HAL
open science

Localisation et navigation d'un robot humanoïde en environnement domestique

Émilie Wirbel

▶ **To cite this version:**

Émilie Wirbel. Localisation et navigation d'un robot humanoïde en environnement domestique. Robotique [cs.RO]. Ecole Nationale Supérieure des Mines de Paris, 2014. Français. NNT : 2014ENMP0058 . tel-01144073

HAL Id: tel-01144073

https://pastel.hal.science/tel-01144073v1

Submitted on 20 Apr 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n° 432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

THÈSE

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure des Mines de Paris

Spécialité doctorale "Informatique temps réel, robotique et automatique"

présentée et soutenue publiquement par

Émilie WIRBEL

le 7 octobre 2014

Localisation et navigation d'un robot humanoïde en environnement domestique

Directeur de thèse : **Arnaud DE LA FORTELLE**
Maître de thèse : **Fabien MOUTARDE**

Jury

David Filliat,	Professeur, ENSTA ParisTech	Rapporteur
Olivier Stasse,	Chargé de recherche, LAAS CNRS	Rapporteur
Roland Chapuis,	Professeur, Institut Pascal	Examinateur
Rodolphe Gelin,	Ingénieur, Aldebaran	Examinateur
Patrick Rives,	Directeur de recherche, INRIA Sophia Antipolis	Examinateur
Pierre Vandergheynst,	Professeur, EPFL	Examinateur
Arnaud de La Fortelle,	Professeur, CAOR Mines ParisTech	Examinateur
Fabien Moutarde,	Maître-assistant, CAOR Mines ParisTech	Examinateur

MINES ParisTech
Centre de Robotique - CAOR Mines ParisTech,
60 boulevard Saint-Michel, 75006 France

THESE

Cette publication n'engage que son auteur et la société Aldebaran ne saurait être tenue responsable de son contenu. Les robots présentés, en particulier le robot Pepper, sont ceux utilisés dans le cadre de la thèse et ne sont pas nécessairement représentatifs des produits finis.

Remerciements

Je tiens tout d'abord à remercier mes encadrants de thèse Arnaud de La Fortelle et Fabien Moutarde, pour leur suivi régulier et leurs conseils tout au long de la thèse, ainsi que Bruno Steux, qui n'aura malheureusement pas pu suivre la thèse jusqu'au bout mais dont l'encadrement pendant ma première année a été précieux. Merci à Arnaud d'avoir trouvé du temps au milieu de son emploi du temps de ministre, à Fabien pour son sérieux qui l'a même fait relire et commenter ce manuscrit depuis son lit d'hôpital, et à Bruno pour son enthousiasme communicatif, une des raisons qui m'ont poussée à entreprendre cette thèse.

Je remercie également ceux qui m'ont suivie du côté d'Aldebaran, tout particulièrement Rodolphe Gelin et Petra Koudelkova, sans qui cette thèse n'aurait jamais été possible. Ils ont été d'une aide précieuse pour la construction de cette CIFRE et mon intégration à la fois dans Aldebaran et les projets collaboratifs associés. De même, je remercie mes différents chefs d'équipe, Gwennaël Gaté, David Gouaillier et Cyrille Collette, tous les trois docteurs et donc compréhensifs vis-à-vis des contraintes d'une thèse, pas toujours compatibles avec le fonctionnement et les impératifs du reste de l'équipe.

Tous mes remerciements pour les membres de mon jury de s'être intéressés à mon sujet et déplacés, depuis assez loin pour certains, pour venir m'écouter. J'espère que le charme légendaire de NAO aura opéré une fois de plus et qu'ils ne sont pas repartis déçus !

Merci aux membres de mon équipe qui m'ont supportée et encouragée à Aldebaran : les gens de Perception, en particulier Guillaume et Corentin pour leur soutien sur toutes les questions de vision sur le robot, puis ceux d'Agility, Aldenis, Cyrille, Jory, Juan, Justine, Lucas, les deux Nicolas et les deux Sébastien. Vous avez fêté avec moi les succès de mes algorithmes, discutés de problèmes subtils d'intégration, supporté mes coups de gueule contre les pauvres robots, le code ou les lentilles, partagé avec moi des quantités inavouables de chocolat et autres sucreries. Bref, une excellente équipe chez qui je me suis toujours bien sentie. Je pourrais citer aussi toute une série d'Aldebaraniens, mais la thèse finirait par devenir un peu trop longue, alors je vais juste me contenter de remercier globalement tous ces gens extraordinaires que j'ai pu rencontrer.

Côté laboratoire, tous les permanents et doctorants ont été toujours accueillants, même si mon passage dans les locaux a été intermittent et que je n'ai pas toujours pu accorder à chacun le temps qu'il méritait. Merci en particulier à ceux qui m'ont accueillie pendant ma période de rédaction, et qui m'ont aidé à trouver un environnement propice. Merci à Christine et Christophe pour leur aide précieuse dans tous les processus administratifs.

Je remercie aussi ma famille et mes amis, qui m'ont soutenue et encouragée pendant toute cette période, en s'intéressant à mes petits robots et à mon travail. C'était un plaisir de voir vous enthousiasmer quand je ramenais un robot à la maison, ou bien suivre avec passion les apparitions d'Aldebaran dans les médias.

Et évidemment, Nadime, toi qui as été à mes côtés pendant ces trois ans pour le meilleur et pour le pire, je te remercie du fond du cœur. Tu as su être intéressé et attentionné, tout en continuant à mener bravement ta propre thèse. Celle-ci t'est dédiée.

Table des matières

Table des matières iii

1 Introduction : contexte et plateformes 1
1.1 Contexte et cas d'utilisations . . . 1
1.1.1 Le contexte de la robotique de service . . . 1
1.1.2 Aldebaran . . . 4
1.1.3 Le CAOR . . . 5
1.1.4 Circonscription du problème . . . 6
1.2 Description des plateformes disponibles . . . 8
1.2.1 NAO . . . 8
1.2.2 ROMEO . . . 10
1.2.3 Pepper . . . 12
1.2.4 Le simulateur Webots . . . 14
1.2.5 Système d'exploitation : NAOqiOS . . . 15
1.3 Conclusion : un problème essentiel et des robots contraints . . . 16

2 Problématique de la localisation et de la navigation 19
2.1 Description du problème général de la localisation et de la navigation . . . 19
2.2 Le formalisme métrique . . . 21
2.2.1 Capteurs métriques . . . 21
2.2.2 Capteurs non métriques : caméras . . . 22
2.2.3 Position du problème par rapport au formalisme métrique . . . 25
2.3 Le formalisme topologique . . . 27
2.3.1 SLAM, localisation et fermeture de boucle . . . 28
2.3.2 Suivi de trajectoire . . . 31
2.4 Cas particulier des robots humanoïdes . . . 33
2.4.1 Localisation et navigation sur le robot NAO . . . 33
2.4.2 Autres robots humanoïdes . . . 36
2.5 Conclusions sur l'état de l'art et approche adoptée . . . 38

3 Structure topologique et méthodes pour la localisation et navigation 41
3.1 Description de la structure topologique . . . 41
3.1.1 Formalisme utilisé : le graphe . . . 41

3.1.2 Unité de base : le nœud . . . 42
3.1.3 Transitions . . . 48
3.2 Localisation du robot par rapport à un nœud . . . 48
3.2.1 Approche locale, partielle et rapide : la boussole visuelle . . . 49
3.2.2 Approche globale plus précise : la corrélation . . . 59
3.2.3 Raffinement du positionnement : les données 3D . . . 64
3.2.4 Renforcement par la structure en cascade . . . 69
3.2.5 Identification du nœud courant . . . 74
3.2.6 Conclusion . . . 76
3.3 Transitions d'un nœud à l'autre . . . 77
3.3.1 Contrôle de trajectoire . . . 78
3.3.2 Mesure de l'éloignement par la corrélation . . . 83
3.4 Conclusion . . . 86

4 Résultats expérimentaux et validation **89**
4.1 Dispositifs expérimentaux . . . 89
4.1.1 Utilisation du simulateur Webots . . . 89
4.1.2 Acquisition d'images en conditions réalistes . . . 90
4.2 Localisation au sein d'un nœud . . . 92
4.2.1 Estimation de l'orientation par la boussole visuelle . . . 92
4.2.2 Estimation de l'orientation par la corrélation . . . 97
4.2.3 Estimation de la position avec l'ICP . . . 99
4.2.4 Structure en cascade . . . 102
4.2.5 Identification du nœud courant par sac de mots . . . 107
4.3 Transitions . . . 115
4.3.1 Correction de trajectoire . . . 115
4.3.2 Estimation de l'orientation et du facteur d'échelle par la corrélation . . . 118
4.4 Applications et intégration dans la suite logicielle . . . 119
4.4.1 Orientation . . . 119
4.4.2 Retour à la maison . . . 121
4.5 Conclusion . . . 121

5 Conclusion et perspectives **123**

A Points d'intérêt et descripteurs **127**
A.1 Détecteurs . . . 127
A.2 Descripteurs . . . 129
A.2.1 Construction du descripteur . . . 129
A.2.2 Comparaison de descripteurs . . . 130

B Exemples de données expérimentales **131**

Bibliographie **135**

CHAPITRE 1

Introduction : contexte et plateformes

Ce chapitre a pour but de présenter le contexte général de la thèse. La partie 1.1 introduit la robotique de service, avec les enjeux économiques et les perspectives du domaine, ainsi que la société Aldebaran et le laboratoire CAOR dans lesquels la thèse s'est déroulée. La partie 1.2 présente plus en détail les robots d'Aldebaran utilisés comme plateformes.

1.1 Contexte et cas d'utilisations

1.1.1 Le contexte de la robotique de service

Le domaine de la robotique en général est un sujet extrêmement vaste et en pleine expansion. La robotique industrielle est très largement utilisée, par exemple sur des chaînes de montage, ce qui permet d'automatiser de nombreuses tâches et d'en réduire les coûts. Le marché de la robotique s'étend à d'autres applications, qui sont nombreuses et promises à une croissance importante. L'enjeu est suffisant pour que l'Union Européenne supervise des projets à long terme sur le sujet. L'association euRobotics (`http://www.eu-robotics.net/`) a pour but de définir une stratégie et une feuille de route pour la recherche et l'innovation d'ici à 2020 (projet Horizon 2020), en collaboration avec la Commission Européenne. L'Agenda de Recherche Stratégique (SRA) établi par euRobotics définit à la fois le contexte du sujet ainsi que les perspectives de marché et de recherche. On y voit notamment (voir Figure 1.1) que les projections d'ici à 2020 annoncent une croissance allant jusqu'à 25% dans le domaine de la robotique de service ; la part de marché dans ce domaine (à l'exclusion du militaire) est de 63%.

La robotique de service est un secteur particulier de la robotique centré sur l'assistance aux personnes. La robotique de service doit être distinguée de la robotique industrielle, au sens où elle est conçue pour assister un humain et non pas pour exécuter une tâche automatique de façon

purement isolée. Au sein de la robotique de service, l'IFR (International Federation for Robotics) distingue la robotique de service professionnelle et la robotique de service personnelle.

La robotique de service professionnelle concerne des robots en charge d'assister un opérateur humain dans une tâche qui peut être répétitive, dangereuse ou encore extrêmement délicate. On peut y trouver des robots agricoles (par exemple des trayeuses automatiques), des robots nettoyeurs qui atteignent des endroits difficiles d'accès, des robots médicaux (par exemple pour l'assistance dans une opération chirurgicale délicate), mais aussi des robots d'intervention dans une situation dangereuse comme le déminage ou encore l'intervention dans une centrale nucléaire. Ces robots sont en général gérés par un opérateur qualifié.

La robotique de service personnelle concerne des robots présents directement au domicile de particuliers. On y trouve des robots domestiques (de nettoyage ou de surveillance), des robots jouets, ou encore compagnons (robots éducatifs, thérapeutiques, d'assistance aux personnes), etc. Ces robots doivent donc avoir un bon degré d'autonomie, puisque, contrairement aux robots professionnels, il n'y a pas d'opérateur formé.

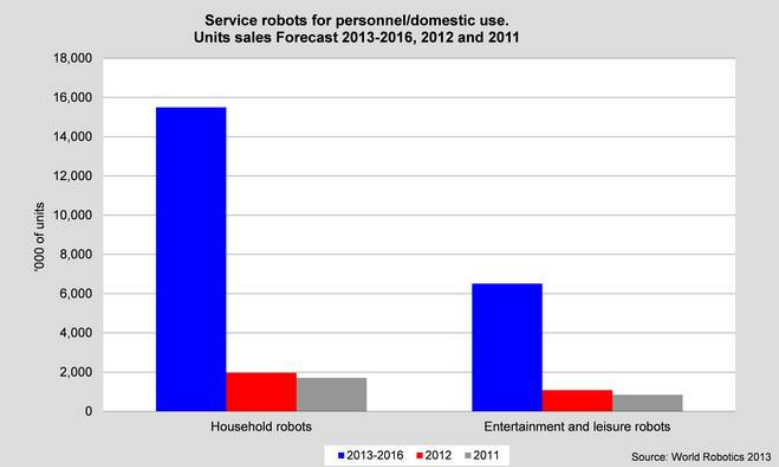


FIGURE 1.1 – Projections de vente dans la robotique de service
Source : http://www.ifr.org/service-robots/statistics/

Il s'agit d'un secteur en pleine expansion, car les applications possibles sont nombreuses et en plein développement. D'après l'IFR, 3 millions de robots de service personnels et domestiques ont été vendus en 2012 pour 1,2 milliards de dollars, soit 20% de plus qu'en 2011. Des prédictions de l'IFR pour la période 2013-2016 estiment les ventes de robots de service professionnels à 17,1 milliards de dollars, et les ventes de robots de services personnels à 5,6 milliards. On voit que même si la robotique de service personnelle représente une part plus faible du marché, elle est en plein développement et son augmentation est plus forte que pour la robotique de service professionnelle. Le secteur dédié à l'assistance aux personnes âgées est parmi ceux qui devraient connaître la croissance la plus importante, en raison du vieillissement de la population. En France, le projet France Robot Initiatives, lancé en mars 2013, vise à encourager la robotique française pour un montant estimé à 100 millions d'euros. Il s'agit donc d'un secteur particulièrement porteur, dans lequel on peut trouver des investissements importants.

Le SRA de euRobotics fait une distinction légèrement différente sur les marchés de la robotique, notamment :

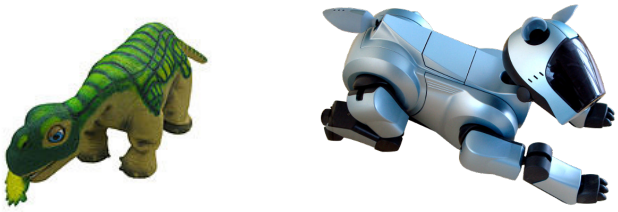
— les robots de consommation (robots de service personnels selon la classification de l'IFR). Il s'agit d'un marché de type Business to Client (B2C) ;
— les robots civils, employés par des organismes publics ou gouvernementaux. Il s'agit d'un marché Business to Government (B2G) : les applications sont par exemple la maintenance d'infrastructures lourdes ou à risque, ce qui peut donc s'assimiler à la robotique de service professionnelle ;
— les robots commerciaux. Ceux-ci sont utilisés par des entreprises pour des applications allant de l'exploitation minière à des applications marketing. Il s'agit donc d'un marché de type Business to Business (B2B) ;
— les robots de transport, chargés de transport des biens ou des personnes, dans des infrastructures publiques ou privées. Il s'agit ici d'un cas particulier de B2C, qui couvre donc notamment les véhicules autonomes ou semi-autonomes ;
— les robots militaires, qui comprennent par exemple les robots d'exploration ou les drones. Cette catégorie ne fait pas partie de celles encouragées et supervisées par le programme organisé par la Commission Européenne.

Le sujet de la thèse se déroule dans le cadre de la robotique de service personnelle dans la classification de l'IFR. Celle-ci contient elle même plusieurs champs avec leurs problématique propres.

FIGURE 1.2 – Le robot ménager Roomba de IRobot
Source : en.wikipedia.org

La robotique domestique constitue une grande part de la robotique de service personnelle. Celles-ci contient notamment les robots ménagers : les robots de la société IRobot en sont un exemple emblématique. Ces robots ménagers doivent donc effectuer de façon autonome une tâche précise. Par exemple, le Roomba de IRobot (voir Figure 1.2) est équipé d'un détecteur de choc et d'un détecteur infra rouge pour éviter les obstacles : il peut ainsi parcourir l'intégralité d'une pièce pour la nettoyer. Il est également capable de se recharger de façon autonome. Le but est donc que le robot ait un maximum d'autonomie dans un cadre restreint bien défini. Les robots ménagers sont en général très spécialisés.

Les robots jouets sont également fortement représentés. Certains sont également programmables et servent aussi de plateforme de recherche bas coût. Le robot AIBO (Artificial Intelligence roBot) fabriqué par Sony est un exemple de robot jouet particulièrement utilisé (Figure 1.3(b)). Ce robot chien est à la fois un jouet et une plateforme de recherche prisée en intel-



(a) Le robot PleoRb (b) Une version du robot AIBO

FIGURE 1.3 – Exemples de robots jouets
Source : en.wikipedia.org

ligence artificielle notamment. On peut aussi évoquer le robot Pleo Rb, un robot dinosaure qui possède une intelligence artificielle et un caractère qui s'adapte à son utilisation (Figure 1.3(a)). Les robots jouets sont en général plutôt polyvalents, centrés sur l'interaction avec l'humain : ils sont équipés de caméras, de microphones et de hauts parleurs. La principale problématique est ici d'interagir avec son entourage de façon pertinente et renouvelée, sans nécessairement requérir une autonomie de mouvement forte.

Les robots compagnons regroupent les problématiques des deux groupes précédents. Il s'agit en effet à la fois d'être autonome dans un environnement domestique mais aussi d'interagir de façon polyvalente avec l'environnement et les utilisateurs. Les robots compagnons peuvent être employés pour l'assistance à des personnes en perte d'autonomie, comme une plateforme éducative etc.

Le problème de la localisation et de la navigation autonomes du robot est crucial dans ce contexte. Il n'est en effet pas question qu'un robot compagnon nécessite d'être télécommandé par un opérateur humain, et le robot a besoin de pouvoir se déplacer d'une pièce à l'autre. Il est également nécessaire d'intégrer un aspect sémantique à la localisation, pour permettre par exemple une meilleure compréhension par un humain. Le cas de la recharge automatique du robot est un cas d'application classique, qui est par exemple traité dans le cas du robot Roomba.

1.1.2 Aldebaran

Aldebaran est une entreprise française de robotique humanoïde, fondée en 2005 par Bruno Maisonnier et spécialisée dans la robotique de service personnelle. Le but de l'entreprise est de développer des robots humanoïdes compagnons. La cible finale de l'entreprise est un robot grand public, au delà d'une plateforme de recherche pour des utilisateurs experts. Depuis 2012, la société japonaise SoftBank, grand opérateur de téléphonie mobile au Japon, a investi massivement dans la société : Aldebaran fait donc partie du groupe SoftBank.

