pour l'image x, la représentation BoF est :

$$BoF(x) = \begin{bmatrix} Nc_0 \\ Nc_1 \\ \vdots \\ Nc_{K-1} \end{bmatrix}_{Nc_i = Card(\left\{ \forall d \mid NN_c(d) = c_i \right\})} \tag{2.28}$$

avec Nc_i le nombre d'occurrences où le descripteur d appartient au type du i^{eme} mot c_i de C dans x. De plus, K est le nombre total de mots présents dans le *codebook* et $BoF(x) \in \mathbb{R}^K$.

Le principe de VLAD est le suivant : pour chaque association c_i, d que vérifie l'équation (2.27), on accumule la différence de c_i et de d sur chacune des j dimensions. La dimension d'une représentation VLAD est $K \times D$. Ainsi, pour calculer l'élément $v_{i,j}$ de $VLAD(x)$ pour le i^{eme} mot et le j^{eme} composant du descripteur, on a :

$$v_{i,j} = \sum_{\forall d | NN(d)=c_i} d_j - c_{i,j} \tag{2.29}$$

Reconnaissance de lieux à grande échelle

On trouve dans la littérature, des méthodes de localisation à grande échelle qui peuvent être intéressantes.

Certains vont utiliser des méthodes basées sur des grandes bases de données GPS/images comme Zamir et Shah [21]. Le but de leur méthode est de trouver la localisation GPS d'une image utilisant la base de données de "*Google Maps Street View*". Le principe est d'extraire les descripteurs SIFT des images de la base de données et de les indexer dans un arbre. Ensuite, ils trouvent les meilleures correspondances avec les SIFT d'une image requête et un plus proche voisin. Finalement, le résultat est donné à l'aide d'un schéma de vote accumulé.

Certains, au lieu d'utiliser des descripteurs "fait main" (comme le SIFT, l'ORB ou le BRIEF) proposent des méthodes d'apprentissages de descripteurs. C'est ce que propose Arandjelovic et al [22] avec NetVLAD qui est une couche entraînable de réseau de neurones qui apprend une représentation VLAD de l'image. La couche NetVLAD se branche à la sortie d'un réseau de neurones à convolution et donne en sortie un descripteur global permettant une récupération d'images dans une large base de données. La principale différence entre VLAD et NetVLAD est que NetVLAD assigne les descripteurs de manière douce (contrairement à VLAD) et utilise directement les descentes de gradients et les réseaux de neurones pour déterminer le *codebook* (contrairement à VLAD qui utilise un k-moyennes).

2.3.2 Méthodes basées sur le LiDAR

Les méthodes basées sur le LiDAR ont l'avantage de donner des informations géométriques sur la scène et de ne pas être affectées par les conditions d'illuminations. Ce qui rend l'approche intéressante quant à la robustesse. Dans cette partie, nous allons introduire des méthodes basées LiDAR utile pour la reconnaissance de lieux.

Méthodes de localisation basées sur les segments

Le concept de la localisation 3D par segments est présenté par Douillard et al dans [23] et différentes méthodes ont émergé par les travaux de Dubé et al dans [24] et dans [25] et aussi par les travaux de Tinchev et al dans [26]. Ces méthodes sont basées sur l'extraction de segments dans un nuage de points 3D. Un segment est le produit d'un processus de segmentation. L'avantage d'utiliser les segments est la compression du nuage de points 3D en un ensemble d'éléments discriminants. De plus, un segment 3D est un bon compromis entre un descripteur local (sur un point clé) et un descripteur global (sur l'ensemble de la scène).

Par exemple, Ramezani et al. dans [13] présentent une manière robuste de faire de la *registration* et de la détection de fermeture de boucle. Ils utilisent l'état estimé initial de l'odométrie pour initialiser leur *auto-tuned* ICP [9] pour la *registration*. Ensuite, ils utilisent la méthode de Tinchev et al. [26] pour détecter la fermeture de boucle. La fermeture de boucle consiste à détecter la même position dans le monde après un déplacement très grand. Par exemple, détecter que l'on est revenu à un point de départ après avoir fait un circuit.

Méthodes basées sur l'intensité LiDAR

La plupart des méthodes basées LiDAR reposent uniquement sur les informations géométriques données par le LiDAR. Ceci dit, ces méthodes manquent de la forte et discriminante information qu'offre la texture que les caméras peuvent offrir. Quelques travaux proposent de coupler les informations géométriques du LiDAR avec son intensité calibrée retournée. Guo et al. [27] propose un nouveau descripteur appelé ISHOT qui est basé sur l'intensité retournée. Cette intensité est influencée par la réflectance de l'objet nous donnant une information sur sa nature des objets de la scène.

Barsan et al. [28] nous font savoir que la calibration du LiDAR pour effectuer une carte d'intensité est un procédé très laborieux et que de nombreux problèmes sont connus. Pour surmonter ce problème, ils proposent un modèle *deep-learning* qui inclut l'intensité LiDAR ainsi que les scans LiDAR dans un espace où la calibration n'est pas nécessaire.

Shan et al [29] proposent d'utiliser un descripteur basé sur la vision (ORB) pour l'appliquer à une image d'intensité LiDAR et ainsi prendre l'avantage des informations géométriques et visuelles.

Méthodes basé sur des nuages de points 3D

Certains travaux sur les nuages de points 3D peuvent servir aux problèmes de localisation. On trouve notamment des travaux pour des descripteurs comme les FPFH proposés par Rusu dans [30] qui est un histogramme local à un point 3D qui par le calcul de paramètres angulaires et de normals de surfaces va représenter une relation entre un point 3D et ses voisins afin de donner le contexte dans lequel ce point se trouve. Par exemple, s'il se trouve dans un coin, sur un plan, sur un cercle, etc.