Aldebaran a développé trois robots humanoïdes, NAO, ROMEO et Pepper (voir partie 1.2 pour les détails techniques des plateformes). NAO est à la fois une plateforme de recherche, un outil éducatif et de démonstration, qui se situe donc plutôt dans le domaine du B2B ou du B2G. ROMEO est une plateforme de recherche pour l'assistance à la personne. Pepper est un robot de grande taille, dont la vente est prévue à un prix accessible (prix d'environ 2000$ annoncé pour

le Japon en février 2015), dont la première application consiste à effectuer l'accueil et le service dans les boutiques de téléphonie mobile. Il s'agit donc d'un robot de service personnel visant à la fois le marché du B2B (robot commercial) et du B2C (robot compagnon).

FIGURE 1.4 – NAO NextGen

FIGURE 1.5 – Le robot humanoïde Pepper

NAO a été sélectionné en 2008 comme plateforme officielle de la ligue standard de la RobotCup Soccer (http://www.robocup2014.org/). Cette ligue fait concourir des équipes universitaires dans des matches de football qui opposent des équipes de 5 robots NAO. Le terrain est fixé et connu à l'avance. En 2010, un spectacle de danse de 20 NAO a été présenté au pavillon français de l'Exposition Universelle de Shanghai. En 2013 est lancé le project Autism Solution for Kids (ASK NAO, http://asknao.aldebaran-robotics.com/) : il s'agit d'un projet éducatif destiné aux enfants autistes. La version actuelle est le robot NAO NextGen. ROMEO a été présenté officiellement au salon InnoRobo à Lyon en 2014 après quatre ans de développement, et Pepper a été révélé en juin 2014 à Tokyo après deux ans de développement.

Aldebaran compte actuellement plus de 400 salariés, répartis entre la maison mère à Paris, et les bureaux de Boston, Shanghai et Tokyo. L'équipe R&D représente une grande partie de ces effectifs. Au sein de l'équipe Software, l'équipe Agility est chargée de la gestion des mouvements du robot en général, et en particulier de la localisation et de l'évitement d'obstacle. La thèse présentée ici s'est déroulée dans cette équipe, et est également rattachée au A-Lab, une structure destinée à la recherche fondamentale à Aldebaran.

1.1.3 Le CAOR

Le Centre de Robotique (CAOR, http://caor-mines-paristech.fr/) est le laboratoire de robotique des Mines ParisTech, intégré au département Mathématiques et Systèmes, et en collaboration avec l'équipe IMARA d'Inria Paris-Rocquencourt (http://www.inria.fr/equipes/imara). Les thématiques de recherche du laboratoire sont centrées autour de la robotique mobile, plus particulièrement l'assistance à la conduite automobile. Le but est d'améliorer l'autonomie et les performances de tels systèmes. Les axes principaux de recherche du laboratoire sont :

— systèmes de contrôle avancés, à savoir la modélisation et la commande de systèmes mécaniques ;
— modélisation 3D, c'est-à-dire le traitement et la modélisation de données 3D acquises par des plateformes mobiles dans un environnement urbain (projet Terra Mobilita) ou encore par des systèmes d'imagerie médicale
— **perception et apprentissage** : fusion temps réel de données multi-capteurs (caméras, laser etc), avec reconnaissance de formes (détection et classification de panneaux routiers, piétons etc) et représentation cartographique appropriée (SLAM laser etc) ;
— réalité virtuelle et augmentée, utilisée notamment pour tenter d'améliorer les interfaces homme-machine dans un contexte industriel (robot d'assemblage en collaboration avec un humain) ou de robotique mobile ;
— **robotique mobile et environnements logiciels**. On y trouve des travaux d'intégration sur des petites plateformes robotiques mobiles, mais aussi des contributions traitant de systèmes embarqués et de SLAM. Le CAOR a notamment été à l'origine du système CoreSLAM. Cet algorithme a été initialement développé pour la compétition CAROTTE, organisée par l'ANR (Agence Nationale de la Recherche) et la DGA (Direction Générale de l'Armement), qui consistait à explorer et cartographier un environnement inconnu, puis repris pour être embarqué sur les robots de Nexter Robotics, entreprise créée à la suite du succès de l'équipe.

Le sujet de la thèse se situe donc principalement dans les axes perception et robotique mobile. Il permet d'étendre les contributions à des plateformes nouvelles, puisque la plupart des travaux précédents se sont concentrés sur des robots mobiles à roues (comme le robot Nerva de Nexter Robotics) ou bien sur des véhicules automatiques ou non.

1.1.4 Circonscription du problème

La thèse porte sur le problème de la localisation et de la navigation des robots humanoïdes compagnons tels que ceux produits par Aldebaran. Ce sujet a notamment été identifié comme une des briques de base pour la réalisation d'un robot d'assistance aux personnes en situation de perte d'autonomie.

Les travaux de la thèse ont été menés dans le cadre des deux projets collaboratifs Romeo (FUI 2009-2012) et Romeo2 (PSPC 2013-2016) pilotés par Aldebaran et dans lesquels de nombreux partenaires industriels et académiques étaient impliqués. Dans le lot 6 du projet Romeo, la navigation était une partie de l'exploitation des informations des capteurs du robot. Les travaux de la thèse ont été définis en voyant les limitations industrielles et d'usage des systèmes de navigation classiquement utilisés en robotique, et testés au début du projet, à base de laser ou de balises artificielles disposés dans l'environnement. Dans le lot 5 du projet Romeo2, dédié à l'interaction physique du robot avec l'utilisateur et son environnement, les travaux de la thèse sur la localisation topologique sont associés à ceux de l'ENSTA ParisTech sur la navigation sémantique pour un enrichissement mutuel et une meilleure adaptation à l'interaction avec l'utilisateur.

Le problème est soumis à des contraintes particulièrement importantes. Tout d'abord, le système doit être autonome. Il n'est donc pas possible de déporter les calculs sur un ordinateur externe, par exemple un serveur. Tous les calculs doivent donc être effectués en embarqué, à une

fréquence raisonnable par rapport à leur utilisation (plusieurs fois par seconde pour une application en mouvement, pas plus de quelques secondes à l'arrêt). Il est éventuellement possible d'inclure de longues phases d'apprentissage, par exemple pendant la nuit.

L'environnement ne peut pas être instrumenté. Cela signifie qu'il n'est pas possible de rajouter des repères externes, passifs (marqueurs visuels) ou actifs (balises). Il n'est également pas possible de rajouter un capteur supplémentaire extérieur. Ce choix est dû au fait que la présence du robot doit être la moins invasive possible, et ne peut pas être supervisée par un utilisateur expert.

Enfin, le robot utilisé ne doit pas être modifié par rapport aux plateformes fournies par Aldebaran. Cela implique donc de se contenter des capteurs disponibles, qui ont une qualité contrainte par le bas coût du robot.

Nous verrons dans le chapitre 2 comment ces contraintes orientent les solutions possibles et constituent un cas particulièrement difficile qui n'est pas directement traité dans l'état de l'art. Nous verrons qu'il est nécessaire de développer des techniques spécifiques pour répondre aux contraintes.

L'environnement envisagé pour la localisation et la navigation correspond au cadre d'utilisation des robots d'Aldebaran. Il s'agit d'un environnement domestique comme un appartement ou plus généralement d'environnements quotidiens comme des bureaux ou une boutique. L'environnement est a donc priori dynamique. Il contient des objets ou des personnes en mouvement, susceptibles de passer devant le robot pendant une localisation. Sa structure peut également changer : il est possible d'ouvrir ou de fermer une porte, de déplacer du mobilier, de changer des posters ou des tableaux de place etc.

Le but est de fournir un système de localisation et de navigation, mais il n'est pas nécessaire que l'apprentissage de l'environnement soit entièrement autonome. Il n'est pas réaliste de demander une carte complète a priori (par exemple un plan d'architecte), mais il est possible d'imaginer un "tour du propriétaire" où l'utilisateur guide le robot et lui fournit éventuellement des indications sémantiques, comme le nom des pièces visitées.

Les exemples suivants sont des cas d'application concrets envisagés.

— Le robot se situe dans une boutique, où il effectue notamment l'accueil des clients. Il évolue dans une zone donnée, autour d'une position de base qui a été apprise au préalable. Le personnel de la boutique a indiqué au robot la direction de certains points d'intérêt, comme la caisse, la sortie ou un produit donné. Comme le robot se déplace de façon fréquente, l'odométrie est rapidement peu fiable, et conduit à des dérives fortes à la fois en position et en orientation. Grâce à la localisation, le robot doit être capable de s'orienter vers une des directions prédéfinies, par exemple pour indiquer un chemin à un client, ou bien pour se tourner vers une zone susceptible de contenir des spectateurs avant une danse. Il doit aussi être capable de revenir vers sa position initiale pour éviter de sortir de sa zone et de perturber les déplacements des clients.
— Le robot se trouve dans un appartement d'un particulier, par exemple dans le salon. Selon son emplacement, il peut effectuer des activités spécifiques : se recharger, jouer à un jeu avec l'utilisateur etc. Ces activités s'exécutent de façon précise si elles sont dans un environnement spécifique : par exemple, le robot s'asservit sur des repères lumineux pour se placer sur son chargeur. Elles ne permettent pas d'amener le robot dans ces

environnements. Le système de localisation permet donc au robot d'atteindre une zone donnée, et de permettre à une activité d'avoir des prérequis en termes d'environnement. Pour cela, il peut être amener à naviguer dans l'environnement, par exemple en passant d'une pièce à l'autre, ou en contournant une table.

On voit à travers ces deux cas que l'information de position du robot n'a pas besoin d'être précise, et qu'elle pourrait même être purement qualitative. Les activités prennent le relais avec des approches spécifiques une fois que le contexte général a été établi (asservissement visuel pour le rechargement, détection de personnes pour l'interaction avec l'utilisateur). En revanche, l'information d'orientation du robot est utile, et doit être aussi précise que possible, puisque les cas d'utilisation rendent les erreurs d'orientation visibles.

1.2 Description des plateformes disponibles

Les sections suivantes décrivent de façon détaillée les plateformes produites par Aldebaran, NAO, ROMEO et Pepper, ainsi que le simulateur Webots. On s'intéressera d'abord à l'aspect mécatronique, puis on donnera quelques éléments sur le système d'exploitation utilisé, NAO-qiOS.

1.2.1 NAO

NAO est un petit robot humanoïde, conçu pour être polyvalent et d'un prix abordable. Il est utilisé à la fois comme plateforme de recherche et d'éducation, mais aussi dans des applications médicales (dans le cadre du projet AskNAO pour l'autisme), dans des musées ou lors de spectacles de danse (par exemple dans cette vidéo).

La version principale de NAO est le H25, qui possède 25 degrés de liberté (voir Figure 1.6). Il mesure 58cm pour un peu plus de 5kg.

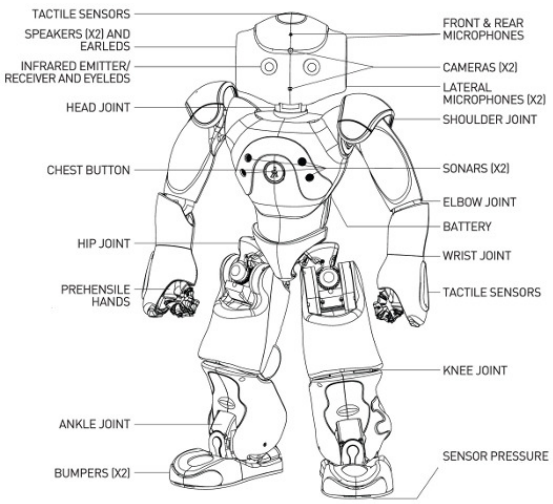
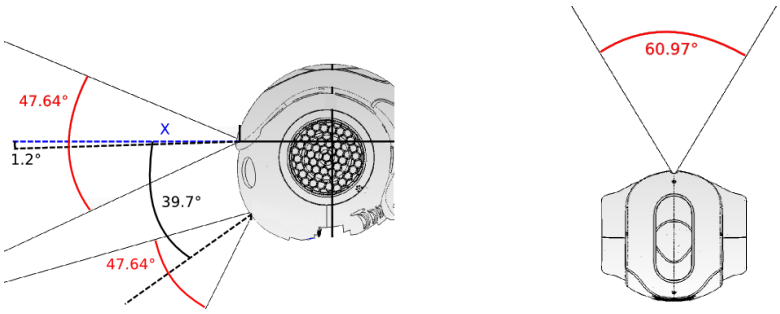


FIGURE 1.6 – Le robot NAO H25 (capteurs et degrés de liberté)

La Figure 1.6 montre l'emplacement des capteurs et (Gouaillier et al., 2009) décrit les détails de la conception mécatronique :

— pour l'audio : quatre microphones et deux haut-parleur situés respectivement sur le sommet de la tête et les oreilles ;
— pour la détection de contact : des capteurs tactiles sur les mains, des capteurs de pression sous les pieds et des détecteurs de choc au bout des pieds ;
— pour la vision : deux caméras situées dans la tête (voir aussi Figure 1.7) ;
— pour l'évitement d'obstacles : deux sonars situés dans le torse (voir aussi Figure 1.8) ;
— pour l'équilibre : une centrale inertielle située dans le torse du robot, constituée de deux gyromètres et d'un accéléromètre trois axes ;
— pour chaque degré de liberté du robot (à l'exception des mains) : un capteur de position Magnetic Rotary Encoder (MRE), d'une précision de 0.1°.

Parmi ces capteurs, les caméras, les sonars et la centrale inertielle pourraient a priori être utilisés pour la localisation.



(a) Vertical (b) Horizontal

FIGURE 1.7 – Champ de vision des caméras de NAO

Les caméras sont des capteurs cruciaux pour la navigation et la localisation. Leur champ de vue et leur positionnement est décrit dans la Figure 1.7. Chaque caméra est une Aptina MT9M114, de résolution maximale 1280x960 avec une fréquence associée de 5Hz, et de fréquence maximale 30Hz à partir de la résolution 640x480 VGA. La résolution de l'image est comprise entre QQVGA 80x60 et 4VGA 1820x960. La distance focale de la caméra est fixe, et son angle d'ouverture est de 60,9° horizontalement et 47,6° verticalement.

Le système d'acquisition de la caméra est un *rolling shutter*, c'est-à-dire que l'image est acquise ligne par ligne. Ce mode d'acquisition implique des effets de déformation verticale de l'image lorsque le robot se déplace : la vitesse de déplacement de la tête est plus importante que la vitesse d'acquisition des colonnes. Les paramètres tels que la balance des blancs, la luminosité ou le temps d'exposition sont réglables. En pratique, lorsque le robot est en déplacement, les images présentent un fort flou cinétique et une déformation due au rolling shutter, sauf lorsque le robot a les deux pieds au sol ou bien qu'il n'est pas en train de faire un pas, car la tête n'est pas stabilisée.

Il est intéressant de noter que les caméras n'ont pas un recouvrement suffisant pour permettre de les utiliser en stéréovision. Il existe une version de la tête où les caméras sont situées dans les

yeux, ce qui donne une paire de caméras avec une base de l'ordre de 10cm, mais seuls quelques exemplaires de cette tête existent.

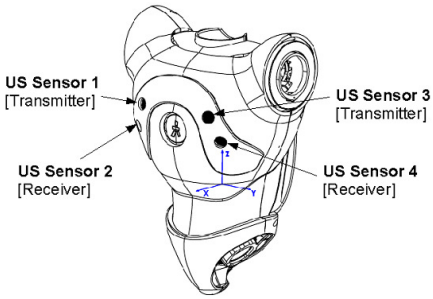


FIGURE 1.8 – Position des sonars de NAO

NAO possède un algorithme de marche stable, décrit dans (Gouaillier et al., 2010). Le contrôle est fait en boucle fermée pour l'équilibre du robot (en utilisant notamment la centrale inertielle et les capteurs MRE), mais n'a pas de retour possible sur le déplacement effectif du robot. Le contrôle est donc en boucle ouverte sur la position.

NAO possède deux émetteur-récepteur sonars, situés sur son torse (Figure 1.8). Ces sonars ont un cône de détection de 60° et un portée allant de 0.25m à 2.55m, avec une résolution théorique de 1cm. Ce très large cône de détection limite leur utilisation à la détection d'obstacles frontaux, et ne permet pas d'envisager une localisation basé sur ces capteurs.

Enfin, la centrale inertielle ne possède au total que cinq axes, pour des raisons de coût. L'axe manquant est l'axe vertical, qui aurait justement été crucial pour tenter de mesurer le cap du robot. Elle est donc uniquement utilisée pour la stabilisation du robot pendant la marche et la détection de chutes.

Le robot possède également un processeur embarqué Intel ATOM Z530 1.6GHz, possédant 1Gb de RAM, ainsi qu'un port éthernet, une connexion Wifi et un port USB.

Il apparaît donc que le système est fortement contraint : il n'y a pas de capteur métrique utilisable, la caméra a un champ de vision restreint et les images prises pendant la marche sont floues, ce qui oblige à arrêter le robot. La puissance de calcul disponible est également limitée, et il est donc plus difficile d'être réactif.

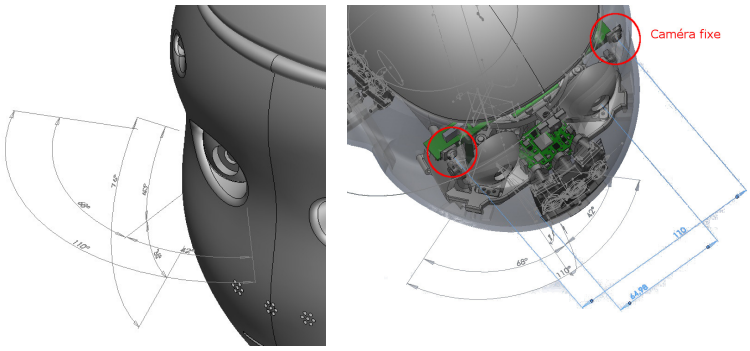
Pour obtenir une image nette de façon certaine, on doit arrêter le robot. Pour augmenter le champ de vision du robot, il faut tourner la tête. Ces deux actions ont un impact fort sur le comportement général du robot, et doivent donc être limitées au maximum.

1.2.2 ROMEO

ROMEO est un robot humanoïde, qui possède 37 degrés de liberté, pour une taille de 1.4m et un poids de 36.6kg. Il s'agit d'une plateforme de recherche, destinée à des applications d'assistance à la personne, en particulier dans le cas de personnes âgées. Le robot est en développement dans le cadre des projets Roméo et Roméo 2, en partenariat avec de nombreuses entreprises et laboratoires de recherche.

Étant donné que le robot n'est pas encore livré et n'a été complètement assemblé que récemment (pour le salon InnoRobo du 18 mars 2014), il n'a pas été possible d'effectuer des expériences directement sur le robot. Cependant, le robot est disponible dans le simulateur Webots (voir section 1.2.4), pour des expériences en simulation.

ROMEO est davantage équipé que NAO en termes de capteurs utilisables pour le problème. En particulier, il est équipé de quatre caméras (voir Figure 1.9). Deux sont fixes et peuvent être utilisées comme une paire stéréo, et deux sont situées dans les yeux et orientables. La distance entre les deux caméras fixes est de 11cm. En comptant les mouvements des yeux, le champ de vision des caméras orientables est de 110° horizontalement et de 76° verticalement.



(a) Caméras mobiles (dans le yeux) (b) Caméras fixes

FIGURE 1.9 – Positions et degrés de liberté des 4 caméras de ROMEO

ROMEO peut également être équipé d'un casque (Figure 1.10) contenant une caméra 3D Asus Xtion, qui donne une information métrique sous forme d'un nuage de points. Notons qu'il s'agit d'un modèle modifié avec uniquement une caméra 3D.

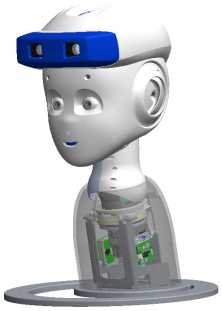


FIGURE 1.10 – Position du capteur 3D sur ROMEO

ROMEO possède au total quatre processeurs Atom, dont un dédié aux traitements de vision. On dispose donc d'une meilleure puissance de calcul que sur les autres robots d'Aldebaran, au prix d'une adaptation de l'architecture logicielle.

1.2.3 Pepper

Le robot humanoïde Pepper a été présenté par Aldebaran le 5 juin 2014, après un développement commencé décembre 2012. Pepper est resté confidentiel jusqu'à sa présentation officielle, si bien qu'il n'existe pas d'articles scientifiques le mentionnant au moment de la rédaction de cette thèse. Le robot Pepper a été développé comme un robot de service personnel. Dans un premier temps, il doit être déployé dans les boutiques de SoftBank au Japon, afin de servir pour l'accueil et le divertissement des clients. Il jouera donc le rôle d'un robot de service commercial, mais focalisé sur les interactions avec des clients non experts. Par la suite, il sera commercialisé pour le grand public, pour servir de robot compagnon.

Le robot a les caractéristiques techniques suivantes :

— 1.21 m de haut (soit presque autant que ROMEO) ;
— poids : 28kg (soit 12kg de moins que ROMEO) ;
— une autonomie de douze heures au moins en utilisation ;
— 20 degrés de liberté ;
— 3 roues omnidirectionnelles, qui en font une plateforme holonome, disposées à chaque extrémité d'une base triangulaire.

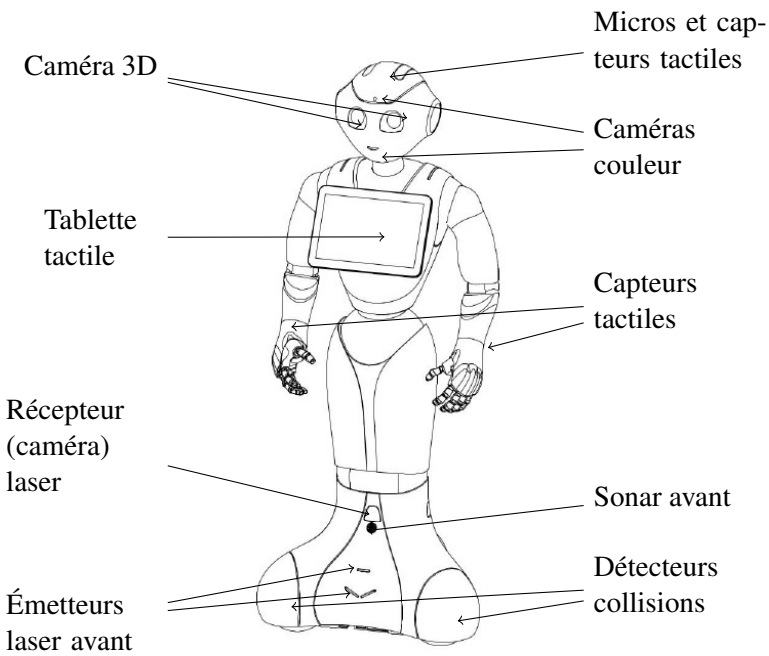


FIGURE 1.11 – Capteurs présents sur le robot Pepper

Pepper possède plus de capteurs que NAO. En particulier, il possède des capteurs métriques. La position de ces capteurs est indiquée dans la Figure 1.11. On trouve :

— des capteurs tactiles situés sur le haut de la tête et les mains ;
— des détecteurs de collisions situés dans la base par dessus les roues du robot ;

— une tablette tactile 10 pouces fixée sur son torse (la tablette n'est pas amovible), qui possède son propre processeur mais peut interagir avec le système d'exploitation NAOqi ;
— deux caméras couleur OmniVision OV5640 situées à un emplacement similaire que pour NAO, en haut et en bas de la tête du robot. Ces caméras ont une résolution et une fréquence similaires à celles de NAO ;
— une caméra 3D Asus Xtion, de résolution maximale 320x240, avec une portée théorique de 0.8m - 3.5m et un champ de vue horizontal de 60°. La caméra est située derrière les yeux de Pepper (l'émetteur derrière un oeil, le récepteur derrière l'autre) ;
— deux sonars situés à l'avant et à l'arrière ;
— des émetteurs laser combinés avec des caméras infra-rouge pour la détection d'obstacles au sol.

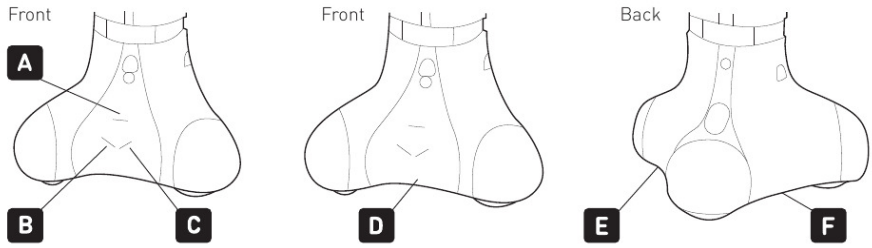


FIGURE 1.12 – Positions des émetteurs lasers dans la base du robot

Les détecteurs lasers sont conçus de la façon suivante : des émetteurs lasers infrarouge qui émettent des lignes sur l'environnement, combinés avec des caméras infrarouge, situées plus haut dans la base et inclinées pour observer le sol. La Figure 1.12 indique la position des émetteurs : trois émetteurs situés dans la partie inférieure de la base (repérés par les lettres D, E, F) qui émettent des lignes horizontales pour détecter les petits obstacles en hauteur, et trois émetteurs sur la partie avant haut (lettres A, B, C) qui émettent des lignes croisées pour détecter les trous et les pentes. Pour calculer une information de distance, on repose sur les déformations des lignes observées par la caméra : en chaque point de la ligne observée par la caméra, l'écart à la position de référence permet de déterminer la présence d'un obstacle et sa distance au capteur. Pour des raisons de débit d'information, on sous-échantillonne très fortement l'information obtenue, et on obtient ainsi une série de quinze points par laser par côté de la base. Ces informations sont actuellement utilisées pour l'évitement obstacles. Ces capteurs n'ont pas été exploités pendant cette thèse, car ils étaient en cours d'intégration : leur utilisation fait partie des perspectives d'amélioration de la thèse.

La caméra 3D est intégrée dans la tête du robot, derrière les yeux. Pour masquer l'électronique présente dans la tête et donner un meilleur aspect au robot, ces yeux sont équipés de lentilles. Ces lentilles ont une forme bombée non sphérique, ce qui pose des problèmes importants d'absorption et de déformation du signal (voir paragraphe 3.2.3.2). L'intégration dans la tête du robot permet d'orienter la caméra et d'étendre ainsi le champ de vision, de la même manière que pour la caméra couleur. En pratique, les champs de vision des caméras couleur et 3D coïncident quasiment.

Contrairement aux autres robots d'Aldebaran, Pepper n'est pas un robot bipède. Il pos-

sède trois roues omnidirectionnelles qui lui permettent d'effectuer des mouvements holonomes. Comme les autres robots d'Aldebaran, ses mouvements peuvent être contrôlés en position ou bien en vitesse. Il peut atteindre 2km/h en translation. Il possède un système de détection et de compensation des perturbations, qui lui permet de minimiser le risque de chutes en bougeant la base en conséquence. Son odométrie est nettement meilleure que celle de NAO ou ROMEO, car les glissements sont beaucoup moins perceptibles sur les roues. Elle reste tout de même assez imprécise, avec une imprécision d'environ 5cm pour 1m de déplacement, et de 2° par tour en rotation. Cette précision a été mesurée expérimentalement, mais n'est pas nécessairement définitive car les algorithmes de contrôle et la nature des roues varie encore. Dans tous les cas, l'odométrie est fortement perturbée quand les roues ne touchent plus correctement le sol, par exemple lorsque le robot doit franchir un petit obstacle, ou si quelqu'un pousse le robot. En effet l'estimation du mouvement est basée sur la mesure de la vitesse des roues, qui peuvent dans ces cas de figure tourner dans le vide.

Pepper est donc mieux équipé que NAO pour la localisation. Il possède une caméra 3D, une meilleure odométrie et une centrale inertielle 6 axes. La tête du robot est également beaucoup plus stable pendant son déplacement, puisqu'on ne retrouve pas le mouvement de balancier dû à la marche bipède. Cependant, l'odométrie a toujours une forte dérive, la caméra 3D est très coûteuse à exploiter et la centrale inertielle bruitée. De plus, les caméras couleur sont également sensibles au flou cinétique, ce qui restreint l'acquisition d'images alors que la tête est en mouvement. Le champ de vision des capteurs, limité à 60°, reste aussi le même.

1.2.4 Le simulateur Webots

Webots (`http://www.cyberbotics.com/overview`) est un simulateur développé par la société Cyberbotics. Il contient un moteur physique et permet de construire un environnement virtuel en simulant différents capteurs (éventuellement bruités), en particulier des caméras 2D ou 3D. Le simulateur contient tous les robots d'Aldebaran, et emploie notamment le même code que le robot réel. On peut donc effectuer les mêmes tests sur le robot réel et sur le robot simulé. Le simulateur est donc utilisé pour dimensionner et tester les algorithmes avant de les intégrer sur un robot réel, mais aussi pour quantifier leur précision et tester leur robustesse à long terme. Par exemple, un test longue durée peut être effectué sans danger pour l'utilisateur ou pour le robot.

Ce simulateur permet d'obtenir notamment une vérité terrain précise, via un superviseur intégré au simulateur. Le superviseur permet notamment de connaître la position exacte de n'importe quel élément du robot. En ce qui concerne le moteur physique, celui-ci est assez réaliste même s'il surestime légèrement les glissements : l'odométrie du robot simulé est donc pire que celle de la plateforme réelle.

Un environnement virtuel a été construit (Figure 1.13), et sera utilisé dans les tests en simulation présenté dans la thèse. Notons que des éléments texturés comme des posters ou des tableaux ont été rajoutés pour se rapprocher du nombre typique de points d'intérêt dans un environnement habituel.

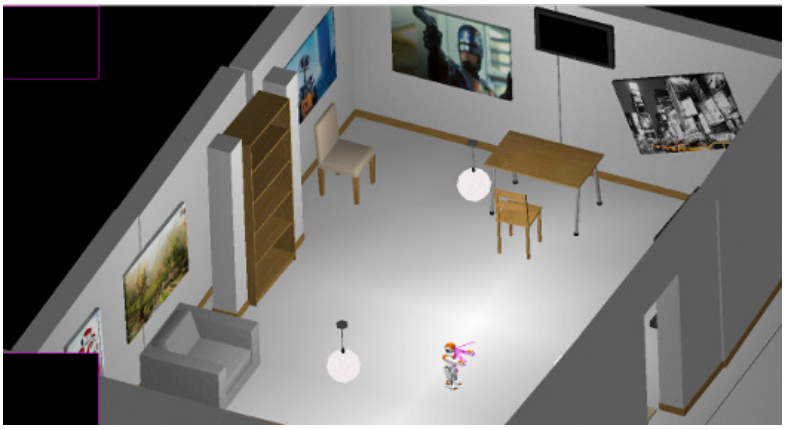


FIGURE 1.13 – Exemple d'environnement simulé Webots

1.2.5 Système d'exploitation : NAOqiOS

Tous les robots d'Aldebaran utilisent un système d'exploitation commun, NAOqiOS, développé en interne. Il s'agit d'une architecture modulaire : chaque module correspond à une fonctionnalité, et fournit une interface de programmation (API), qui peut à son tour être utilisée par les autres modules. NAOqiOS permet de s'abstraire autant que possible des robots au niveau des algorithmes clients. Les modules de bas niveau communs à tous les robots sont notamment :

— **ALVideoDevice**, qui a pour rôle de récupérer les images des différentes caméras présentes sur le robot. L'interface donne une liste des caméras disponibles et permet de récupérer les données sous un format commun. Elle gère notamment la présence de caméra(s) 3D, la résolution et la fréquence désirées ;

— **ALMotion**, qui gère l'équilibre, le déplacement de la base et des articulations. Le module fournit également des fonctions d'interpolation entre différentes positions clés, particulièrement utiles pour réaliser des animations. Pour le déplacement de la base, il est possible de spécifier une position cible dans le plan (x, y, θ), ou encore de fixer les vitesses selon les axes X et Y et la vitesse de rotation autour de l'axe Z. Il existe ensuite des adaptations particulières pour chaque robot, par exemple la fréquence de marche et la hauteur des pas pour NAO, ou bien les vitesses maximales possibles. Ces paramètres sont en général fixés indépendamment des appels aux méthodes génériques, et peuvent donc être clairement séparés du tronc commun ;

— **ALAudioDevice** qui permet de récupérer les données des microphones et d'envoyer une sortie sur les haut-parleurs.

Beaucoup de modules sont basés sur l'exploitation de ces fonctionnalités. Par exemple, ALBasicAwareness utilise à la fois les données audio, la détection de personnes (ALPeoplePerception) et la détection de mouvements (basée sur les données caméra) pour détecter et suivre les personnes de son environnement.

Les modules forment la base du système. Ils sont codés en C++ à l'aide du Software Development Kit (SDK) développé par Aldebaran. Ce SDK contient notamment la bibliothèque de traitement d'images OpenCV, très largement utilisée dans cette thèse pour les bases des algo-

rithmes.

Le système de communication entre modules permet d'abstraire presque complètement la plateforme sur laquelle le module s'exécute. On peut notamment exécuter un module sur un ordinateur séparé du robot mais situé sur le même réseau, de façon totalement transparente. Cette fonctionnalité est particulièrement utile pour la résolution de problèmes ou le prototypage et l'évaluation d'algorithmes.

Il est possible d'étendre la base des modules en rajoutant soit un nouveau module, soit une application. Une application est généralement codée en Python, à l'aide du logiciel de programmation graphique Choregraphe. Les robots d'Aldebaran embarquent un système de gestion d'applications nommé "vie autonome". Il s'agit d'un système qui vise à gérer le déclenchement approprié des activités sur le robot. Parmi ces activités, on distingue les activités interactives et les activités autonomes. Les premières sont faites en interaction avec un ou plusieurs humains : un dialogue, une prise de photo etc. Les deuxièmes sont faites lorsque le robot n'a personne avec qui interagir, et visent à donner du mouvement au robot, éventuellement à rechercher des interactions potentielles. La vie autonome gère un ensemble de conditions, qui sont calculées par différents modules du robot, et choisit une activité parmi celles qui correspondent. Par exemple, un dialogue pourra être déclenché si au moins une personne est détectée proche du robot, ou bien une animation faisant tourner la tête du robot sera déclenchée si une source de son est détectée.

Parmi les sources potentielles de déclenchement d'applications, on trouve :

— les différents détecteurs de personnes, basés sur les caméras couleur (détection de visage) et 3D (estimation de la position des personnes présentes) ;
— le moteur de reconnaissance vocale et a fortiori le moteur de dialogue. Le dialogue permet de reconnaître des phrases types à l'aide d'expressions régulières, et de déclencher ensuite d'autres activités ;
— la date et l'heure, qui permettent par exemple de déclencher des danses régulières, ou bien de déclencher un dialogue avec un contenu spécifique (selon la saison, l'heure de la journée etc).

1.3 Conclusion : un problème essentiel et des robots contraints

Les robots de service personnels, en particulier les robots compagnons, sont amenés à se développer et à viser un public de plus en plus large. Le domaine du B2C est en pleine expansion, et la stratégie de la société Aldebaran s'inscrit dans cette dynamique, notamment avec l'arrivée de Pepper et la plus grande accessibilité de NAO. Il est donc impératif de développer des solutions adaptées sur des robots à bas coût et capables d'intervenir dans des situations de la vie quotidienne.

Pour être acceptés et évoluer parmi des utilisateurs non experts, les robots doivent pouvoir être autonomes. Plus cette autonomie est grande, plus il sera facile à un utilisateur non expert d'interagir avec le robot. On peut distinguer quelques fonctionnalités essentielles à cette autonomie : détecter et suivre des utilisateurs, reconnaître des commandes vocales et engager un dialogue, assurer la sécurité du robot et des personnes en évitant les obstacles, mais aussi évoluer dans l'environnement, comme par exemple aller chercher un objet dans une pièce voisine et revenir, ou aller se recharger. Sans la capacité de se repérer et de se déplacer, le robot perd

beaucoup d'autonomie, ainsi que des occasions d'agir de façon pertinente, comme par exemple aller chercher de la nourriture dans la cuisine.

On s'intéresse ici à une variante particulière du problème de la localisation et de la navigation : les plateformes sont fixées, et ont été conçues pour pouvoir remplir des fonctions génériques et non pas spécifiquement la localisation. On s'oppose donc au cas de figure où les robots sont conçus spécifiquement pour l'exploration et la navigation, ou bien possèdent des capteurs particuliers qui fournissent une information très riche. De plus, l'environnement est a priori inconnu, et ne peut pas être modifié pour les besoins de la navigation, par contraste avec le contexte industriel, où les robots peuvent s'appuyer sur des repères dans l'environnement.

On verra dans le chapitre 2 que même si l'état de l'art dans ce domaine est particulièrement riche, il n'existe pas d'approche qui couvre tous les aspects du problème à résoudre.

CHAPITRE 2

Problématique de la localisation et de la navigation

Ce chapitre situe la thèse par rapport à l'état de l'art sur le sujet. Après une description du problème général de la localisation et de la navigation, nous présenterons les deux approches possibles (métrique et topologique) en justifiant et situant celle choisie par rapport aux travaux existants.