De plus, certains travaux tentent d'apprendre les descripteurs d'un nuage de points 3D comme le réseau PointNet [31] proposé par Qi et al. qui étant donné un nuage de points 3D va donner en sortie une classification et segmentation qui peut servir de descripteur global.

Depuis les précédents travaux présentés comme NetVLAD [22] et de PoinNet [31] a émergé PointNetVLAD présenté par Uy et al. [32] qui est une adaptation de NetVLAD au contexte des nuages de points 3D et appliqué à la représentation donnée par PointNet pour finalement donner un descripteur global.

De PointNetVLAD a émergé les travaux de Komorowski [33] qui propose MinkLoc3D reposant essentiellement sur un réseau de neurones basé sur une représentation voxelisée parcimonieuse du nuage de points 3D et sur des architectures convolutives parcimonieuses pour l'extraction de signatures.

Elbaz et al. [34] proposent l'algorithme LORAX qui divise le nuage de points 3D en un ensemble de super-points grâce à l'algorithme RSCS et représente ces super-points dans un nouvel espace utilisant un auto-encoder pour permettre la localisation.

Kim et Kim [35] proposent un nouveau descripteur basé sur le LiDAR : *Scan Context* où il en-

registre directement la structure 3D de l'espace visible depuis le capteur. Ce descripteur est adapté pour des nuages de points bruités. Cette approche permet de calculer la similarité entre deux *Scan Context* afin de pouvoir détecter une fermeture de boucle, notamment.

À propos de fermeture de boucles, Chen et al dans [36] propose OverlapNet, un réseau de neurones siamois qui est spécialisé dans la détection de fermeture de boucles. Il estime le chevauchement et l'angle *yaw* entre deux scans. Ce réseau extrait les informations du scan du LiDAR incluant la profondeur, les normales, et l'intensité notamment.

Méthodes basées sur l'attention

Zhang et Xiao [37] ont développés PCAN pour *Point Cloud Contextual Attention Network* étant un réseau de neurones qui, étant donné un contexte, apprend la représentation d'un nuage de points en utilisant une carte d'attention. Ils utilisent PointNet [31] pour extraire les signatures locales et produire une carte d'attention qui estime les poids d'attentions, basés sur l'information contextuelle, pour chaque point. Finalement, il applique une couche NetVLAD [22] pour avoir les descripteurs globaux.

Xia et al proposent SOE-Net [38], un réseau de neurones qui implémente un module d'*encoding* de l'orientation et une unité d'auto-attention qui se branche au NetVLAD pour permettre une reconnaissance de lieux.

Méthodes basées LiDAR et camera

Komorowski et al [33] présente un descripteur multi-mode basé sur le couplage du LiDAR et d'une caméra. Il extrait les signatures depuis les points 3D avec MinkLoc3D [39] et depuis l'image avec une partie de ResNet18 [40] afin de concaténer les informations des deux descripteurs en un seul descripteur multi-mode.

3 Solutions proposées

Cette partie présente les différentes méthodes proposées pour la localisation. Avec une formulation mathématique des méthodes, une description de leur implémentation, et leurs résultats. De plus, une description des données et de leurs gestions sera faite.

3.1 Gestion des données

Avant de parler des stratégies et méthodes envisagées, il est important de parler de la gestion des données. L'idée est d'apprendre au robot à se localiser dans la salle Bauzil du LAAS à l'aide de capteur LiDAR. Il a donc été nécessaire de construire un ensemble de scènes 3D taguées par leurs positions.

3.1.1 3D Bauzil Dataset

3D Bauzil Dataset est une *Dataset* qui a été produite durant ce stage. Elle a été faite à partir de la salle Bauzil, une grande salle d'expérimentation utilisée dans le LAAS. Cette salle a été modélisée en 3D (voir figure 1.5). De plus, le capteur LiDAR Ouster OS1-64, le capteur utilisé sur Pyrène, est modélisé et simulé. Grâce à la simulation du capteur et la modélisation de la salle Bauzil il a été possible de faire N acquisition à N positions différentes.

La *Dataset* a été construite à partir d'un script python implémenté pendant le stage. Il permet de générer N positions et acquisitions en simulant le LiDAR sur la salle Bauzil modélisé, grâce à Gazebo et ROS. Il va ensuite charger les N positions les unes après les autres, et enregistrer la scène dans un dossier et la position dans un fichier CSV. Autrement dit, on créer un ensemble de paire $(\tilde{o_k}, {s_k}^*)$ étant donné la modélisation $\tilde{f}$. (En référence à l'équation 2.19).

Le script a plusieurs options, il peut :

— générer N positions et orientations de manière aléatoire

— gérer la portée du LiDAR sur la simulation

— générer N positions et orientations en suivant la logique d'une grille dont on précise les dimensions