2.1 Description du problème général de la localisation et de la navigation

Le problème de la localisation et de la navigation autonomes est un sujet très vaste en robotique. Il peut se diviser en différents aspects :

- la construction complètement autonome d'une carte appelé Simultaneous Localization and Mapping (SLAM) ;
- la localisation pure, dans une carte définie a priori (progressive ou robot capturé) ;
- le suivi de trajectoire, avec ou sans évitement d'obstacles ;
- la détection de fermeture de boucle.

Comme mentionné dans la section 1.1.4, on se restreindra ici au cas d'un environnement non instrumenté, c'est-à-dire qu'il n'est pas possible de rajouter des marqueurs artificiels ou des dispositifs actifs (comme des balises) pour permettre au robot de se localiser. Ce genre de dispositif est fréquemment utilisé dans un contexte industriel, mais nécessite une configuration préalable lourde et n'est par exemple pas possible chez un particulier. Notre problème doit donc inclure un degré d'autonomie du robot.

Le principe du SLAM est que le robot doit être totalement autonome. Il ne possède pas d'information a priori sur son environnement, et construit incrémentalement et automatiquement une carte. Le robot doit donc se repérer par rapport aux données qu'il possède déjà et étendre ces informations au fur et à mesure de ses déplacements.

Le problème de la localisation se décompose en deux cas. Le premier est une localisation incrémentale : on localise le robot au fur et à mesure, en ayant un historique des positions précédentes. Le deuxième est le problème dit du "robot capturé" : on doit localiser le robot sans aucune information a priori. Par exemple, ce cas se pose lorsque le robot redémarre dans un environnement qu'il a appris, ou bien qu'il a subi un déplacement sans avoir de moyen de le mesurer. A priori, le deuxième cas est plus compliqué que le premier, puisqu'on ne dispose pas d'un historique des états précédents du robot. La localisation peut aussi être rattachée au problème de la fermeture de boucle, qui cherche à compenser les dérives de la localisation.

Le suivi de trajectoire porte soit sur les actuateurs du robot, soit sur la trajectoire du robot lui même. Il n'est pas forcément nécessaire ici d'avoir une information de localisation le long de la trajectoire.

La détection de fermeture de boucle est un problème qui se pose lorsque le parcours du robot contient des cycles, mais que par exemple la dérive de la localisation ne permet pas de le détecter de façon fiable. Le but est donc d'identifier que le robot est revenu à un endroit qu'il a déjà parcouru, alors que les données acquises ne sont pas nécessairement identiques. Le problème est compliqué par la présence éventuelle d'ambiguïtés qui peuvent rendre certains lieux identiques du point de vue de certains capteurs. Par exemple, deux pièces peuvent avoir le même profil vu par des capteurs laser.

Les approches sont très nombreuses pour répondre à ces problèmes, en fonction des capteurs disponibles, des contraintes de temps réel, des conditions de l'environnement (intérieur/extérieur, statique/dynamique) etc.

On peut toutefois distinguer deux approches fondamentalement différentes : l'approche métrique et l'approche topologique. La première s'attache à fournir une localisation métrique explicite du robot (une position plus ou moins complète) dans un repère absolu, alors que la deuxième cherche à localiser le robot qualitativement. Il existe également des approches hybrides qui combinent avantageusement ces deux formalismes.

Dans la suite :

— le terme "état" désignera la localisation complète du robot, à savoir l'ensemble des données de positions disponibles. Il s'agit des coordonnées du robot dans le repère considérées, plus éventuellement les informations des positions des différentes articulations du robot. Dans le cas d'un repérage du robot en coordonnées cartésiennes, il s'agit des coordonnées (x, y, θ) ;
— le terme "position" désigne les coordonnées du robot dans un repère donné, sans tenir compte de l'orientation du robot. Dans le cas d'un repèrage du robot dans un plan, cela correspond aux coordonnées (x, y) ;
— les termes "orientation" ou "cap" désignent plus particulièrement la position angulaire du robot par rapport à l'axe vertical.

2.2 Le formalisme métrique

Historiquement, l'approche métrique a été la première a être envisagée, d'abord à l'aide de capteurs qui fournissent directement une information métrique (sonars, laser, caméras 3D etc) puis de façon indirecte, à l'aide d'une ou plusieurs caméras.

Le but du formalisme métrique est de fournir une localisation métrique absolue. Cette localisation peut être plus ou moins poussée : on peut par exemple se contenter de la position du robot dans un plan et de son orientation (x, y, θ), ou bien au contraire donner l'état du robot dans 6 dimensions (la position et les rotations autour des 3 axes). Par exemple, dans le cas d'un robot guide dans un musée, la première approche est tout à fait suffisante, mais ce n'est pas le cas pour un drone volant.

Les approches métriques du problème peuvent être basées sur des capteurs eux-mêmes métriques ou bien utiliser une information indirecte. Nous évoquerons brièvement la première catégorie dans la section 2.2.1 : en raison de l'absence de capteurs métriques sur NAO et des particularités de celui de Pepper, il n'est pas possible d'appliquer directement ces méthodes, mais certains algorithmes peuvent être repris et adaptés dans des cas indirects. La section 2.2.2 décrit comment il est possible d'utiliser des caméras dans un formalisme métrique.

2.2.1 Capteurs métriques : lasers, caméras 3D, sonars

(Thrun, 2002) est un rapport très complet qui donne un aperçu des techniques principales de SLAM basées sur des capteurs métriques. L'auteur distingue plusieurs approches possibles : par filtre de Kalman, par des algorithmes de type *Expectation-Maximization* (EM), ou par grilles d'occupation. Le Tableau 2.1, tiré du même article, compare les principales approches de localisation, cartographie et navigation métrique. On y détaille les caractéristiques de chaque famille pour plusieurs aspects critiques :

— la méthode est elle incrémentale (c'est-à-dire qu'elle peut être construite en même temps que l'exploration) ?
— y a-t-il besoin de connaître les positions successives du robot ? Autrement dit, s'agit-il d'un problème de SLAM ou de localisation ?
— la méthode permet-elle de résoudre des problèmes de fermeture de boucle (correspondance) ?
— supporte-on un environnement dynamique ?

Les grilles d'occupation constituent historiquement la première approche de cartographie métrique. L'environnement est représenté par une grille, échantillonnée, qui contient dans chaque case la probabilité d'occupation de la grille. Ces probabilités sont mises à jour au fur et à mesure du déplacement du robot à l'aide de modèles probabilistes d'observation des capteurs. Il s'agit donc d'une représentation simple qui permet d'effectuer la cartographie de l'environnement, mais cela nécessite de connaître les états successifs du robot.

La première famille d'algorithmes est basée sur des filtres de Kalman. Le but en général est de construire une carte tout en estimant l'état du robot dans cette carte : on évalue la position de différents amers à partir de mesures de distance (venant par exemple d'un laser). On représente ici, à l'aide de Gaussiennes, la probabilité d'un état donné du robot et de la carte courante sachant l'ensemble des observations au cours du temps. La complexité de ces approches dépend

Caractéristiques	Kalman	EM	Grilles d'occupation
Représentation	Position d'amers	Obstacles ponctuels	Grilles d'occupation
Incrémental	oui	non	oui
Demande état du robot	non	non	oui
Correspondance	non	oui	oui
Environnement dynamique	partiellement	non	partiellement

Tableau 2.1 – Tableau comparatif de différents algorithmes de cartographie métrique
Source : Tiré et adapté de (Thrun, 2002)

de façon quadratique du nombre d'amers observés. La représentation par filtre à particules est une approximation courante des filtres de Kalman, qui permet de rendre les calculs plus légers lorsque le nombre d'amers augmente. Le principe est de simuler en parallèle un nombre donné d'hypothèses (les particules), et de les faire évoluer d'un pas à l'autre en suivant un modèle donné (par exemple un modèle d'odométrie). Une fois ces nouvelles hypothèses émises, on trie les particules par pertinence en évaluant leur qualité à partir des observations courantes. On peut éventuellement ré-échantillonner les particules en fonction de ce score.

Un autre type d'approche repose sur des méthodes de type Expectation-Maximization (EM). La méthode est constituée de deux étapes. Pendant l'étape E, on calcule pour une carte donnée la probabilité des états successifs du robot. Pendant l'étape M, on calcule la carte la plus vraisemblable étant données ces états. On obtient ainsi itérativement des cartes de plus en plus précises. Ces méthodes ont l'avantage de gérer de façon appropriée les fermetures de boucle, mais ne peuvent pas être effectuées en ligne pendant l'exploration et sont également très consommatrices en temps de calcul.

2.2.2 Capteurs non métriques : caméras

Selon les plateformes, il n'est pas toujours garanti d'avoir des informations métriques directement exploitables. En particulier, les capteurs comme les lasers sont coûteux et consommateurs en énergie. Les caméras classiques ont l'avantage d'être peu coûteuses et très facile à intégrer sur une plateforme. Cette section décrit les différentes possibilités de localisation métrique utilisant des caméras.

La paire stéréoscopique constitue quasiment un capteur métrique. Ce capteur est constitué d'une paire de caméras classiques, situées à une distance fixe et connue, dont le champ de vision se recouvre. La portée des estimations dépend de l'écartement des caméras. Ce capteur est partiellement métrique, puisqu'il fournit une information de position des points, mais donne aussi des informations visuelles. Notons que comme le montre la Figure 1.7, la position des deux caméras de NAO ne permet pas de construire une paire stéréo, mais ROMEO en possède une (cf

Figure 1.9(b)).

Les points d'intérêt constituent une des représentations les plus utilisées en vision dans le cadre du problème considéré. Nous mentionnerons ici le type de points d'intérêt employé pour chaque méthode, en particulier les points d'intérêt SIFT et SURF. Pour une présentation plus détaillée des points d'intérêt et une comparaison de leurs caractéristiques, se référer à l'Annexe A.

2.2.2.1 SLAM visuel

Il est possible de réaliser un SLAM métrique à partir d'une ou plusieurs caméras. Il convient de distinguer les algorithmes utilisant une paire stéréoscopique de ceux qui utilisent uniquement une caméra monoculaire.

Les méthodes basées sur une paire stéréoscopique possèdent une information très proche des méthodes décrites en section 2.2.1. (Konolige and Agrawal, 2008) décrit par exemple l'algorithme du FrameSLAM, qui exploite l'information d'une paire stéréo. FrameSLAM utilise un algorithme d'odométrie visuelle basée sur des appariements de points 3D détectés par la paire stéréo : il construit incrémentalement un graphe (pour réduire la dimension du problème), tout en fermant les boucles pour compenser la dérive de l'odométrie visuelle. En somme, il s'agit d'une adaptation des algorithmes les plus classiques avec des capteurs métriques, mais sur des données visuelles. (Konolige and Bowman, 2009) traite d'une extension possible à des environnements dynamiques en introduisant un système de réparation et d'agrandissement de la carte, tout en éliminant les données obsolètes. Cette méthode a été testée sur des trajets longs en extérieur (10 km) pour le premier article et en intérieur avec des intervalles de plusieurs jours pour le deuxième.

Parmi les approches basées sur une paire stéréoscopique et l'observation d'amers, on peut aussi citer l'approche par Smoothing and Mapping (SAM), qui consiste à construire la carte en estimant conjointement l'ensemble des positions du robots et des amers suivis (aussi appelé *bundle adjustement*). Pour cela, (Dellaert and Kaess, 2006) représente l'ensemble des positions et des observations sous la forme d'un graphe des facteurs (*factor graph*) qui intègre l'odométrie et les observations des amers. Il élimine ensuite les variables redondantes pour simplifier ensuite l'optimisation qui permet de calculer les différentes positions, rendant ainsi l'approche possible à grande échelle. Cette approche est par exemple testée dans (Beall et al., 2011), sur une base de données volumineuse pour la cartographie de fonds marins : on y voit que cette approche est valable sur un volume important de données et plus performante que d'autres approches classiques. (Kaess et al., 2012) est une version incrémentale du SAM, nommée iSAM2, elle aussi basée sur un graphe de facteurs sous-jacent, mais qui lui associe également un réseau bayésien. Ce réseau peut être mis à jour par exemple dans le cas d'une fermeture de boucle, et permet de calculer la modification minimale du graphe de facteurs à reporter en conséquence.

L'enjeu principal des méthodes basées sur une caméra monoculaire est d'obtenir des informations métriques complètes à partir des informations de l'image, qui ne donnent par construction que des projections. L'approche la plus classique consiste à trianguler des repères visuels au cours d'observations successives. Il existe également des approches directes qui utilisent des informations denses.

Une approche classique basée sur l'appariement d'amers visuels consiste à résoudre le problème de correspondance à n points (Perspective n-Point, PNP). Le principe consiste à déduire

l'état du robot à partir d'un ensemble d'observations d'amers visuels. L'observation correspond à la projection du point sur un plan, ici le plan image. La résolution du problème varie selon le nombre de points considérés et les hypothèses qu'ils respectent (par exemple le fait qu'ils soient coplanaires, ou alignés). Une grande partie des travaux présentés dans la suite utilisent une variante de ce problème.

(Goncalves et al., 2005) et (Karlsson et al., 2005) décrivent le vSLAM, basé sur l'information de vision et sur l'odométrie. Goncalves et al. présentent une interface qui estime l'emplacement de repères visuels en les appariant sur trois états successifs du robot. Les points sont détectés et appariés en utilisant les points d'intérêt SIFT (décrits dans (Lowe, 1999)). Pour calculer ces positions, l'algorithme utilise l'appariement de trois images successives prises à 20cm d'intervalle. Karlsson et al. raffinent ensuite les positions des points 3D à l'aide de filtres de Kalman, puis localisent le robot en utilisant l'information de l'odométrie et un filtre à particules. Pour que l'estimation de position soit fiable, la caméra doit avoir un grand champ de vision (supérieur à 60°), car les points observés dans la direction du mouvement ont une forte incertitude en profondeur.

Une autre approche utilise uniquement une caméra monoculaire. Il s'agit du MonoSLAM de (Davison et al., 2007). Le MonoSLAM consiste à suivre des repères visuels naturels, au sens où ils ne correspondent pas à des marqueurs connus ajoutés à l'environnement. La position 3D de ces repères visuels est estimée à partir d'une initialisation manuelle de points connus, puis d'observations successives du même point. Leur incertitude au fur et à mesure des observations est gérée par un modèle probabiliste bayésien. Cet algorithme a été testé sur le robot HRP-2, ainsi que sur des caméras bas coût tenues à la main. Il est important de noter que cette méthode est extrêmement proche de notre cas d'application. Cependant, comme expliqué dans la section 1.2.1 , au vu de la petite taille du robot NAO, du flou cinétique important et de son champ de vision réduit, il est quasiment impossible de suivre des repères visuels de façon fiable. On ne peut donc pas assurer le suivi et la mise à jour régulière des points de repère comme le requiert cette méthode. De plus, les premiers repères visuels doivent être initialisés manuellement si le seul capteur disponible est une caméra monoculaire.

Un autre type d'approche basé sur une caméra monoculaire est décrit dans (Courbon et al., 2008). Contrairement à la méthode du MonoSLAM, les auteurs n'utilisent pas des repères visuels mais une approche directe basée sur l'alignement de régions de l'image. Ces régions sont sélectionnées sur la base de valeurs de gradients de l'image. Les régions aberrantes sont éliminées sur la base de critères photométriques et géométriques : les régions qui présentent une variation trop importante de ces critères sont éliminées. Les régions initiales sont appariées à l'aide de points d'intérêt. Cette approche a l'avantage d'éviter les écueils habituels des méthodes basées sur des points d'intérêt, et de se contenter d'une caméra monoculaire. Cependant, il est nécessaire de faire des approximations pour pouvoir calculer les alignements des régions en temps réel.

2.2.2.2 Localisation

Une approche très fréquente consiste à utiliser une carte construite a priori pour déduire une information de position métrique à partir de capteurs non métriques. La carte peut avoir été construite ou non par des capteurs métriques.

Un premier cas possible est une localisation métrique dans une carte qui ne contient pas d'informations métriques. Par exemple, Andreasson et al. présentent un système de localisation dans une carte construite a priori par un algorithme de SLAM externe. Cette carte contient des images panoramiques espacées de 0.4m au maximum. Lors de la localisation, le robot utilise une caméra de type fish-eye. Les auteurs apparient l'image courante avec la carte en utilisant une variante des points d'intérêt SIFT, puis localisent le robot à l'aide d'un filtre à particules. Le système est robuste aux variations d'environnement avec le temps écoulé depuis la prise de la carte, ainsi qu'à des occultations partielles.

(Murillo et al., 2007) présente un système de localisation basé sur l'estimation de position de points d'intérêt. On suppose que la carte est composée d'images panoramiques annotées et positionnées. On sélectionne les deux images panoramiques les plus proches de l'image courante, et on estime la position du robot en appariant les points d'intérêt aux deux images de référence. Les auteurs comparent les performances de plusieurs descripteurs, et concluent que les descripteurs SURF sont le meilleur compromis entre robustesse et temps de calcul, en particulier par rapport aux descripteurs SIFT, communément utilisés mais plus lourds à calculer.

Il existe également des systèmes de localisation qui exploitent une information métrique contenue dans la carte. Lategahn and Stiller utilisent une caméra monoculaire grand angle (sans être panoramique) pour se localiser dans une carte contenant des informations acquises par une caméra stéréoscopique. On trouve d'abord le point le plus proche dans la carte existante. On apparie ensuite les repères visuels 3D correspondant avec les points courants. La méthode consiste à minimiser la distance entre la position en pixel du repère dans l'image projetée à partir de l'estimation et la position observée. Les auteurs utilisent le formalisme du RANSAC (Fischler and Bolles, 1981) pour éliminer d'éventuels points aberrants. Les expériences réalisées montrent qu'on obtient ainsi une précision en position de l'ordre du centimètre et inférieure au degré pour l'orientation. Les auteurs utilisent cette localisation précise pour une application de réalité augmentée.

(Meilland et al., 2010) présente une approche différente pour la localisation du robot par rapport à des points clés. La mémoire est constituée de représentations sphériques de l'environnement qui combinent l'information visuelle (en niveaux de gris), et l'information de profondeur, obtenue à partir d'images stéréo. Ces sphères sont construites à des résolutions différentes pour des raisons de temps de calcul. Pendant le suivi de chemin, on utilise uniquement une caméra monoculaire. Pour localiser le robot on déforme les pixels pour les replacer sur la sphère de référence et déduire le changement de position global à partir de ces déformations. Pour réduire la dimension du problème, on ne considère pas l'ensemble des pixels de l'image sphérique. On sélectionne uniquement des points saillants, sur la base de la valeur des gradients de l'image.

2.2.3 Position du problème par rapport au formalisme métrique

En raison des contraintes des plateformes, il est nécessaire d'écarter les méthodes qui reposent sur des capteurs métriques précis ou à longue portée comme des capteurs lasers, décrites dans la section 2.2.1. Le problème doit donc être résolu à l'aide d'une ou de plusieurs caméras monoculaires.

Cependant, il apparaît que le problème se prête mal à un formalisme métrique. Le Tableau 2.2 résume les différentes méthodes présentées ici en mettant en valeurs les capteurs et

Méthode	Capteurs	Données utilisées	Préliminaire	Non applicable car...
FrameSLAM (Konolige and Agrawal, 2008)	Paire stéréo	Amers visuels 3D	non	Capteur absent
vSLAM (Goncalves et al., 2005)	Caméra mono	Points d'intérêt	non	Odométrie insuffisante
MonoSLAM (Davison et al., 2007)	Caméra mono	Points d'intérêt	Initialisation manuelle	Intervention manuelle + taille du robot
(Courbon et al., 2008)	Caméra mono	Régions de l'image	non	Puissance de calcul

Tableau 2.2 – SLAM métriques basés caméra

les données utilisées, tout en montrant pourquoi elles ne sont pas applicables ici. On peut par exemple y voir que le MonoSLAM (Davison et al., 2007), qui utilise une caméra monoculaire en temps réel, est la solution la plus proche, mais il nécessite une première initialisation manuelle, ainsi qu'un suivi régulier des points qui n'est pas possible pendant la marche de NAO. De plus, les méthodes basées sur des triangulations de repères visuels reposent en général sur l'hypothèse d'une caméra grand angle, ce qui n'est pas le cas sur la plateforme à moins de bouger la tête du robot, ce qui ne peut pas être fait à haute fréquence. Les approches directes comme celles de (Courbon et al., 2008) sont trop demandeuses en temps de calcul et nécessitent également un champ de vision large.

Pour ce qui est de la localisation pure, les méthodes existantes utilisent nécessairement une information métrique, même si elles utilisent uniquement une caméra monoculaire pour la localisation proprement dite. Le Tableau 2.3 résume ces informations. On y voit notamment qu'on a besoin d'une construction préalable qui enregistre des données avec une localisation précise, ce qui n'est pas possible dans le cadre de notre problème.

Cependant, ces méthodes mettent en valeur plusieurs aspects importants de la localisation par caméras. On peut voir notamment que le champ de vision considéré est crucial. Plus le champ de vision est grand, plus la localisation sera précise et efficace. Beaucoup utilisent une caméra panoramique ou grand angle, quitte à corriger des déformations dues à la lentille. Il est donc nécessaire de trouver des méthodes pour compenser le champ de vision restreint de la caméra.

Le suivi de points d'intérêt est une méthode récurrente. Des expériences préliminaires dans la thèse ont montré qu'il n'était pas possible de suivre des points d'intérêt de façon suffisamment fiable pour estimer leur position. Ce problème est dû notamment à la grande imprécision de l'odométrie, à la petite taille de NAO qui augmente les effets de parallaxe et à l'impossibilité de suivre des points latéraux en mouvement. Ces résultats sont décrits notamment dans (Wirbel et al., 2013a). On y voit tout de même que certaines catégories de points d'intérêt peuvent être suivis, en particulier les points d'intérêt situés à l'infini droit devant le robot. Ces points sont

Méthode	Capteurs	Données utilisées	Préliminaire	Non applicable car...
(Andreasson et al., 2005), (Murillo et al., 2007)	Caméra mono fish-eye	Points d'intérêt	Position des images clés	Champ de vision, données position image
(Meilland et al., 2010)	Caméra mono grand angle + paire stéréo	Points saillants images mono et profondeur	Position des images clés	Capteur absent, données position image
(Lategahn and Stiller, 2014)	Caméra mono grand angle	Repères visuels 3D	Construction carte avec paire stéréo	Capteur absent pour la construction préalable

Tableau 2.3 – Localisation métrique basée caméra

parmi les moins informatifs pour les approches décrites précédemment, mais nous verrons qu'il est tout de même possible d'en tirer une information partielle et de l'exploiter pour la localisation.

On conclut donc, en raison du manque d'informations et des contraintes importantes du système, qu'une approche métrique directe est impossible dans notre cas, même en exploitant des caméras comme le font les algorithmes présentés dans la section 2.2.2. Pour cette raison, il faut choisir un autre formalisme, plus adapté au problème, qui supporte de n'avoir que des informations partielles ou qualitatives.

2.3 Le formalisme topologique

Le problème de la localisation et de la navigation peut être également traité par une approche topologique. L'information obtenue est alors une information qualitative et non plus une position absolue. Ce formalisme est souvent représenté par un graphe : les nœuds du graphe représentent des emplacements possibles du robot, et les transitions le passage d'un nœud à un autre. Le problème de la fermeture de boucle est particulièrement important dans ce formalisme, puisqu'il permet de savoir si le robot est revenu sur ses pas lors de la construction de la carte, ou pendant la navigation dans le graphe. Ce formalisme est plus souple que le formalisme métrique et plus adapté à des informations qualitatives comme celles données par la vision. Ce formalisme semble donc particulièrement adapté pour exploiter la faible quantité d'informations disponibles.

Il existe des approches hybrides (ou topométriques) qui utilisent globalement le formalisme topologique mais fournissent également une information de position locale par rapport à un nœud. Cette approche permet notamment de traiter de façon plus efficace les données en ne considérant que celles qui sont pertinentes et non pas la totalité. Certains des travaux décrits en

section 2.2.2 utilisent par exemple cette méthode pour réduire le champ de recherche pour la localisation. Un formalisme hybride est ici pertinent, car il permet d'intégrer de façon harmonieuse les capteurs métriques supplémentaires de ROMEO et de Pepper.

2.3.1 SLAM, localisation et fermeture de boucle

Il est important de noter que dans le cas du formalisme topologique, les problèmes de la construction de la carte, de la localisation dans la carte et celui de la fermeture de boucle sont très proches. En effet, la détection d'une fermeture de boucle consiste à déterminer si deux nœuds sont identiques. Si on a une méthode détection de la fermeture d'une boucle, alors la construction de la carte topologique consiste à rajouter des nœuds au cours de l'exploration tant qu'il n'y a pas de fermeture de boucle, et d'introduire une boucle dans le graphe sinon. On traitera donc dans cette section de ces trois problèmes.

2.3.1.1 Systèmes hybrides à capteurs métriques

Citons tout d'abord quelques exemples de systèmes hybrides topologique et métrique qui possèdent des capteurs métriques mais tirent parti du formalisme topologique.

(Blanco et al., 2009) cherche à segmenter les données obtenues par une technique de SLAM métrique habituel pour atteindre un formalisme topologique. Pour cela, les états et les données associées enregistrées pendant la construction de la carte sont regroupées à l'aide d'une mesure de similarité appelée Sensed Spaced Overlap (SSO). Cette mesure de similarité est implémentée pour différents types de données (points de repères, ligne laser) et permet de partitionner le graphe des états originels de façon pertinente.

(Konolige et al., 2011) utilise le formalisme topologique pour construire une carte globale. Les nœuds sont associées à des états 2D globaux, et à une carte locale construite avec une grille d'occupation de façon classique (voir section 2.2.1). La navigation est ramenée à une recherche de chemin dans le graphe ainsi défini. Ce formalisme hybride permet de diminuer la complexité et l'impact en mémoire de la carte, puisqu'il n'est pas nécessaire de maintenir une carte globale cohérente : il suffit de considérer la carte locale pertinente et de se localiser par rapport à elle. Cela permet notamment de construire des cartes à grande échelle avec un coût réduit. De même, le problème de la navigation dans un graphe est une approche simple, en particulier si certaines des transitions disparaissent suite à un changement dans l'environnement ou à un obstacle.

(Badino et al., 2011) présente un système de localisation topométrique embarqué sur une voiture équipée d'un GPS, de deux caméras de 45° d'ouverture et d'une centrale inertielle. Un graphe est d'abord construit a priori, en enregistrant des images et en les positionnant à l'aide de leurs coordonnées GPS (qui sont utilisées uniquement pendant cette phase). La localisation est ensuite effectuée avec un filtre de Bayes discret qui tient compte de la vitesse du véhicule (estimée avec la centrale inertielle) pour évaluer la probabilité de transition d'un nœud à l'autre et la fiabilité de la comparaison des images. Les images sont comparées en utilisant des descripteurs U-SURF (Upgraded - SURF), une variante du descripteur SURF plus rapide à calculer qui est notamment sensible aux rotations. (Badino et al., 2012) présente une extension de ce travail. Ces travaux introduisent notamment un descripteur global, basé sur un descripteur SURF calculé sur l'ensemble de l'image, le Whole Image - SURF (WI-SURF). Un autre descripteur global est

construit à partir des données 3D fournies par une ligne laser fixée sur le véhicule, en calculant simplement la moyenne et l'écart-type des valeurs de profondeur données. L'ajout de la ligne laser améliore les performances du système, notamment en matière de vitesse de convergence.

Tous ces formalismes reposent fortement sur la présence d'un capteur métrique, et ne peuvent donc pas être directement utilisés pour le problème considéré.

2.3.1.2 Systèmes basés caméra

On s'intéresse ici aux systèmes qui n'utilisent que des caméras pour résoudre le problème de la localisation, comme c'est le cas dans cette thèse. Il est possible que la carte construite a priori utilise des données métriques externe, mais la localisation elle-même utilise uniquement l'information visuelle. On montrera ici que la plupart des approches sont basées sur les points d'intérêt pour se localiser, mais aussi sur des descripteurs plus globaux qui ne sont pas nécessairement ponctuels. Nous traiterons d'abord les approches par points d'intérêt, puis détaillerons quelques approches alternatives pertinentes.

L'approche par sacs de mots (*bag-of-words*) est particulièrement utilisée dans ce cadre de l'identification d'un nœud à partir de son apparence visuelle. Les sacs de mots, introduits par (Sivic and Zisserman, 2003), s'inspirent d'une méthode originellement utilisée dans le traitement de texte. Ils consistent à résumer l'information contenue dans des descripteurs par une série de mots, contenus dans un dictionnaire. On construit un dictionnaire en regroupant les descripteurs dans un nombre plus faible de descripteurs (par exemple avec un k-means). Chaque descripteur est rattaché au mot le plus proche du dictionnaire, ce qui permet donc de réduire la dimension du problème au nombre de mots du dictionnaire. Pour identifier un lieu à partir d'un ensemble de mots, Sivic and Zisserman utilisent le Term Frequency - Inverse Document Frequency (TF-IDF), qui repose à la fois sur la fréquence d'un mot dans le lieu considéré et sa fréquence dans l'ensemble des mots présents. Par exemple, un mot visuel très fréquent dans une image, mais également très fréquent dans d'autres lieux, aura un TF-IDF faible et sera donc considéré comme peu significatif. A l'inverse, un mot visuel rare dans un lieu mais absent ailleurs est très significatif. Cette mesure est communément reprise dans d'autres travaux.

(Filliat, 2007) introduit des descripteurs pour l'apprentissage et la reconnaissance de pièces. Il s'agit d'une combinaison de descripteurs SIFT classiques et d'histogrammes construits sur la teinte de l'image. Ces descripteurs sont construits en ligne au fur et à mesure de l'apprentissage. (Angeli et al., 2008) utilise une approche par sac de mots pour obtenir une localisation incrémentale, basée sur ces descripteurs. Le score de TF-IDF est utilisé comme une mesure de vraisemblance, et intégré dans un formalisme bayésien. Lorsqu'une fermeture de boucle potentielle est détectée, une étape de vérification géométrique de la position des points d'intérêt est rajoutée pour confirmer ou infirmer.

Ces travaux sont complétés dans (Angeli et al., 2009). On rajoute ici une information d'odométrie, pour donner une information de position relative entre les nœuds du graphe. Lorsqu'une fermeture de boucle est détectée, la position cumulée le long des transitions ne correspond pas exactement à la position de départ, en raison de la dérive de l'odométrie. Pour résoudre ce problème, les auteurs utilisent un algorithme de relaxation des contraintes qui adapte les positions stockées dans les transitions pour correspondre à la fermeture de boucle. Tant que l'odométrie donne des informations raisonnables, en particulier en termes d'orientation, cette étape est suf-

fisante pour obtenir une localisation cohérente et satisfaisante. (Bazeille and Filliat, 2011) poursuit ces travaux en intégrant notamment un modèle de l'odométrie plus réaliste et un meilleur algorithme de relaxation des contraintes. Il s'agit donc ici d'un formalisme topométrique : la structure est topologique, constituée de nœuds et basée sur la détection de fermeture de boucle, mais les transitions contiennent des informations de positions métriques.

Le TF-IDF n'est pas la seule approche possible pour estimer la vraisemblance d'une fermeture de boucle. (Cummins and Newman, 2008) décrit le SLAM topologique Fast Appearance Based Mapping (FAB-MAP), basé uniquement sur l'apparence visuelle, et qui utilise également une approche par sac de mots. On calcule des mots visuels basés sur les descripteurs SURF des points présents dans les nœuds du graphe. Chaque nœud est caractérisé par une distribution sur les mots visuels, ce qui permet de calculer une fonction de densité de probabilité d'appartenance à un nœud. Cette distribution n'est pas évaluée à l'aide du TF-IDF, mais utilise une approximation à l'aide d'arbres de Chow-Liu (Chow and Liu, 1968). Cette approximation permet notamment de mettre en valeur les co-occurrences de mots visuels dans un lieu donné, par exemple des mots visuels correspondant à différentes parties d'une fenêtre. Le vocabulaire de mots visuels et l'arbre sont construits a priori dans une phase d'apprentissage hors-ligne. Cette méthode permet de traiter efficacement des cas d'ambiguïté visuelle, par exemple lorsque le lieu contient des mots très communs, comme les briques d'un mur. Cette méthode a été testée en extérieur avec des résultats très satisfaisants, notamment une absence de fausse détection de fermeture de boucle, et donc une précision de 100%. Le taux de rappel (taux de fermetures de boucle effectivement détectées par rapport aux fermetures effectivement existantes) est lui aux alentours de 45%.

Les travaux décrits précédemment utilisent les mots visuels détectés sans tenir compte de leur répartition dans le lieu. Certains descripteurs cherchent à capturer cette information de façon efficace. C'est le cas par exemple de (Chapoulie et al., 2011), qui travaille sur des images sphériques. On utilise ici un système de sac de mots basé sur des descripteurs SIFT, tout en rajoutant une description de la répartition des mots dans la sphère. Pour cela, la sphère est discrétisée en tranches horizontales, et on construit un histogramme qui quantifie le nombre de mots visuels dans chaque tranche de la sphère. Cette description permet alors d'introduire une variation du système décrit par exemple dans (Angeli et al., 2009), en remplaçant le TF-IDF par le Term Similarity - Inverse Document Frequency (TS-IDS). Le TS est calculé à partir de l'histogramme décrit précédemment et compare les densités des distributions autour du mot visuel identifié. On revient ensuite au formalisme bayésien habituel, en utilisant donc le TS-IDS comme mesure de vraisemblance, pour détecter les fermetures de boucle.

En parallèle des approches par sac de mots et points d'intérêt, il existe des méthodes qui cherchent à localiser le robot en utilisant des descripteurs globaux.

(Oliva and Torralba, 2006) présente le descripteur GIST. Celui-ci repose sur l'étude de l'enveloppe spatiale de la scène, calculée à l'aide de transformées de Fourier spatiale. On classifie alors les scènes selon quatre critères obtenus à l'aide de l'étude de ces caractéristiques : l'ouverture (*openness*), l'extension (*expansion*), le caractère naturel (*naturalness*) et la rugosité (*roughness*). Ce descripteur est repris dans (Chapoulie et al., 2012), et adapté aux images sphériques. On utilise ici les transformées de Fourier du GIST pour détecter les "points de changement" et ainsi segmenter une série d'images sphériques. Des expériences conduites en environnement

intérieur montrent une segmentation pertinente. Ces travaux portent davantage sur la création de nouveaux nœuds que sur la fermeture de boucle, mais sont utiles par exemple pour ajouter des informations sémantiques au graphe.

(Courbon et al., 2008) présente un système de localisation topologique par rapport à une mémoire visuelle préexistante constituée d'images sphériques (prises par une caméra catadioptrique). Pour localiser le robot, on utilise d'abord un descripteur global pour trouver les images les plus proches dans la mémoire visuelle, puis des descripteurs locaux pour discriminer l'image la plus pertinente. Pour construire le descripteur global d'une image, on assimile les niveaux de gris de l'image à des altitudes, et on enregistre l'altitude de points de contrôle définis sur un réseau triangulaire, en approchant la surface ainsi définie par une interpolation linéaire. Les descripteurs locaux sont construits à partir du calcul de corrélation croisée normalisée (Zero Normalised Cross Correlation, ZNCC) sur des régions d'images définies autour de points de Harris (Harris and Stephens, 1988). Des expériences, réalisées sur des images sphériques acquises par un UAV, ont montré que ces descripteurs étaient plus adaptés que des descripteurs SIFT ou SURF plus classiques, notamment en termes de compromis entre temps de calcul et précision. Cette approche est également mentionnée dans (Courbon et al., 2009a), où elle vient en complément d'un système de suivi de chemin, détaillé dans la section suivante.

Le Tableau 2.4 résume les différentes manières présentées ici de détecter une fermeture de boucle dans un formalisme topologique. On y détaille la technique employée, le type de représentation de données utilisées, et on vérifie si les descripteurs utilisés tiennent compte d'une structure globale ou locale du nœud.

Méthode	Technique employée	Données	Structure globale ?
(Angeli et al., 2008), (Angeli et al., 2009)	Sac de mots (tf-idf)	SIFT (+ odométrie)	non
(Chapoulie et al., 2011)	Sac de mots (ts-ids)	SIFT + répartition spatiale	oui
FAB-MAP (Cummins and Newman, 2008)	Chow-Liu (co-occurences)	SURF	non
(Courbon et al., 2008)	Vote	Surface de contrôle + corrélation	oui
(Oliva and Torralba, 2006)	Détection points de changement	GIST	oui

Tableau 2.4 – Différentes méthodes de fermeture de boucle

2.3.2 Suivi de trajectoire

Le suivi de trajectoire est un aspect essentiel de la navigation. Dans le cas d'un formalisme topologique, on ne peut pas se contenter d'asservir la position du robot à partir d'informations

de localisation métrique, puisque ces informations ne sont pas nécessairement disponibles. On s'intéresse ici au suivi de trajectoire sans localisation explicite.

Une approche commune consiste à suivre un chemin qui a été parcouru précédemment en s'appuyant sur des images clés enregistrées a priori. Ces images constituent des points de contrôle par lesquels le robot tente de passer. Le problème est alors de s'asservir sur ces images avec des informations de position partielles tirées des images courantes.