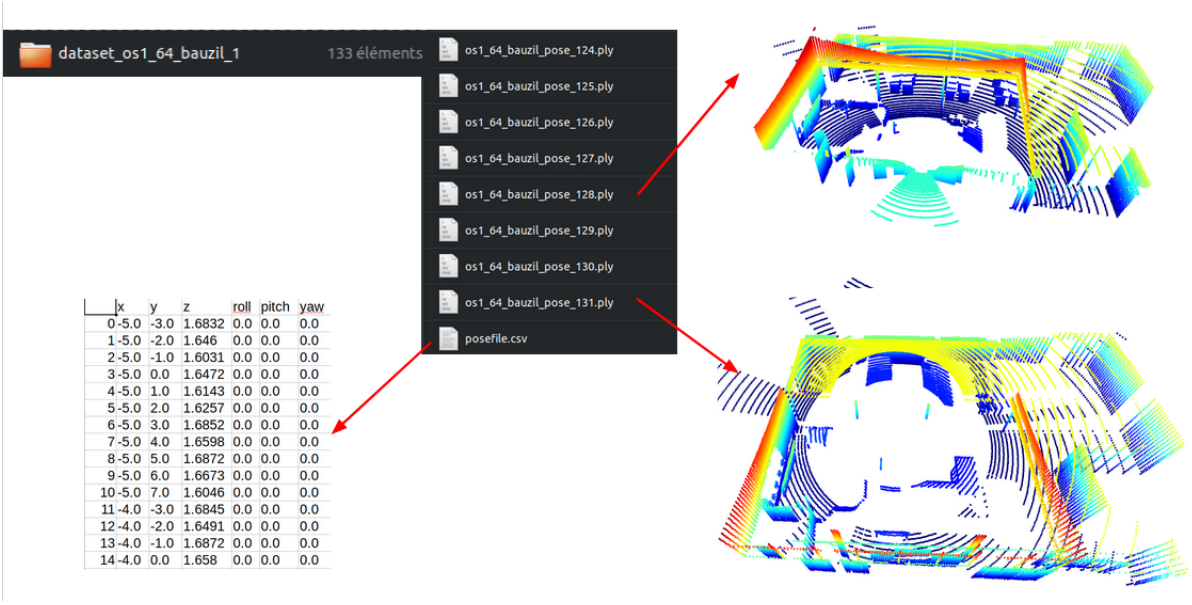

FIGURE 3.1 – Aperçu de l'organisation du dossier d'une *3D Bauzil Dataset*

3.1.2 GT Bauzil Dataset

GT Bauzil Dataset, pour *Ground Truth Bauzil Dataset*, est une *Dataset* produite pendant le stage et fait à partir de plusieurs mesures de nuages de points 3D de la salle Bauzil. Les positions ont été enregistrées grâce à un système de *motion capture* [41] et les points 3D ont été enregistrés à partir d'un LiDAR réel. Le LiDAR réel produit une image de 1024×64 (64 cycles de 1024 échantillons). Le jeu de données a été créé grâce au mécanisme de *rosbag* de ROS qui enregistre chaque évènement sur un topic.

Un script python a été implémenté afin de pouvoir récupérer les événements intéressant. Ainsi, au moment où l'algorithme déterminait qu'il était d'intérêt, c'est-à-dire si le robot était immobile, à une hauteur de moins d'1 mètre 76, et suffisamment loin d'un précédent point d'intérêt, il sauvegardait la scène dans un dossier et la position dans un fichier CSV (comme pour *3D Bauzil Dataset*). Il est à noter que la position du robot a été limité par la zone que couvrent les caméras de *motions captures*. Seule une petite partie de la salle Bauzil a donc été mesurée.

3.1.3 Construction d'image à partir d'une scène 3D

Depuis un nuage de points 3D, on effectue une construction d'images en (cycle×sample). Afin de faire ça, on tire profit du fonctionnement du capteur qui acquiert les échantillons séquentiellement. Il suffit seulement de reconstruire l'image en remplissant les cellules d'une matrice dans la même direction que le capteur acquiert les échantillons.

À noter que la perception du LiDAR équivaut à une image cylindrique comme on peut le voir sur la figure 2.1.

FIGURE 3.2 – Aperçu d'une image 2D construite à partir d'un élément de *3D Bauzil Dataset* dans Gazebo

3.2 Description de la stratégie proposée

Cette partie présente une description des idées et des méthodes utilisées durant ce stage.

3.2.1 Stratégie

Nous tentons de procéder à une localisation à l'aide de mesures LiDAR. La stratégie choisie est inspirée des travaux de Uy et al. avec PointNetVLAD [32]. Cette méthode divise la tâche en deux parties avec un premier module d'extraction de signatures ou de descripteurs locaux, puis un second module de création de descripteurs globaux. Le processus est décrit dans la figure 3.3. Ainsi, la stratégie pour obtenir le descripteur global v_k de l'observation $Z_k \in \mathbb{R}^{N\times 3}$ de N points, au temps k se présente comme ceci :

$$v_k = \mathcal{R}(\mathcal{E}(Z_k)) \tag{3.1}$$

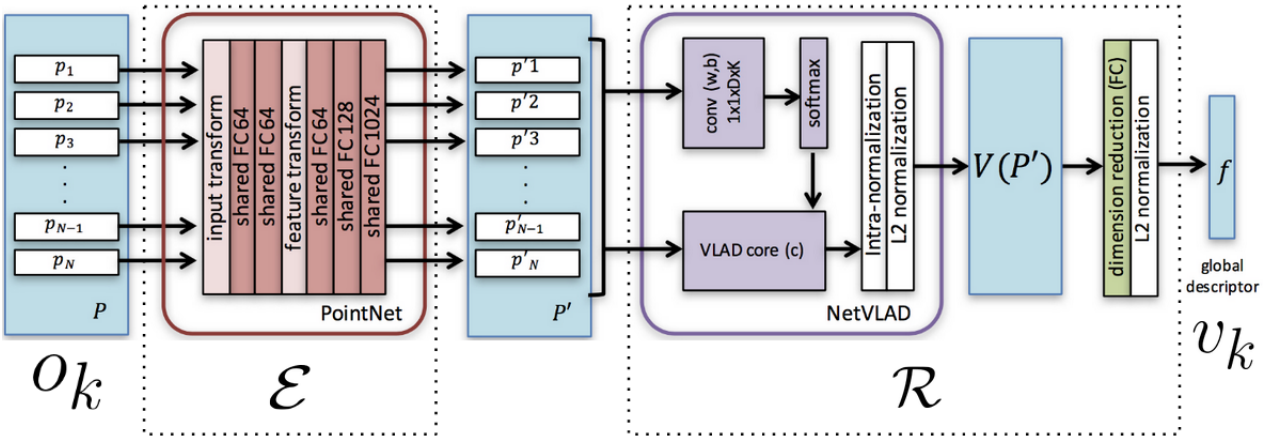

FIGURE 3.3 – Architecture du réseau PointNetVLAD

où $\mathcal{E}(.)$ est la fonction d'extraction de signatures ou de descripteurs locaux, et $\mathcal{R}(.)$ la fonction de représentation pour la création des descripteurs globaux.

Le problème avec PointNetVLAD, c'est que ce réseau doit être entraîné avec énormément de données hétérogènes que nous n'avons pas, puisque notre étude se porte uniquement sur une salle. De plus, la précision renvoyée par PointNetVLAD est de l'ordre de la dizaine de mètres, et nous cherchons une précision à la dizaine de centimètres. C'est pour cela que ce réseau a juste inspiré la stratégie de la méthode.

Une fois que nous avons calculé notre descripteur global v_k nous cherchons le vecteur d'état $\hat{s_k}$ tel que :

$$\hat{s_k} = \underset{s_k}{argmin} \; ||v_k - \mathcal{R}(\mathcal{E}(m; s_k))||_2^2 \tag{3.2}$$

Finalement, si on implémente la carte m comme un KD-Tree, on peut effectuer une recherche par le plus proche voisin NN pour minimiser $||v_k - \mathcal{R}(\mathcal{E}(m; s_k))||_2^2$.

$$\hat{s_k} = \underset{s_k}{NN}(v_k, KDTree(\; \mathcal{R}(\mathcal{E}(m; s_k))\;)) \tag{3.3}$$

Dans la partie finale, on effectue une *registration* entre m_k et $\hat{s_k}$ afin d'avoir la matrice de transformation et ainsi de déduire la position 6DoF du robot.

3.2.2 Extraction des signatures à partir des FPFH

Pour l'extraction des descripteurs locaux, il a été choisit d'utiliser le descripteur FPFH [30]. Le FPFH est un descripteur local de dimension 33. Ainsi, pour l'observation Z_k (équation 2.9), on a :

$$\mathcal{E}(Z_k) = \bigcup_{\forall z \in Z_k} FPFH(z) \tag{3.4}$$

avec $z \in \mathbb{R}^3$ un point 3D :

$$FPFH(z) = \begin{bmatrix} h_1 & h_2 & \dots & h_{33} \end{bmatrix} \tag{3.5}$$

avec h_i étant la i^{eme} valeur de l'histogramme.

Le FPFH a été développé à la base pour permettre de faire de la *registration* 3D à l'aide de descripteurs. Ainsi, en calculant le FPFH sur un point, on obtient une signature suffisamment discriminante pour l'associer au bon point et trouver la transformation recherchée. C'est principalement cette caractéristique discriminante qui a encouragé au choix de ce descripteur. De plus, le FPFH est un variant du PFH qui a une complexité de calcul moins important.

Le FPFH, pour être calculé, prend en compte les points voisins du point étudié. Ainsi, deux paramètres sont nécessaires pour définir le comportement du descripteur :

— *radius* : le rayon de la sphere de centre z_i dans laquelle on va prendre en compte les voisins $z_{i'}$.

— *maximum number of neighbor* : le nombre de points pris en compte seront limités par ce paramètre.

Le FPFH donne un contexte autour d'un point, c'est-à-dire que deux points se trouvant sur un plan auront des signatures plus proches qu'avec un point sur un coin. Ainsi, si on applique un k-moyennes sur les FPFH de chaque point d'un nuage de point, et qu'on associe au point correspondant la couleur de sa classe FPFH, on observe bien une cohérence de la couleur des points en fonction de leur contexte (figure 3.4).

3.2.3 Localisation avec les FPFH moyennés

Dans un premier temps, une étude sur les FPFH a été menée pour déterminer si l'ensemble des signatures pouvait nous donner une information discriminante sur la position. C'est-à-dire, que sur