(Diosi et al., 2007) repose sur le suivi de points d'intérêt au cours du temps pour asservir le robot sur un chemin. Pendant la phase d'apprentissage, des points d'intérêt sont suivis avec un algorithme de suivi Kanade-Lucas-Tomasi (KLT), et leur position est estimée avec l'algorithme basé sur 5 points et caméra calibrée. On construit alors une carte qui contient ces points estimés. Lors du suivi du chemin, on apparie les points d'intérêt visibles avec les points d'intérêt de la carte, et on estime la position du robot par rapport à ces points. Un contrôle proportionnel est utilisé pour la vitesse de rotation du robot, et la vitesse de translation est maintenue constante. (Diosi et al., 2011) présente une évaluation expérimentale de cette technique. Les paramètres testés ici sont la méthode de suivi des points d'intérêt, et les méthodes d'appariement des points. La robustesse de la méthode est également évaluée pour des changements d'environnement ou pour la vitesse du véhicule. Diosi et al. concluent qu'en utilisant une information de position 3D localement et en compensant les changements de contraste, le système est raisonnablement robuste pour la navigation en extérieur.

La loi de contrôle décrite dans (Diosi et al., 2007) est très simple, et repose sur une approximation de la position 3D complète des points. Il existe des approches qui reposent sur les points d'intérêt mais calculent des lois plus adaptées au robot, ou qui reposent sur une information moins complète et donc moins coûteuse à calculer.

(Courbon et al., 2009b) décrit un système de suivi de chemin visuel pour un drone volant (Unmanned Aerial Vehicle, UAV). On commence par enregistrer un chemin de référence en définissant des images clés, acquises par une caméra grand angle (120°) montée sur le drone. Pour suivre le chemin, on trouve l'image clé la plus proche en appariant des points de Harris à l'aide de ZNCC (comme décrit précédemment dans (Courbon et al., 2008)). Une loi de contrôle adaptée aux particularités du drone volant est utilisée. Elle exploite l'information de position relative de l'image courante par rapport à l'image clé, qui est calculée à un facteur d'échelle prêt à partir de la mise en correspondance de cinq points entre les images. Les points aberrants sont éliminés avec un RANSAC.

(Courbon et al., 2009a) contient à la fois un système de localisation (décrit dans la section précédente) et de suivi de chemin. Les nœuds du graphe correspondent à des images clés, et on peut donc suivre un chemin visuel en parcourant les transitions sur un chemin dans le graphe topologique. En particulier, Courbon et al. recherchent le chemin le plus court à l'aide de l'algorithme de Dijkstra. Le principe du suivi de chemin est essentiellement le même que pour (Courbon et al., 2009b), à ceci près que le système considéré ici est un robot à roues et non plus un système sous-actionné comme le drone. Ce système a été testé en extérieur sur une trajectoire de 750m, à bord d'un véhicule à roues équipé d'une caméra grand angle (185°).

(Becerra et al., 2010) présente un système de suivi de chemin, basé ici aussi sur des images panoramiques, en utilisant la géométrie épipolaire. Le principe consiste à aligner latéralement l'épipôle calculé entre l'image courante et l'image de référence, afin d'adapter le cap du robot.

Celui-ci est calculé en appariant des points de la même manière que précédemment pour (Courbon et al., 2009b) par exemple. Aligner l'épipôle avec l'image de référence revient à annuler la déviation latérale du robot par rapport au chemin. Les auteurs prennent soin ici de donner des vitesses continues en rotation et en translation, en particulier lorsque le robot passe d'une image clé à la suivante. Il est intéressant de noter que cet asservissement repose uniquement sur le calcul de l'épipôle, et non pas sur les coordonnées du robot à un facteur d'échelle près comme (Courbon et al., 2009b) et (Courbon et al., 2009a).

(Cherubini et al., 2014) combine asservissement visuel le long d'un chemin et évitement d'obstacles. Les obstacles potentiels sont détectés à l'aide d'un laser, et on estime leur vitesse à l'aide d'un filtre de Kalman. On estime ensuite les trajectoires potentielles des objets, et on examine leur intersection avec les chemins possibles du robot, appelés "tentacules". On peut alors déterminer les chemins qui impliquent une collision ou une situation dangereuse, et associer une fonction de risque correspondante pour chaque tentacule. La loi de contrôle est alors adaptée pour tenir compte de cette fonction de risque, notamment en stoppant le robot avant les collisions. Ce système a été testé avec succès en environnement urbain, en présence de piétons et d'autres véhicules.

2.4 Cas particulier des robots humanoïdes

Les robots humanoïdes constituent un cas à part dans le problème de la localisation et de la navigation. En effet, ils présentent des caractéristiques particulières qui rendent la tâche parfois plus compliquée :

— leur odométrie est généralement de moins bonne qualité car la marche bipède est soumise à des glissements importants ;
— la marche du robot implique des mouvements du corps du robot, ce qui peut provoquer du flou cinétique sur les images même à basse vitesse ;
— la forme humanoïde rajoute des contraintes pour l'intégration de capteurs, par exemple pour l'équilibre du robot.

Àl'inverse, les robots humanoïdes ont des avantages par rapport aux robots à roues. Par exemple, ils peuvent être capables de franchir des petits obstacles en les enjambant, voire de monter des escaliers. Cela nécessite cependant de gérer la navigation en trois dimensions, ce qui augmente la complexité.

2.4.1 Localisation et navigation sur le robot NAO

Le problème de la localisation et de la navigation a été traité sur le robot NAO par plusieurs travaux. Pour les détails de la plateforme dans sa configuration par défaut, se référer à la section 1.2.1.

Pour localiser le robot, certains travaux s'appuient sur une instrumentation de l'environnement : on rajoute alors des marqueurs extérieurs, voire des capteurs séparés du robot.

(George and Mazel, 2013) utilisent des datamatrix pour construire une carte de l'environnement et y localiser le robot. Les datamatrix sont constituées de code barres 2D dont les dimensions sont connues a priori. Il est possible de détecter ces marqueurs et d'estimer leur position

par rapport au robot. On construit d'abord une carte, qui agrège les différentes observations des marqueurs. La navigation est effectuée en s'appuyant sur les marqueurs de la carte qui constituent des points de contrôle. Ce système permet une navigation et une localisation rapide, et utilise uniquement une caméra monoculaire, mais nécessite la présence de marqueurs connus, visibles et fréquents.

(Delfin et al., 2012) utilise le robot NAO pour construire un modèle 3D d'un objet à partir des informations de la caméra monoculaire. Le robot se localise par rapport à des amers visuels colorés, et calcule une trajectoire idéale pour reconstruire l'objet progressivement à partir des vues successives. Le problème est donc ici que l'algorithme nécessite une instrumentation précise de l'environnement, et se limite à un espace restreint.

(Cervera et al., 2012) utilise une caméra 3D (Kinect) extérieure au robot, qui observe la pièce où le robot se déplace. Le robot est localisé et suivi à l'aide d'un algorithme disponible dans la bibliothèque PCL (Point Cloud Library), et sa trajectoire est contrôlée en vitesse. Les calculs sont également réalisés en dehors du robot et optimisés pour être réalisés en temps réel. Ici aussi, on ne respecte pas la contrainte d'un environnement instrumenté, ni les contraintes de calculs embarqués.

Une alternative à l'instrumentation de l'environnement consiste à rajouter des capteurs métriques sur NAO. Il existe une variante de la tête de NAO, équipée d'un laser, visible sur la Figure 2.1. Elle contient un capteur de type Hokuyo, d'une portée de 5 mètres environ, avec un angle d'ouverture de 240°. Notons qu'en tournant légèrement la tête du robot, on obtient des données à 360° sur un plan, mais qu'il est aussi possible de pencher la tête ou le corps du robot pour avoir un information plus importante. Cette tête laser ne fait pas partie de la plateforme standard, notamment parce que le poids supplémentaire affecte l'équilibre pendant la marche. Notons également que le type de signal et l'utilisation de cette tête est différent de ceux fournis par les émetteurs laser de sécurité de Pepper.

FIGURE 2.1 – Schéma d'une tête laser pour un robot NAO

Il est également possible de rajouter une caméra 3D Asus Xtion, qui utilise la même technologie que la Kinect, à quelques différences près. Même s'il suffit de la brancher sur le port USB de NAO pour avoir accès aux données depuis le robot, cette caméra n'est pas officiellement prise en charge et affecte également l'équilibre.

Certains travaux concernent la localisation du robot dans un environnement connu a priori, en l'occurrence une carte 3D.

(Hornung et al., 2010) présente un système de localisation qui estime la position 6D du torse du robot dans un environnement intérieur, en utilisant les données du laser. Pour cela, on utilise une carte 3D de l'environnement connu a priori. La localisation se fait à l'aide d'un filtre à particules : un modèle de l'odométrie est utilisé pour mettre à jour les particules, et leur poids est calculé en comparant la distance supposée à l'environnement aux distances effectivement données par le laser. Notons que ce système calcule également l'altitude du robot, ce qui permet de suivre sa position notamment lors de la montée d'un escalier. Les calculs sont ici déportés sur un ordinateur extérieur au robot : toutefois il est possible d'exécuter les calculs directement sur le processeur du robot. (Maier et al., 2012) reprend le même principe, mais cette fois en utilisant une caméra Xtion. La tête du robot est légèrement penchée vers l'avant, ce qui permet d'intégrer un système d'évitement d'obstacles. Une fois de plus, les calculs sont déportés sur un ordinateur externe à quatre coeurs, et ne sont plus réalisables directement en embarqué sur le robot.

La phase d'expérimentation décrite dans (Hornung et al., 2010) contient une montée d'escalier, qui est construit à une échelle raisonnable pour NAO, soit des marches de 7cm de haut. Celle-ci est détaillée dans (Osswald et al., 2011). Le système utilise l'estimation de l'état global pour suivre l'avancement dans l'escalier, mais repose sur la caméra pour franchir la marche suivante. Pour cela, on estime la position de la marche suivante en détectant les contours puis les coins de la marche dans une série d'images et en renforçant l'estimation à partir des limites théoriques estimées. Le robot vérifie finalement la présence d'une marche en cherchant à la toucher avec le détecteur de choc situé sur son pied. Ces travaux sont poursuivis et améliorés dans (Osswald et al., 2012). La détection des contours de la marche est ici intégrée directement au filtre à particules de la localisation globale, en évaluant la distance de Chamfer entre les lignes observées et les lignes théoriques pour mettre à jour la vraisemblance des particules. Ce système augmente la précision et la robustesse par rapport à l'article précédent.

(Maier et al., 2013) combine un système de localisation incrémentale, de planification de trajectoire et de franchissement d'obstacles. La localisation est réalisée en intégrant les données d'odométrie et de centrale inertielle, et en alignant des nuages de points 3D pris par la caméra Xtion à l'aide d'un Generalized-ICP (GICP). L'environnement est représenté dans une carte qui contient les élévations maximales en chaque point. Cette carte est utilisée pour détecter des collisions potentielles sur l'ensemble du corps du robot. Le chemin du robot est également planifié dans la carte, en utilisant un ensemble d'actions disponibles : douze pas de base et une série de trois pas permettant d'enjamber un petit obstacle. Ces derniers sont modulables en fonction de la hauteur détectée de l'obstacle.

La question de la navigation et de l'évitement d'obstacles est également traitée sur NAO. Certaines approches utilisent des capteurs additionnels (notamment (Maier et al., 2013) évoqué précédemment), d'autres uniquement la caméra et les sonars.

(Maier et al., 2011) présente un évitement d'obstacle basé sur la reconnaissance d'objets et l'exploitation ponctuelle de données laser. La méthode consiste à déterminer des zones franchissables de l'environnement, en utilisant des classifieurs basés sur l'information de couleur et de texture. Les données du laser sont utilisées pour détecter le sol, et les classifieurs permettent de distinguer les objets du sol lui même. Pour les entraîner, on utilise la partie de l'image de la

caméra qui correspond à un obstacle détecté par le laser. L'utilisation de ces classifieurs permet de détecter les obstacles même dans des zones que le laser ne couvre pas mais qui sont visibles par la caméra du bas (voir Figure 1.7), par exemple aux pieds du robot.

(Brooks et al., 2013) propose un système de planification de trajectoire et d'évitement d'obstacles basé uniquement sur les sonars de NAO. On utilise ici l'algorithme Game-theoretic Optimal Deformable Zone with Inertia and Local Approach (GODZILA), qui cherche à optimiser les mouvements du robot en pénalisant les mouvements vers les obstacles ou qui s'écartent de la cible visée. Cette approche permet un évitement d'obstacles simple, mais souffre des imprécisions de mesure des sonars et de la largeur de leur cône d'ouverture.

Il est important de noter que tous ces travaux reposent sur des hypothèses qui ne sont pas compatibles avec le cas d'utilisation envisagé pour notre problème. Soit les calculs sont déportés et impossibles en embarqué, soit on utilise des capteurs inexistants sur le robot en temps normal, voire des capteurs extérieurs, soit le système se limite uniquement à l'évitement d'obstacle et à la navigation. Le Tableau 2.5 résume les différents algorithmes et leurs inconvénients : capteurs supplémentaires nécessaires, caractère embarqué ou non. Il est donc nécessaire de développer un système qui tienne compte des limites de la plateforme, tout en s'inspirant des approches de navigation topologique.

Méthode	Permet :	Nécessite :	Embarqué ?
(George and Mazel, 2013)	SLAM	Caméra NAO + datamatrix	oui
(Cervera et al., 2012)	SLAM	Caméra 3D externe	non
(Maier et al., 2013)	SLAM	Caméra 3D NAO	non
(Hornung et al., 2010) / (Maier et al., 2012)	Localisation	Laser / caméra 3D	non
(Maier et al., 2011)	Évitement d'obstacles	Laser + caméra NAO	possible
(Brooks et al., 2013)	Évitement d'obstacles	Sonars NAO	possible

Tableau 2.5 – Différents algorithmes de localisation et navigation sur le robot NAO

2.4.2 Autres robots humanoïdes

Il existe différents autres robots humanoïdes qui sont utilisés comme plateforme pour la navigation et la localisation.

Le robot HRP-2 (Kaneko et al., 2004) est une plateforme très utilisée par des laboratoires de recherche. Ce robot de 1.54m de hauteur, 30 degrés de liberté pour 58kg est utilisé pour des expériences de marche, de navigation, de préhension etc. Les caractéristiques de ce robot sont assez proches de celles de ROMEO, en termes de taille et de degrés liberté, ainsi que de capteurs équipés (une paire stéréo dans la tête et une caméra monoculaire).

FIGURE 2.2 – Le robot humanoïde HRP-2

(Dune et al., 2010) s'intéresse au problème du mouvement de balancier induit par la marche bipède. Ce mouvement perturbe les algorithmes d'asservissement visuel habituels en induisant des mouvements de déviation latérale. Dune et al. annulent les effets de ces oscillations en adaptant leur loi de contrôle.

(Kwak et al., 2009) présente un système de SLAM qui exploite la paire stéréo montée sur la tête de HRP-2 (baseline 144m). Le système repose sur la construction de grilles d'occupation 3D, et sur un filtre à particules qui utilisent le recalage des grilles d'occupation. Pour économiser du temps de calcul, les particules ne sont pas mises à jour systématiquement mais seulement lorsque des critères donnés sur le nuage de particules sont atteints.

(Alcantarilla et al., 2013) utilise une carte construite a priori à l'aide de données stéréovision pour localiser le robot uniquement à l'aide d'une caméra monoculaire. La carte est construite avec une méthode de SLAM externe basé sur des données obtenues par stéréovision. Comme certains travaux décrits dans la section 2.2.2, le but est alors de calculer la localisation du robot en résolvant le problème PnP. Pour associer les données, on s'appuie sur la prédiction de la visibilité des points 3D de la carte. On choisit les points de la carte supposés être les plus visibles, et on aide l'appariement avec les points détectés dans l'image de la caméra monoculaire en projetant ces points dans l'espace de la caméra. Cette méthode, qui fonctionne en temps réel, est comparée avantageusement avec les résultats obtenus par la méthode PTAM.

FIGURE 2.3 – Le robot ASIMO (Honda)

ASIMO (Sakagami et al., 2002) (Advanced Step in Innovative Moblity) est un autre robot humanoïde notable. Ce robot mesure 1m30 pour 48kg et 26 degrés de liberté dans sa dernière

version. Il possède une paire de caméras dans sa tête et une fixée sur son torse. Cependant, sa localisation et sa navigation reposent sur des marqueurs au sol observés par la caméra du torse.

(Chestnutt et al., 2005) construit une méthode de placement de pas pour ASIMO pour réaliser de l'évitement d'obstacles. La position du robot est calculée à l'aide d'un système optique Vicon externe, et la trajectoire du robot est déterminée à l'aide d'une recherche de chemin A*. Ici aussi, on a besoin d'instrumenter l'environnement, avec des capteurs de grande qualité qui fournissent une information très complète, qu'on est loin d'avoir sur les robots d'Aldebaran.

2.5 Conclusions sur l'état de l'art et approche adoptée

Le problème de la localisation et de la navigation est un sujet extrêmement vaste, et de nombreuses approches existent pour le traiter. Le cas particulier d'une plateforme à roues disposant de capteurs métriques est très largement couvert et quasiment résolu. L'utilisation de capteurs visuels, combinée avec l'approche topologique, est également de plus en plus traitée. On note tout de même que le cas des robots humanoïdes reste parmi les plus difficiles, en raison des contraintes d'intégration des capteurs et des spécificités de la marche bipède. Le problème traité ici se classe donc parmi les plus difficiles du domaine.

Une approche métrique pure du problème ne peut pas être envisagée. En effet, les plateformes ne sont pas équipées de capteurs métriques, ce qui exclut les algorithmes les plus classiques, décrits dans la section 2.2.1. Les approches qui utilisent uniquement des caméras pour la localisation ou la navigation (section 2.2.2) ne sont pas non plus envisageables : elles requièrent au mieux une intervention manuelle, ou des données métriques de vérité terrain. De plus, une tentative d'utilisation de techniques basées sur l'estimation de position de points d'intérêt visuels a montré que la taille et le champ de vision de NAO rendent cette méthode impraticable. Il existe des approches métriques réalisées sur NAO, mais elles nécessitent d'ajouter des capteurs supplémentaires, et en général ne sont pas portables en embarqué sur le robot. Une approche métrique serait peut être davantage envisageable sur ROMEO, mais ne serait pas applicable sur NAO.

Il paraît donc naturel d'utiliser un formalisme topologique. Celui-ci est couramment utilisé pour la vision monoculaire, et permet de traiter efficacement des informations qualitatives. On se basera donc sur un graphe topologique qui enregistre des images clés. La partie 2.3 montre que beaucoup d'approches se basent sur des points d'intérêt, ce qui paraît une solution peu coûteuse et efficace. Il apparaît aussi qu'il est important de se baser sur des images au champ de vision le plus large possible, et de développer une représentation adaptée. Enfin, on peut voir qu'il est également possible de contrôler des trajectoires de façon simple en utilisant uniquement des images clés, sans avoir besoin de positions complètes pour effectuer le suivi. Ces approches basées sur la vision donnent des pistes intéressantes pour le problème considéré.