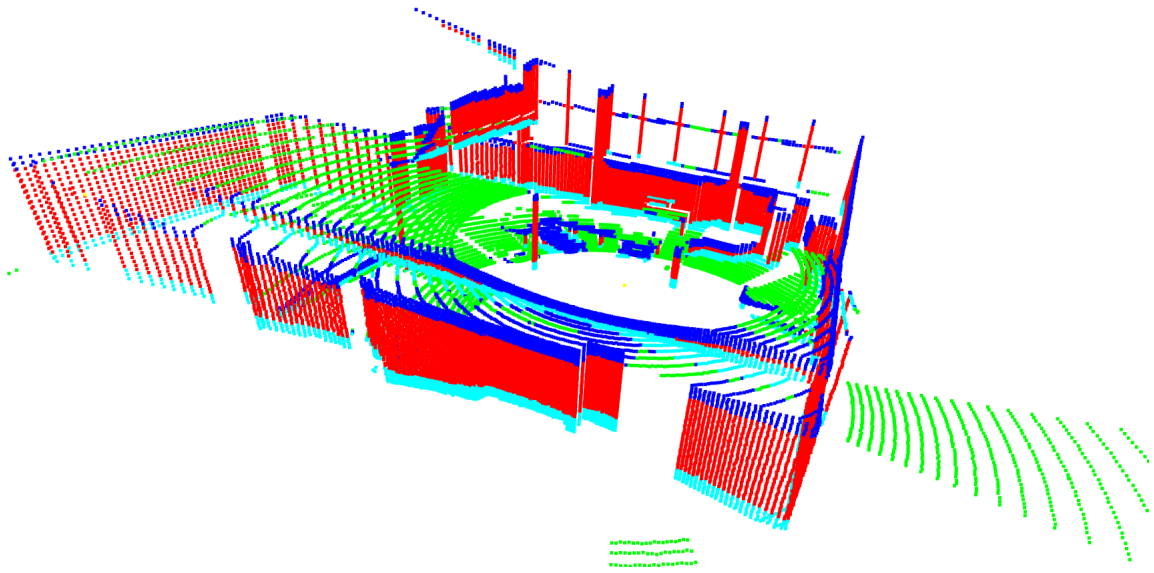

FIGURE 3.4 – Points colorisés en fonction de la classe (donné par k-moyennes) de leur FPFH calculé

deux positions à peu près similaires, la signature sera à peu près similaire, et sur deux positions éloignées, la signature sera différente. Étant donné qu'on a plusieurs milliers de points par scène, il était préférable de compacter les informations. L'idée a été de prendre la somme de tous les FPFHs d'une scène et de normaliser cette somme entre 0 et 1 comme le décrit l'équation suivante :

$$\mathcal{J}(P_s) = \mathcal{N}(\sum_{\forall i \in P_s} FPFH(P_s[i])) \tag{3.6}$$

avec $\mathcal{N}$ la fonction qui normalise de 0 à 1, et i le point 3D appartenant à $P_s \in \mathbb{R}^{N\times 3}$ un nuage de points 3D de N points pour l'état s donné. On aurait pu donner l'observation o_k s'il s'agissait d'une observation, mais P_s fait partie de la base de données et est tagué par son vecteur d'état s.

Ainsi, pour comparer deux scènes, on définit un score de similarité $\mathcal{L}$ étant égale à la distance euclidienne $\mathcal{D}$ de la différence entre ces deux scènes P_s, et $P'_{s'}$.

$$\mathcal{L}(P_s, P'_{s'}) = \mathcal{D}(\mathcal{J}(P_s) \; - \; \mathcal{J}(P'_{s'})) \tag{3.7}$$

Avec ce score $\mathcal{L}$, on peut comparer deux nuages de points entre eux.

De plus, afin d'améliorer la capacité discriminante de cette compression, il a été pensé de faire un encodage à plusieurs canaux. Le premier canal prenant la moyenne des FPFH et le second

prenant l'écart-type. Ainsi, on a :

$$\mathcal{J}'(P_s) = \left[\frac{\mathcal{J}(P_s)}{\sqrt{\mathcal{N}(\sum_{\forall i \in P_s} FPFH(P_s[i])^2) - \mathcal{J}(P_s)^2}} \right] \tag{3.8}$$

3.2.4 Localisation avec l'encodage VLAD des FPFH

Afin, d'avoir une première estimation de la position du robot, la méthode consiste à encoder les nuages de points 3D grâce à la représentation VLAD [19]. Afin de faire un encodage VLAD, nous avons besoin de construire un dictionnaire de descripteurs mots. On le fait avec l'algorithme des k-moyennes. Ensuite, étant donné notre dictionnaire de K mots et notre descripteur FPFH de dimension 33, on calcule l'encodage VLAD (comme expliqué ici [2.29]) sur toutes les scènes et on se retrouve avec des représentation en $K \times 33$ (avec $K << N$). Finalement, on retrouve notre première estimation, comme expliqué en [3.2, 3.3] nous donnant un nombre de candidats. Ensuite, on raffine la position et estime l'orientation en faisant une *registration* 3D à l'aide de RANSAC [42] avec la requête comme source et les candidats comme cibles. Les transformations initiales sont les premières estimations données par les candidats. Finalement, la transformation finale sera celle qui permet d'avoir un *fitness* le plus faible. Ce paramètre est calculé à partir du RANSAC et évalue la réussite de la *registration* 3D.