Il est important de noter que les algorithmes existants reposent sur des hypothèses qui ne sont pas satisfaites dans notre cas ou bien ne proposent pas une solution complète. Il est en général supposé que les images ont un champ de vue large, et peuvent être prises à une fréquence limitée uniquement par la rapidité de l'algorithme. Ces hypothèses ne sont plus valides dans le cas des plateformes considérées : on peut tourner la tête pour agrandir le champ de vision uniquement lorsque le robot est à l'arrêt, et les images prises pendant la marche sont floues. La question se

pose donc de savoir quelles informations peuvent malgré tout être tirées des données capteurs et dans quelle mesure elles peuvent être exploitées. Il convient donc de s'adapter aux contraintes spécifiques en s'inspirant des approches classiques de vision monoculaire.

L'approche choisie est donc topologique, et cherche à tirer le maximum d'informations des données image, tout en restant progressif et économe en termes de temps de calcul. On se basera donc sur un graphe topologique, dont les nœuds contiennent des images prises par les caméras, et les transitions reposent sur des images clés tirées de ces nœuds. On cherchera à atteindre et à exploiter une représentation sphérique, malgré les limites du champ de vision du robot.

Par rapport à l'état de l'art, la contribution principale consiste à trouver des solutions qui soient valables dans un contexte très contraint, notamment où les informations exploitables sont très réduites. Le travail de la thèse a consisté à identifier les données pertinentes, et à construire un système robuste à partir de ces informations. Cela a notamment permis :

— d'explorer un des cas les plus difficiles du domaine, un **robot humanoïde à bas coût dans un environnement libre**. La forme humanoïde contraint l'emplacement et le coût des capteurs, ce qui limite notamment le champ de vision et la qualité. L'environnement est complètement libre, voire dynamique, puisqu'il contient des utilisateurs qui évoluent ;

— de fournir **une méthode robuste d'estimation de l'orientation du robot** à l'aide d'une caméra monoculaire, à l'aide de la boussole visuelle (section 3.2.1) et d'une méthode de corrélation croisée (section 3.2.2) ;

— de fournir un **cadre générique de localisation topologique** adapté aux différents robots, et indépendant des capteurs effectivement présents. Cela permet notamment d'envisager l'extension à d'autres robots, ROMEO dans un premier temps ;

— de développer une **structure hiérarchique flexible** permettant d'alléger et de rendre les calculs plus robustes, mais aussi d'étendre le calcul à une estimation de position si le robot possède un capteur métrique (caméra 3D ou paire stéréo).

CHAPITRE 3

Structure topologique et méthodes pour la localisation et navigation

Ce chapitre détaille l'approche utilisée pour permettre au robot de se localiser et de naviguer dans une carte topologique, représentée par un graphe. La structure du graphe et sa construction sont détaillées dans la partie 3.1, puis les algorithmes employés pour la localisation par rapport à un nœud sont décrits dans la partie 3.2, et ceux pour effectuer les transitions dans la partie 3.3. On n'expose ici que les algorithmes, les résultats qualitatifs et quantitatifs sont présentés dans le chapitre 4.

3.1 Description de la structure topologique

Cette section décrit les éléments du graphe topologique : les nœuds du graphe et les transitions. Tout d'abord, le formalisme utilisé est détaillé en section 3.1.1, puis on détaille le contenu d'un nœud en section 3.1.2 et d'une transition en section 3.1.3. Ce formalisme a été choisi suite à la réflexion présentée dans le chapitre 2 : on y a établi qu'il s'agissait du plus adapté en raison des contraintes du problème et des capteurs disponibles.

3.1.1 Formalisme utilisé : le graphe

Le formalisme choisi est topologique : la représentation du monde est basée sur un graphe. Le robot est positionné par rapport à un nœud donné de ce graphe. Les nœuds ne sont pas positionnés globalement de façon absolue, puisque la représentation n'est pas métrique, mais il est possible d'avoir une information de position locale du robot par rapport à un nœud donné. Les

transitions représentent des passages possibles d'un nœud à l'autre, sans hypothèse particulière sur la nature de ce chemin. Elles sont a priori orientées, car même si le chemin d'un nœud à un autre peut être parcouru en sens inverse, il n'est pas garanti qu'il existe des informations suffisantes pour contrôler cette trajectoire. Elles contiennent l'information nécessaire pour effectuer le passage entre les nœuds qu'elles relient, comme par exemple une information de position relative.

On s'intéresse ici au problème de la localisation et de la navigation, mais pas de la construction de la carte. On suppose donc qu'il y a eu une phase d'apprentissage supervisée préalable pour construire le graphe de référence. Cette phase permet donc de construire les nœuds de référence ainsi qu'un certain nombre de transitions les reliant. Il n'y a pas d'hypothèses sur la nature du graphe, qui est simplement un graphe orienté. En particulier, le graphe n'a pas de raison d'être complet, c'est-à-dire qu'il n'existe pas de transition directe entre tous les nœuds, ou de n'avoir qu'une seule composante connexe. On peut par exemple supposer que les transitions sont uniquement celles qui ont été parcourues pendant la phase d'apprentissage. Le problème de la construction non supervisée du graphe, ou de son extension a posteriori, n'est pas traité ici et constitue une perspective d'extension aux travaux. Une fois le graphe de référence construit, la question est donc de savoir quelles informations minimales sont nécessaires pour offrir une localisation et une navigation satisfaisantes.

En ce qui concerne la localisation, l'information minimale attendue est le nœud dans lequel le robot se trouve, ou à défaut le plus proche si ces nœuds ne couvrent pas la totalité de l'environnement. Cependant, une information aussi limitée n'est pas suffisante dans le cas de figure étudié. Pour un robot guide par exemple, il se pourrait que le robot tourne totalement le dos à son public, même s'il est situé au bon endroit pour faire son discours. Il est donc crucial d'obtenir au minimum une information d'orientation du robot relativement au nœud. On recherche ici une information locale par rapport au nœud, et non par rapport à l'ensemble de la carte, ce qui n'est pas pertinent. Nous verrons par la suite que cette information est justement la plus simple à déduire des informations contenues dans un nœud. Pour cette raison, les nœuds seront construits avec le souci de contenir un maximum d'informations sur l'orientation.

Les transitions d'un nœud à un autre reposent sur le fait qu'on connaît l'orientation du robot dans le nœud de départ, et exploitent les informations du nœud source, comme décrit dans la section 3.1.3. Pour naviguer, il suffit donc d'identifier le nœud courant, puis de déterminer un chemin dans le graphe topologique (s'il existe), et enfin d'emprunter les transitions d'un nœud à l'autre.

3.1.2 Unité de base : le nœud

Le nœud est construit de façon à pouvoir obtenir des informations de localisation quelle que soit la position du robot au moment de l'acquisition de la référence et du calcul de la position relative. Ces variations dépendent de la position de la base du robot et de la posture actuelle du robot. Le nœud doit contenir des informations nécessaires pour être identifié et pour déduire des informations de positions relatives par rapport à lui. Pour cela, on associe à chacun un ensemble d'images, acquises par les caméras monoculaires et 3D : on utilisera par la suite ces images pour déduire l'orientation du robot, et si possible une information sur sa position.

La posture du robot est considérée comme connue et contrôlée : la précision des différents capteurs de position est de l'ordre de 0.1°, ce qui entraîne des erreurs négligeables par rapport à la précision de localisation attendue. Cette position est connue précisément, et on peut donc se placer dans le repère de la base du robot. À l'arrêt, il est possible de contrôler la position du robot tant que celle ci est stable. On supposera donc par la suite que lors d'une localisation, le robot est dans une pose fixe, le buste droit et la tête légèrement relevée. Lors de la marche, la position du robot est contrôlée de façon plus lâche, car la marche est adaptative pour maintenir l'équilibre du robot. Il est cependant possible de contraindre certains paramètres, comme la hauteur et la fréquence des pas, ou l'inclinaison du torse.

En ce qui concerne la position de la base, le nœud doit donc être robuste dans sa zone de validité à des changements de position et d'orientation. Le nœud est donc construit de façon à couvrir la totalité des rotations autour de l'axe Z vertical. Les autres axes ne sont pas pertinents : le robot a toujours les deux pieds au sol et sa position est connue. Un nœud doit donc couvrir un champ de vision de 360° autour de l'axe Z (les axes sont visibles notamment dans la Figure 3.2).

3.1.2.1 Construction du nœud

Comme on peut le voir dans la section 1.2.1 et la section 1.2.2, les capteurs utilisables sur les plateformes cibles, à savoir les caméras RVB et éventuellement la caméra 3D se trouvent dans la tête du robot. Pour obtenir une information panoramique, il faut donc que chaque direction soit contenue dans le champ de vision de ces capteurs.

Les caméras du robot ont un champ de vision limité (voir Figure 1.7). Il est possible d'étendre ce champ de vision en faisant bouger la tête du robot. La tête dispose de deux degrés de liberté : rotation autour de l'axe vertical Z et autour de l'axe Y.

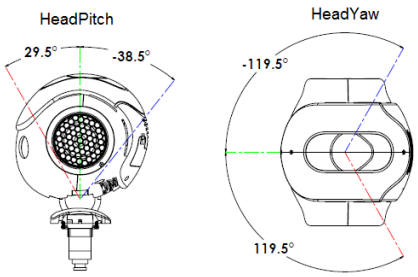


FIGURE 3.1 – Degrés de liberté et limites sur la position de la tête de NAO

Les limites de ces rotations sont indiquées dans la Figure 3.1. On voit que ces limites ne permettent pas de couvrir 360° autour de l'axe Z sans bouger la base du robot. Pour atteindre un champ de vision panoramique, il faut donc déplacer à la fois la tête et la base du robot et prendre une série d'images, au minimum 6 (si on n'effectue aucun recoupement). Notons que dans le cas du capteur laser visible sur la Figure 2.1, ce débattement sur la tête est suffisant pour couvrir la totalité du champ de vision sans bouger la base du robot.

Par souci de clarté, on définit pour chaque nœud une origine égale à l'état du robot au début de l'acquisition du nœud. Chaque image du nœud est repérée par sa position angulaire dans le

nœud autour des axes Y et Z. Comme le nœud est panoramique, la position de départ n'a en réalité pas d'influence et sert uniquement de repère commun. Ce système de positionnement est décrit dans la Figure 3.2 : on y voit la position des axes de référence et le repérage d'une image. Les paramètres visibles ici sont les rotations autour des axes Y et Z.

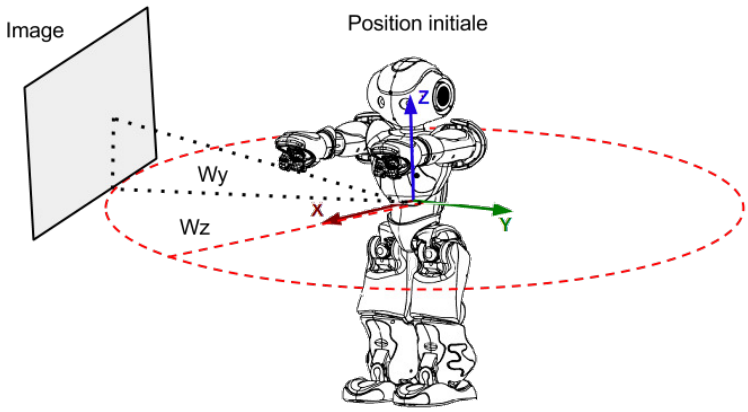


FIGURE 3.2 – Repère et positionnement des images dans un nœud

Pour positionner les images dans le nœud, la difficulté vient du fait que la mesure de la rotation de la base est dépendante de l'odométrie, qui est particulièrement imprécise. Pour connaître la position angulaire de chaque image, il y a donc deux choix possibles :

— utiliser un algorithme d'assemblage (*stitching*) pour repositionner correctement les images. Ce choix a l'avantage d'être robuste aux imprécisions du mouvement : il suffirait d'effectuer une rotation approximative, en gardant une marge suffisante pour s'assurer que tout le champ est couvert. En contrepartie, cette méthode est dépendante du succès de l'algorithme, qui en particulier nécessite des points d'intérêt nombreux, et rajoute une complexité supplémentaire à l'algorithme ;

— contrôler et / ou connaître la rotation effectuée avec suffisamment de précision. On peut alors se contenter de combiner la mesure de cette rotation avec le positionnement de la tête par rapport au corps pour connaître la position de chaque image. En pratique, c'est cette solution qui a été adoptée, car une méthode fiable de contrôle de trajectoire a été développée (voir section 3.3.1).

On supposera donc par la suite que lors de la construction d'un nœud, toutes les positions angulaires sont connues par rapport à la position initiale du robot.

Une fois les images obtenues et positionnées dans le nœud, il est nécessaire de connaître la correspondance entre les coordonnées en pixel dans une image et les coordonnées angulaires correspondantes, afin de connaître la position angulaire de chaque pixel par rapport au centre de l'image correspondante, et a fortiori par rapport au nœud. Pour cela, il faut tenir compte des paramètres extrinsèques et intrinsèques de la caméra.

Les paramètres extrinsèques de la caméra, c'est à dire sa position dans le repère de la base du robot, sont considérés comme connus et fixes. Les caméras sont fixées de façon rigide dans la tête et la position de la tête est connue, ce qui permet de remonter simplement à la position de la caméra. Les incertitudes viennent de la déformation plastique des pièces, qui sont réelles mais dont on néglige l'amplitude par rapport à la précision visée pour la localisation.

En ce qui concerne les paramètres intrinsèques de la caméra, on négligera l'effet des déformations de l'image. Le passage de coordonnées en pixels à des coordonnées angulaire se fait donc par une simple règle de proportionnalité, en utilisant le rapport entre la résolution en pixels et l'ouverture angulaire de la caméra. Cependant rien n'empêche d'injecter un modèle plus sophistiqué si les paramètres intrinsèques de la caméra sont connus (ce qui n'est pas systématiquement le cas sur les robots utilisés). Dans la suite, on considérera donc qu'il est équivalent de connaître la position d'un point en pixels dans une image pour connaître sa position angulaire par rapport au centre de l'image, et inversement.

Pour des raisons pratiques, en particulier des problèmes de flou cinétique sur les caméras RVB et de place nécessaire en mémoire, un nœud n'est constitué que d'un nombre restreint d'images, prises avec la tête à l'arrêt. Ces images sont partiellement redondantes, notamment pour augmenter la robustesse aux erreurs de positionnement et aux variations d'environnement pendant l'acquisition. Ce choix ne contraint pas la suite du problème. Toutes les méthodes décrites par la suite reposeront sur les hypothèse suivantes :

— quelle que soit la direction choisie par rapport à l'origine du nœud, il existe au moins une image qui contient une information de référence dans cette direction ;
— inversement, il est possible de connaître au moins la direction de n'importe quel point d'une image du nœud de référence.

On construit donc le nœud comme suit :

— définition de l'origine par la position initiale du robot ;
— prise d'une première série d'image en gardant la base du robot fixe et en bougeant uniquement la tête. La série consiste en 8 images consécutives, réparties entre -90° et 90° et espacées de 25°. Il y a donc un recoupement d'une image sur l'autre, pour avoir une information redondante. Pour chaque image, la tête du robot est arrêtée. On enregistre alors l'information disponible, à savoir une image couleur monoculaire et si disponible un nuage de points 3D ;
— rotation du robot avec une consigne de 180° et vérification de l'orientation finale α ;
— prise d'une deuxième série d'images identiques à la première. On positionne alors les images à partir de la position de la tête et de la rotation α connue.

Au final, chaque nœud contient donc un "panorama" constitué de 16 images successives, qui se recouvrent partiellement deux à deux. Chaque image correspond à une orientation donnée par rapport à l'origine du nœud.

La Figure 3.3 montre un exemple d'images couleur prises pendant l'acquisition d'un nœud. On remarque que malgré l'absence de recalage, les images sont positionnées de façon cohérente entre elles. On peut noter quelques imprécisions, notamment au niveau du haut des rideaux qui n'est pas tout à fait continu, mais il apparaît assez clairement qu'un algorithme d'assemblage d'images aurait un effet assez limité.

Pour réduire la complexité du calcul, on ne tient pas compte ici de la totalité du nuage de

FIGURE 3.3 – Exemple d'images prises pendant un panorama et repositionnées
Plateforme utilisée : Pepper

points 3D mais simplement d'une coupe horizontale du nuage de points. Cela permet également d'avoir un formalisme identique pour le cas particulier du NAO à tête laser (voir partie 2.4). La Figure 3.4 montre un exemple de données 3D collectées dans un nœud. L'image est graduée en mètres et la position initiale du robot est indiquée au centre. On distingue sur la figure la forme des murs adjacents, avec un bruit fort (voir paragraphe 3.2.3.2 pour davantage de détails).

Il est aussi intéressant de noter qu'on n'utilisera pas ici l'information de couleur. Les images sont stockées en RVB pour permettre une meilleure visualisation, mais l'information réellement utilisée dans la suite est le niveau de gris. L'information couleur est une donnée pertinente, mais aussi lourde à traiter et à stocker. De plus, la teinte est très sensible aux changements de luminosité, et donc peu reproductible, contrairement aux niveaux de gris qui sont plus robustes et peuvent éventuellement être normalisés.

3.1.2.2 Domaine de validité du panorama

Comme décrit dans la paragraphe 3.1.2.1, le nœud est constitué d'une série d'images. Étant donné le formalisme topologique choisi, le domaine couvert par ce nœud n'est pas fixe et connu a priori, même si on peut déduire des limites théoriques.

Pour pouvoir localiser le robot par rapport à un nœud, il faut que celui-ci soit visible. Les phénomènes d'occultation et de parallaxe sont donc limitants, mais imprévisibles a priori. Cependant, il ne suffit pas non plus d'une ligne droite dégagée entre le point courant et le centre du nœud.

Une autre limitation incontournable vient de la résolution de la caméra, car la précision des calculs est limitée (ici il s'agit d'une résolution pixelique), ce qui implique un éloignement maximum par rapport au nœud.

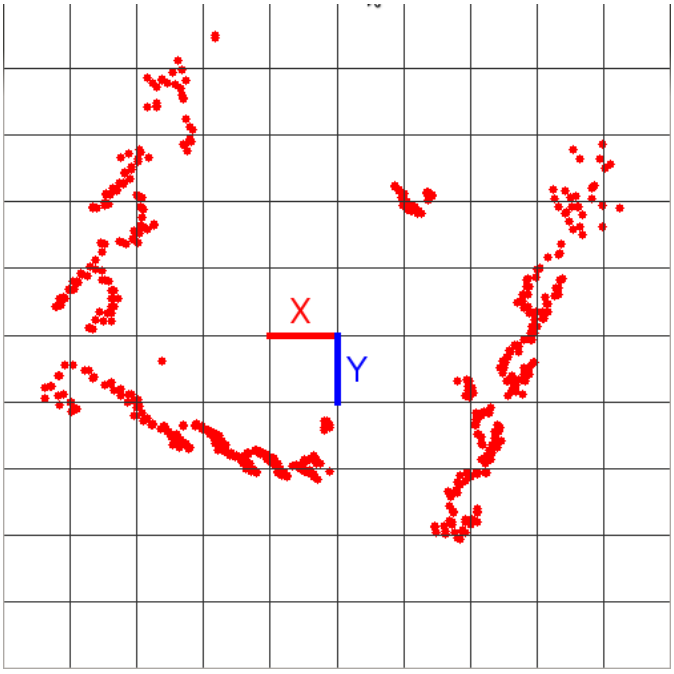


FIGURE 3.4 – Exemple de profil de panorama
Plateforme utilisée : Pepper

En plus des limites spatiales d'un panorama, il est important de se poser la question des limites temporelles. En effet, comme précisé dans la partie 1.1, l'environnement dans lequel le robot évolue est dynamique, même si il est possible de garantir qu'il reste statique pendant l'acquisition de la référence.