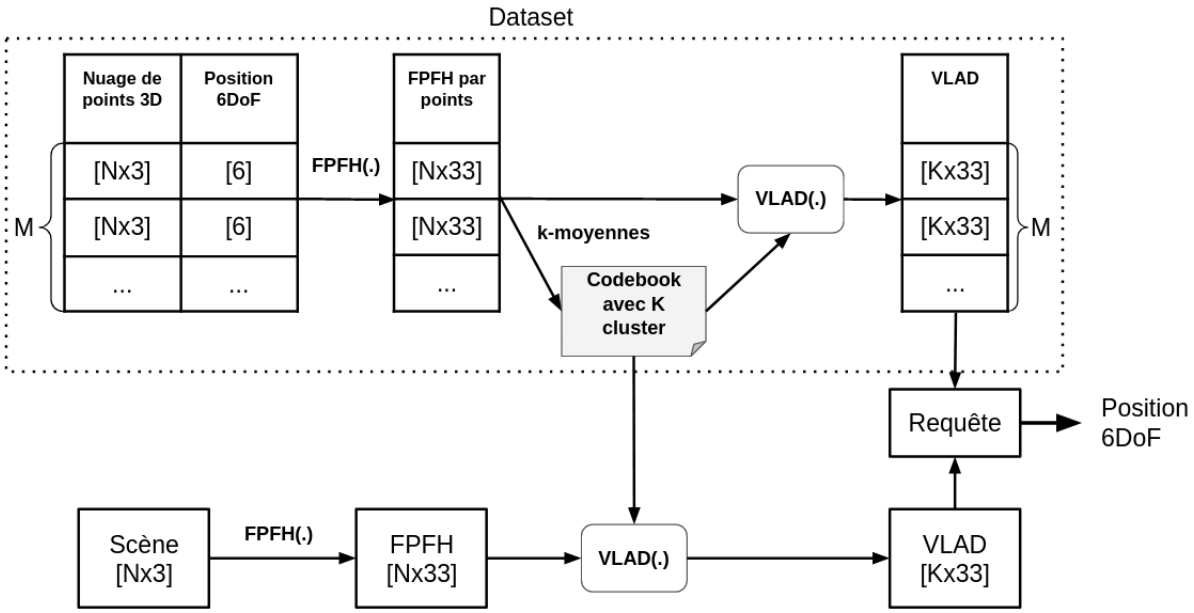

FIGURE 3.5 – Récapitulatif de la localisation avec VLAD et FPFH

4 Résultats

4.1 Résultats de la première estimation de position avec les FPFH moyennés

Afin de procéder à la récupération de la position avec $\mathcal{L}$, on calcul $\mathcal{J}(P_s)$ pour tout nuage de point 3D P_s du jeu de données. Ensuite, on compare les observations 3D $o_{s'}$ acquises depuis une position s' aléatoire inconnue dans la simulation de la *3D Bauzil Dataset*. L'état reconnu $\hat{s}$ est égale à l'état s qui donne P_s et qui minimise la distance $\mathcal{L}$ avec $o_{s'}$.

$$\hat{s} = \underset{s}{argmin}\ \mathcal{L}(P_s, o_{s'}) \tag{4.1}$$

L'état le plus proche s^* est l'état dans le jeu de données qui est le plus similaire à s', il est le meilleur état que l'on puisse reconnaître.

$$s^* = \underset{s}{argmin}\ \mathcal{D}(s - s') \tag{4.2}$$

De plus, l'état s' est exprimé en tant que vérité de terrain.

Ainsi, on génère 500 positions aléatoires sur la simulation du LiDAR Ouster OS1-64 et de la modélisation de Bauzil, et compare avec la *3D Bauzil Dataset*, et enfin, on regarde les 5 meilleurs candidats. La simulation c'est faite sur une portée maximale de 20 mètres. Et les paramètres des FPFH étaient d'un *radius* de 0.5 mètre, et d'un *maximum number of neighbor* de 4 000. On obtient dans le top 5 des meilleurs candidats 68.4% de bonnes reconnaissances comme nous le montre ce tableau (4.1). Avec plus de précision donné par ce tableau (4.2)

La distance de *GT and Recognize* est l'erreur effective entre la reconnaissance et la vérité terrain. La distance de *Recognize and Nearest* représente l'erreur d'avoir eu la meilleure réponse possible. La distance de *GT and Nearest* représente la meilleure erreur que l'on puisse avoir avec cette *dataset*, et ces paramètres. Elle montre les limites des méthodes.

Nombre de candidats pris en compte	Pourcentage cumulé
Top 1	31.60%
Top 2	47.00%
Top 3	57.20%
Top 4	63.60%
Top 5	68.40%

TABLE 4.1 – Résultats de la première estimation de la position des FPFH moyennés. Pourcentages de bonnes reconnaissances sur 500 positions avec un *radius* de 0.5, un *maxnn* de 400 et une portée de 20 mètres

Distance	Moyenne	Median	Min	Max	RMS	STD
GT and Recognize	1.699	0.973	0.014	11.817	2.514	1.852
Recognize and Nearest	1.594	1.000	0.000	11.180	2.470	1.886
GT and Nearest	0.410	0.400	0.0140	1.102	0.451	0.186

TABLE 4.2 – Résultats de la première estimation de la position des FPFH moyennés. Statistiques sur les erreurs de reconnaissances sur 500 positions. Avec un *radius* de 0.5, un *maxnn* de 400 et une portée de 20 mètres

De plus, si on applique $\mathcal{J}'$ comme présenté ici (3.8) on obtient les résultats suivants donnés par le tableau (4.3) et le tableau (4.4).

4.2 Résultats de la première estimation de position avec VLAD-FPFH

Afin d'avoir le vecteur d'état estimé $\hat{s}$, la méthode minimise :

$$\hat{s} = \underset{s}{argmin}\, VLAD(P_s) - VLAD(o_{s'}) \tag{4.3}$$

avec P_s le nuage de points 3D de la *3D Bauzil Dataset* et $o_{s'}$ une acquisition aléatoire ayant une position s' prétendu inconnu. Les résultats sont présentés dans les tableaux (4.5) et (4.7).

Nombre de candidats pris en compte	Pourcentage cumulé
Top 1	55.2%
Top 2	73.4%
Top 3	80.4%
Top 4	84.2%
Top 5	87.2%

TABLE 4.3 – Résultats de la première estimation de la position des FPFH moyennés avec $\mathcal{J}'$. Pourcentages de bonnes reconnaissances avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33)

Distance	Mean	Median	Min	Max	RMS	STD
GT and Recognize	1.013	0.533	0.014	10.950	1.908	1.616
Recognize and Nearest	0.861	0.000	0.000	10.770	1.871	1.662
GT and Nearest	0.411	0.402	0.014	1.102	0.451	0.186

TABLE 4.4 – Résultats de la première estimation de la position des FPFH moyennés avec $\mathcal{J}'$. Statistiques sur les erreurs de reconnaissances sur 500 positions avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33)

4.3 Résultats de l'estimation finale de l'état avec VLAD-FPFH

Cette partie est additionnelle et n'a pas été rédigé dans le rapport de stage rendu à l'Université

Ici, on présente les résultats finaux sur l'estimation finale de la localisation cherchant à estimer s_0 (2.16). Tout d'abord, 5 versions de la modélisation 3D de Bauzil ont été utilisé. La première version est celle par défaut. Des modifications additionnelles ont été apportées aux versions suivantes. Ces modifications sont l'ajout d'objets, la suppression d'objet, la modification de la taille, de la position et de l'orientations d'objets. De plus que l'ouverture et la fermeture des portes ou des volets. Plus l'itération de la version augmente, plus le nombre de modifications est important par rapport à la première version. De plus, il a été utilisé deux *GT Bauzil Dataset*.

Nombre de candidats pris en compte	Pourcentage cumulé
Correct recognition	76.4%
Top 2	91.4%
Top 3	94.8%
Top 4	95.8%
Top 5	97.0%

TABLE 4.5 – Résultats de la première estimation de la position avec VLAD-FPFH. Pourcentages de bonnes reconnaissances avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33)

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

TABLE 4.6 – Résultats de la première estimation de la position avec VLAD-FPFH. Statistiques sur les erreurs de reconnaissances sur 500 positions avec une portée de 20 mètres, avec ces paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33)

Dataset	Requête	Nbr	Réussite	Erreur (x,y)	Erreur (yaw)
3D Bauzil Dataset v1	3D Bauzil v1	500	99.6 %	0.0775 m	1.18 °
3D Bauzil Dataset v1	3D Bauzil v5	500	92.6 %	0.1975 m	2.4865 °
GT Bauzil Dataset	GT Bauzil	13	100 %	0.1784 m	1.6538 °

TABLE 4.7 – Résultats de l'estimation finale avec VLAD-FPFH. Paramètres : (FPFH *radius* : 0.5 mètre, FPFH *maxnn* : 4000, nombre de centre de *cluster* du dictionnaire : 33)

5 Discussion et Conclusion

5.1 Discussion sur les problèmes rencontrés

Durant ce stage, plusieurs problèmes ont été rencontrés. Notamment, l'inéluctable problème de la Covid-19 qui a rendu ce stage difficile d'un point de vue sociale et mentale. Ce problème a bien été géré à l'aide de deux réunions hebdomadaires : l'une avec l'équipe de Gepetto qui travaille sur ROB4FAM, et l'autre avec Olivier STASSE et Thibaud LASGUIGNES. De plus, Element, un réseau de communication, a aidé à garder contact efficacement. D'un point de vue technique, certains problèmes sont survenues :

— liés à l'étendue du domaine d'étude qui complexifie la compréhension,

— venant des environnements Linux et ROS utilisés, où un temps d'adaptation a été nécessaire

5.2 Conclusion

Ce rapport de stage a proposé une formulation et description du problème du robot kidnappé dans le contexte d'une scène en intérieur avec peu de données. Nous avons proposé des méthodes de reconnaissance de lieu inspiré de travaux de reconnaissance de lieu en extérieur et fonctionnant avec des capteurs LiDAR. Les méthodes sont encourageantes, et si elles sont suivies par d'autres processus de raffinement et de robustesses, elles permettront d'aboutir à une estimation plus précise et plus juste de la position. Ces méthodes seront développées durant la suite du stage. Ces stratégies post-rapport ont commencé à être implémenté, mais les résultats n'ont pas encore aboutit. Leur aboutissement viendra sûrement après le rendu de ce rapport. L'idée est d'utiliser la puissance des réseaux de neurones à convolution sur des images 2D. Du coup, on construit une image 2D à partir d'un nuage de points 3D, comme décrit dans cette partie (3.1.3). Associé aux premières stratégies, ces images pourront permettre de raffiner la position et de limiter les erreurs.

References

[1] T. FOISSOTTE, O. STASSE, A. ESCANDE, P.-B. WIEBER et A. KHEDDAR, « A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot », Robotics and Automation, ICRA, 2009.

[2] O. STASSE, « SLAM and Vision-based Humanoid Navigation », Humanoid Robotics: A Reference, 2018.

[3] H. DURRANT-WHYTE et T. BAILEY, « Simultaneous localization and mapping: part I », Institute of Electrical and Electronics Engineers (IEEE), Robotics & Automation Magazine, 2006.

[4] T. BAILEY et H. DURRANT-WHYTE, « Simultaneous localization and mapping (SLAM): part II », Institute of Electrical and Electronics Engineers (IEEE), Robotics Automation Magazine, 2006.

[5] N. KWAK, O. STASSE, T. FOISSOTTE et K. YOKOI, « 3D grid and particle based SLAM for a humanoid robot », 9th IEEE-RAS International Conference on Humanoid Robots, 2009.

[6] M. GARCIA, O. STASSE, J.-B. HAYET, C. DUNE, C. ESTEVES et J.-P. LAUMOND, « Vision-guided motion primitives for humanoid reactive walking: Decoupled versus coupled approaches », The international journal of robotics research, 2015.

[7] O. STASSE, T. FLAYOLS, R. BUDHIRAJA et K. e. a. GIRAUD-ESCLASSE, « TALOS: A new humanoid research platform targeted forindustrial applications », International Conference on Humanoid Robotics (ICHR), 2017.

[8] T. LASGUIGNES, I. MAROGER, M. FALLON, M. RAMENAZI, L. MARCHIONNI, O. STASSE et N. MANSARD, « ICP Localization and Walking Experiments on the Humanoid Robot TALOS », Submitted : International Conference on Advanced Robotics (ICAR), 2021.

[9] S. NOBILI, R. SCONA, M. CARAVAGNA et M. FALLON, « Overlap-based ICP tuning for robust localization of a humanoid robot », IEEE International Conference on Robotics and Automation (ICRA), 2017.

[10] Syllabus Master 1 IARF, disp. à l'adr. : https://www.univ-tlse3.fr/syllabus/SYL_M1_INF-IARF.pdf.

[11] Syllabus Master 2 IARF, disp. à l'adr. : https://www.univ-tlse3.fr/syllabus/SYL_M2_INF-IARF.pdf.

[12] M. FALLON, « Accurate and robust localization for walking robots fusing kinematics, inertial, vision and LIDAR », Interface Focus, Royal Society, 2018.

[13] M. RAMEZANI, G. TINCHEV, E. IUGANOV et M. FALLON, « Online LiDAR-SLAM for Legged Robots with Robust Registration and Deep-Learned Loop Closure », IEEE International Conference on Robotics and Automation (ICRA), 2020.

[14] H. JO et E. KIM, « New Monte Carlo Localization Using Deep Initialization: A Three-Dimensional LiDAR and a Camera Fusion Approach », Institute of Electrical and Electronics Engineers (IEEE) Access, 2020.

[15] R. SMITH, M. SELF et P. CHEESEMAN, « A stochastic map for uncertain spatial relationships », Proceedings of 4th International Symposium on Robotics Research, 1987.

[16] E. RUBLEE, V. RABAUD, K. KONOLIGE et G. BRADSKI, « ORB: an efficient alternative to SIFT or SURF », International Conference on Computer Vision (ICCV), 2011.

[17] M. CALONDER, V. LEPETIT, C. STRECHA et P. FUA, « BRIEF: Binary Robust Independent Elementary Features », European Conference on Computer Vision (ECCV), 2010.

[18] D. LOWE, « Object recognition from local scale-invariant features », Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV), 1999.

[19] H. JÉGOU, M. DOUZE, C. SCHMID et P. PEREZ, « Aggregating local descriptors into a compact image representation », Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[20] R. ARANDJELOVIC et A. ZISSERMAN, « All About VLAD », IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[21] A. R. ZAMIR et M. SHAH, « Accurate Image Localization Based on Google Maps Street View », European Conference on Computer Vision (ECCV), 2010.

[22] R. ARANDJELOVIC, P. GRONAT, A. TORII, T. PAJDLA et J. SIVIC, « NetVLAD: CNN Architecture for Weakly Supervised Place Recognition », CVF Computer Vision Fondation, 2016.

[23] B. DOUILLARD, A. QUADROS, P. MORTON, J. P. UNDERWOOD, M. D. DEUGE, S. HUGOSSON, M. HALLSTRÖM et T. BAILEY, « Scan segments matching for pairwise 3D alignment », IEEE International Conference on Robotics and Automation (ICRA), 2012.

[24] R. DUBÉ, A. CRAMARIUC, D. DUGAS, J. NIETO, R. SIEGWART et C. CADENA, « SegMap: 3D Segment Mapping using Data-Driven Descriptors », Robotics: Science and Systems XIV, 2018.

[25] R. DUBÉ, D. DUGAS, E. STUMM, J. NIETO, R. SIEGWART et C. CADENA, « SegMatch: Segment based loop-closure for 3D point clouds », IEEE International Conference on Robotics and Automation (ICRA), 2017.

[26] G. TINCHEV, A. PENATE-SANCHEZ et 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, 2019.

[27] J. GUO, P. V. K. BORGES, C. PARK et A. GAWEL, « Local Descriptor for Robust Place Recognition using LiDAR Intensity », Submitted : ICRA 2019.

[28] « Learning to Localize Using a LiDAR Intensity Map », Presented at the 2nd Conference on Robot Learning (CoRL), 2018.

[29] T. SHAN, B. ENGLOT, F. DUARTE, C. RATTI et D. RUS, « Robust Place Recognition using an Imaging Lidar », International Conference on Robotics and Automation (ICRA), 2021.

[30] R. B. RUSU, N. BLODOW et M. BEETZ, « Fast Point Feature Histograms (FPFH) for 3D registration », IEEE International Conference on Robotics and Automation (ICRA), 2009.

[31] C. R. QI, H. SU, K. MO et L. J. GUIBAS, « PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation », Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[32] M. A. UY et G. H. LEE, « PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition », CVF Computer Vision Fondation, 2018.

[33] J. KOMOROWSKI, M. WYSOCZANSKA et T. TRZCINSKI, « MinkLoc++: Lidar and Monocular Image Fusion for Place Recognition », Accepted for International Joint Conference on Neural Networks (IJCNN) 2021.

[34] G. ELBAZ, T. AVRAHAM et A. FISCHER, « 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder », IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, Honolulu, HI.

[35] G. KIM et A. KIM, « Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map », IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

[36] X. CHEN, T. LÄBE, A. MILIOTO, T. RÖHLING, O. VYSOTSKA, A. HAAG, J. BEHLEY et C. STACHNISS, « OverlapNet: Loop Closing for LiDAR-based SLAM », Robotics : Science and Systems (RSS), 2020.

[37] W. ZHANG et C. XIAO, « PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval », Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[38] Y. XIA, Y. XU, S. LI, R. WANG, J. DU, D. CREMERS et U. STILLA, « SOE-Net: A Self-Attention and Orientation Encoding Network for Point Cloud based Place Recognition », Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[39] J. KOMOROWSKI, « MinkLoc3D: Point Cloud Based Large-Scale Place Recognition », Winter Conference on Applications of Computer Vision (WACV), 2021.

[40] K. HE, X. ZHANG, S. REN et J. SUN, « Deep Residual Learning for Image Recognition », IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[41] Motion Capture, disp. à l'adr. : `https://fr.wikipedia.org/wiki/Capture_de_mouvement`.

[42] D. KONSTANTINOS G., « Overview of the RANSAC Algorithm », 2010.