L'environnement peut évoluer sur une échelle de temps lente et progressive. Par exemple, dans le cas d'un appartement, le mobilier peut être déplacé progressivement. Le changement de luminosité au cours de la journée et en fonction des conditions météorologiques est également un facteur déterminant qui n'est pas caractérisé par des transitions brutales (contrairement cependant aux changements dû à l'extinction ou l'allumage de lumières artificielles). Il est donc important de se poser la question de la robustesse des méthodes employées à ces changements, et des solutions éventuelles apportées. Par exemple, une solution possible serait d'associer à chaque nœud plusieurs instances de panoramas, en fonction de l'heure de la journée ou de toute autre information pertinente.

Un autre problème potentiel vient du fait que l'environnement peut évoluer sur une échelle de temps plus courte. Par exemple, il est possible que quelqu'un soit présent devant le robot au moment où une localisation est effectuée, alors qu'il ne l'était pas pendant la prise de référence. Un autre cas de figure, plus problématique, serait qu'une personne passe devant le robot au moment de la prise d'images, si bien qu'il figurerait sur plusieurs images consécutives alors qu'en réalité il n'est présent dans aucune.

Ces problèmes proviennent du fait que la localisation n'est pas effectuée de façon continue, mais ponctuellement. En effet, comme il a été expliqué dans la paragraphe 3.1.2.1, il est néces-

saire d'effectuer des mouvements de tête au minimum. Ces mouvements sont longs et peu naturels, et ne peuvent donc pas être utilisés en permanence. Il s'agit d'une contrainte importante, puisqu'une localisation effectuée de façon continue ou plus fréquente permettrait de détecter ce genre d'anomalies.

Pour compenser ce manque, il est donc nécessaire de fournir un certain nombre de diagnostics qui permettent de détecter la perte de validité du panorama, et donc d'agir en conséquence. Les méthodes de localisation décrites par la suite (partie 3.2) sont donc conçues à la fois pour être robustes à des évolutions à court terme, et pour fournir un indicateur de confiance.

3.1.3 Transitions

Les transitions reposent sur une hypothèse fondamentale : le nœud cible est visible depuis le nœud de départ, et ce tout au long de la transition. Grâce à cette hypothèse, les transitions reposent sur l'information contenue dans le nœud de départ et le nœud d'arrivée. Cela permet d'avoir un impact moins important sur la mémoire nécessaire, mais rajoute des contraintes supplémentaires. En particulier, cela peut causer une augmentation du nombre de nœuds afin de satisfaire l'hypothèse de visibilité. Par exemple, dans le cas d'un tournant sec dans un couloir, il devient nécessaire d'avoir un nœud au niveau de l'angle, qui est visible depuis les deux branches. La Figure 3.5 illustre ce cas : les centres des nœuds sont indiqués en rouge et les transitions en noir.

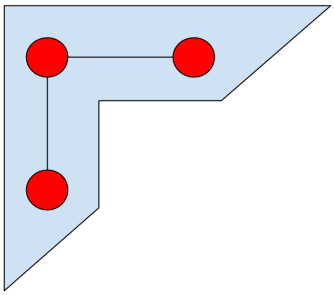


FIGURE 3.5 – Exemple de contrainte sur les transitions

Chaque transition contient l'information de cap du robot initial dans le nœud source. Il suffit alors d'aligner le robot dans la direction définie par la transition de façon contrôlée, puis d'avancer en ligne droite dans cette direction (voir section 3.3.1), tout en surveillant une mesure qui caractérise l'éloignement par rapport au panorama suivant. Cette mesure ne nécessite pas d'être métrique. En pratique, il s'agit de la méthode décrite en section 3.3.2.

3.2 Localisation du robot par rapport à un nœud

Cette section décrit la résolution du problème de la localisation du robot dans le graphe. Ce problème se divise en deux points principaux : la localisation par rapport au nœud courant (localement), et l'identification du nœud courant (globalement dans la carte).

La localisation dans un nœud répond à la problématique suivante : comment obtenir une information de localisation, même partielle, sous les contraintes particulières du problème ? Pour cela, on peut utiliser les informations contenues dans les images et profils métriques stockés dans les nœuds pour obtenir l'orientation du robot et éventuellement sa position. On verra qu'il est possible d'obtenir une orientation du robot rapidement et à faible coût, mais avec une précision limitée (voir section 3.2.1). Cette approche est ensuite complétée dans la section 3.2.2 par une méthode plus précise mais plus lourde, qui donne également une mesure de facteur d'échelle. Il est possible de raffiner encore davantage l'information de position en utilisant des données 3D, comme décrit dans la section 3.2.3, en combinant intelligemment les résultats des différentes méthodes (section 3.2.4).

L'identification du nœud courant, décrite dans la section 3.2.5, permet de résoudre le problème du robot capturé.

Dans toute la suite, les données extraites du nœud courant seront appelées *images de référence*. Les données acquises pendant le processus de localisation et comparées aux images de référence seront nommées *images de requête*.

3.2.1 Approche locale, partielle et rapide : la boussole visuelle

On décrit ici une première approche possible pour déterminer l'orientation du robot. Celle-ci est basée sur une information discrète, l'appariement de points d'intérêt, et se base sur la comparaison d'une image de référence avec une image de requête. Cette méthode, combinée avec la section 3.3.1, a fait l'objet d'un brevet (Wirbel et al., 2013b) et d'un article (Wirbel et al., 2013a).

3.2.1.1 Justification de l'approche

La partie 2.4 a montré qu'il n'était pas réaliste d'espérer suivre des amers visuels situés à des distances finies pour les trianguler. Cependant, il existe une catégorie d'amers qui est par nature plus facile à suivre : les amers situés à l'infini. Le déplacement de ces points dans le champ de vue du robot est lié au changement d'orientation du robot, mais leur apparence ne varie pas si le robot se déplace dans en translation.

En pratique, de tels amers à l'infini n'existent pas exactement. Cependant, il est possible d'approcher le comportement de beaucoup de points comme des points à l'infini. Cette approximation est valable grâce à plusieurs facteurs :

— la configuration de l'environnement par rapport à la taille des déplacements typiques du robot dans un nœud et à la taille de la zone de validité du nœud. Un nœud peut se baser sur des points d'intérêt situés sur des murs à plusieurs mètres, alors que les déplacements du robot dans un nœud sont de l'ordre du mètre ;

— le fait que lorsque le robot effectue une rotation pure, le comportement de points à l'infini est identique à ceux pour des points à distance finie. Or la plupart des mouvements du robot dans un nœud en dehors des transitions est constitué de rotations pures.

L'approche adoptée ici est la suivante : on construit une méthode qui estime l'orientation d'une image requête par rapport à une image de référence donnée. On utilise ensuite cette in-

formation, éventuellement combinée à d'autres estimations, et le positionnement de l'image de référence dans le nœud, pour en déduire l'orientation du robot

3.2.1.2 Estimation de l'orientation par rapport à une image

Cette section traite l'estimation de l'orientation entre une image de requête et une image de référence. On suppose ici que les deux images se recouvrent au moins partiellement. La recherche de l'image de requête correspondante est décrite dans la paragraphe 3.2.1.3. L'idée de l'algorithme est la suivante : on apparie des points d'intérêt de l'image de référence à l'image de requête, et on déduit le changement d'orientation de l'image de requête à partir des déplacements relatifs de l'image. La Figure 3.6 résume les différentes étapes du processus : calcul des points d'intérêt de l'image de requête K_i, appariement avec les points de référence K_i^{ref}, filtrage des appariements et calcul de l'orientation par RANSAC.

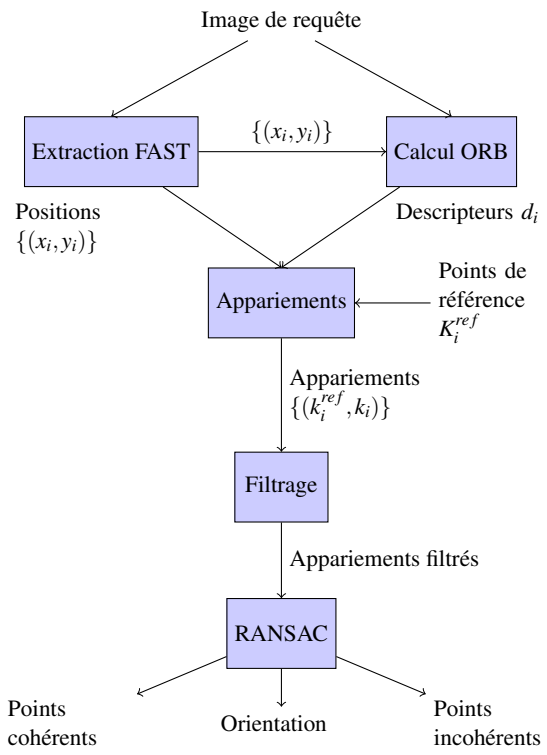


FIGURE 3.6 – Principe du calcul de l'orientation d'une image

La première étape consiste à apparier des points d'intérêt d'une image à l'autre. Pour cela, on utilise une approche classique, l'appariement des descripteurs. On commence par utiliser un extracteur pour obtenir une série de positions et d'échelles associées. Puis pour chaque point ainsi défini, on calcule un descripteur. Celui-ci se présente en général sous la forme d'un vecteur fini, sur lequel on peut donc définir une distance.

Le détecteur utilisé est un FAST (voir l'Annexe A). Il est sensible à l'échelle, et on le calcule donc sur une pyramide image afin de capturer l'information à différentes échelles. Il s'agit d'un

bon compromis entre vitesse d'extraction et robustesse, notamment en comparaison avec les détecteurs SIFT ou SURF, plus stables et robustes à l'échelle mais aussi plus lourds à calculer.

Le descripteur associé est un Oriented Binary Robust Independent Elementary Features ORB (Calonder et al., 2010) et (Rublee et al., 2011). Il s'agit d'un vecteur binaire de taille 256. Celui-ci correspond au résultat de comparaisons entre des paires de pixels répartis dans la zone du détecteur : si la valeur du premier pixel de la paire est inférieure à celle du deuxième, la valeur de la coordonnée est 0, sinon 1. Ce descripteur a l'avantage d'être :

— très rapide à calculer (il s'agit uniquement de comparer des valeurs) ;
— très rapide à comparer avec la distance de Hamming ;
— facile à stocker (puisqu'il peut être représenté sur 256 bits).

Par opposition, le descripteur SURF est :

— plus long à extraire puisqu'il implique d'estimer localement des gradients, même de façon approchée ;
— plus long à comparer puisqu'il utilise une distance euclidienne ;
— plus lourd à stocker puisqu'il s'agit d'un vecteur de 64 ou 128 flottants.

À partir de l'image de référence, on construit un index. La recherche dans cet index permet alors, pour chaque point d'intérêt de la requête, de trouver le plus proche voisin dans la référence. Cette première phase d'appariement est sujette à des fausses détections. Pour compenser, on filtre les appariements. Cela peut se faire en appliquant un seuil sur la distance maximale tolérée entre deux descripteurs appariés. Le problème dans ce cas est qu'il faut déterminer un seuil absolu qui est difficile à interpréter et varie d'un descripteur à l'autre et d'une distance à l'autre. Une autre solution, adoptée ici, consiste à considérer l'appariement inverse, c'est-à-dire le plus proche voisin de chaque point de référence dans l'image de requête. On sélectionne alors uniquement les points de la requête tels que le point apparié dans la référence est son unique antécédent. La Figure 3.7 montre un exemple d'appariements incorrects (en rouge pointillé) et corrects (en vert) entre deux images. Cette approche permet de retenir uniquement les points qui ne sont pas ambigus. En contrepartie, cette méthode nécessite d'effectuer deux fois un calcul de plus proche voisins. Quelle que soit la méthode adoptée, on supposera par la suite que les appariements contiennent un taux non nul d'appariements corrects, mais sans exclure une proportion importante d'erreur.

On travaille avec un modèle et des hypothèses très simples : d'une image à l'autre, les déplacements des points d'intérêt sont uniquement dûs aux changements d'orientation du robot, c'est-à-dire aux rotations autour des trois axes. Ce modèle est vrai dans le cas où le mouvement du robot est une combinaison de rotations autour des axes ou si les points suivis sont situés à l'infini par rapport au robot. On voit donc bien que le modèle est trop simple et n'expliquera pas les mouvements des points mal appariés ou trop proches pour être considérés à l'infini. Le problème est donc le suivant : on a un modèle simple pour expliquer les déplacements des points, et une série de données qui peuvent être bruitées ou même ne pas suivre le modèle choisi. Le formalisme du RANSAC (RANdom SAmple Consensus) est particulièrement adapté dans ce cas (Fischler and Bolles, 1981).

L'algorithme du RANSAC repose sur le fait qu'on utilise un modèle pour expliquer le comportement d'un grand nombre de points, mais qui peut se calculer rapidement sur un nombre restreint de points en supposant que ces points sont bien choisis. Si ce modèle a été calculé sur

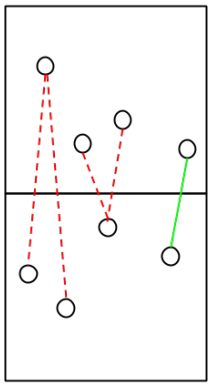


FIGURE 3.7 – Appariements corrects (vert plein) et incorrects (rouge pointillé) de points d'intérêt

des points cohérents avec d'autres, alors le nombre de points acceptés sera plus grand que si on se base sur des points non significatifs. On sélectionne donc au hasard, pour un nombre d'itérations données, un sous ensemble de points pour calculer le modèle. Pour qu'il soit cohérent, tous ces points sélectionnés doivent l'être. On itère ce tirage pour un nombre d'itérations pour avoir une probabilité donnée p qu'au moins un de ces tirages contiennent des points cohérents.

Le nombre d'itérations dépend de p et du taux de points cohérents, noté c. On note $N(c,p)$ le nombre d'itérations, et n le nombre de points nécessaires pour calculer le modèle. La probabilité que tous les points sélectionnés soient cohérents est c^n, et donc la probabilité qu'aucun des points ne soit cohérent est $(1-c^n)$. Donc la probabilité que chaque itération contienne au moins un point invalide estimation $(1-c^n)^{N(c,p)}$. Donc si on note p la probabilité souhaitée d'avoir au moins un échantillon cohérent, la probabilité que tous les échantillons contiennent au moins un point incohérent est $1-p$. Alors on a $1-p=(1-c^n)^{N(c,p)}$ et donc finalement

$$N(c,p)=\frac{log(1-p)}{log(1-c^n)}$$

Notons que le taux de points cohérents est d'abord estimé grossièrement, puis mis à jour au fur et à mesure que le meilleur modèle est calculé (voir Algorithme 1).

Le modèle du RANSAC est le suivant : on prend en entrée deux paires de points appariés entre la référence et la requête. Comme expliqué dans la paragraphe 3.1.2.1, on a une correspondance directe entre la position en pixels dans l'image et la position angulaire du point correspondant. Les calculs suivants se feront donc avec des coordonnées en pixels, puis le résultat sera converti en angulaire. On suppose ici que l'origine du repère en pixel est situé au centre de l'image, et que les axes sont cohérents avec les conventions des sens de rotation. En pratique, il faut effectuer un changement de repère car les coordonnées images sont données par rapport à un repère situé en haut à gauche de l'image.

Pour simplifier les notations, on utilisera ici les coordonnées complexes pour repérer les points dans le plan de l'image. On note X_i et Y_i les axes de l'image. On considérera ici que le point de coordonnées $u=0$ est situé au centre de l'image. On note $u_1=x_1+iy_1$ la position du

premier point dans la référence, et u'_1 la position de son appariement dans la requête. De même pour le deuxième point on utilise u_2 et u'_2. On note ω_x, ω_y et ω_z les rotations estimées de la requête par rapport aux axes X, Y et Z de la référence.

La rotation autour de l'axe X du robot correspond à une rotation autour du centre l'image. Une rotation autour de l'axe Y du robot correspond à une translation sur l'axe Y_i de l'image. Enfin, une rotation autour de l'axe Z du robot correspond à une translation sur l'axe X_i de l'image. Ces translations peuvent être obtenues à partir de la correspondance entre pixels et angle d'ouverture (voir paragraphe 3.1.2.1). On supposera ici qu'il s'agit d'un facteur de proportionnalité, noté μ. Selon les conventions choisies pour représenter les rotations, on applique d'abord la rotation autour de l'axe X, puis Y, puis Z. On a donc

$$\begin{aligned} u'_1 &= u_1 e^{i\omega_x} + \mu(\omega_y + i\omega_z) \\ u'_2 &= u_2 e^{i\omega_x} + \mu(\omega_y + i\omega_z) \end{aligned}$$

Donc

$$\begin{aligned} \frac{u'_1 - u'_2}{u_1 - u_2} &= \frac{u_1 e^{i\omega_x} - u_2 e^{i\omega_x}}{u_1 - u_2} \\ &= e^{i\omega_x} \end{aligned}$$

On note $o_1 = u'_1 - e^{i\omega_x} u_1$

On note $\Im(u)$ la partie imaginaire et $\Re(u)$ la partie réelle d'un nombre complexe u. Donc finalement on a :

$$\begin{aligned} \omega_x &= \arg\left(\frac{u'_1 - u'_2}{u_1 - u_2}\right) \\ \omega_y &\propto \Im(o_1) \\ \omega_z &\propto \Re(o_1) \end{aligned}$$

Pour estimer l'adéquation d'un point au modèle, il suffit de comparer l'image théorique d'un point à sa position réelle dans l'image. Si le modèle calculé est $(\omega_x, \omega_y, \omega_z)$, alors on note

$$\tilde{u} = e^{i\omega_x} u + \mu(\omega_y + i\omega_z)$$

et on peut utiliser comme mesure d'erreur $d = ||\tilde{u} - u||$.

Notons que ce modèle repose sur le fait que les transformations d'un point à son image sont uniquement constitués de rotation et translations. Cette hypothèse est donc loin d'être valide pour tous u_1, u'_1, u_2, u'_2. En particulier, on a un cas d'erreur si $u_1 = u_2$ ou plus généralement si $||u_1 - u_2|| \approx 0$, puisqu'on a une division par zéro. Ce modèle est également faux si on a $||u'_1 - u'_2|| \neq ||u_1 - u_2||$, c'est à dire qu'il y a également eu un facteur d'échelle appliqué entre les deux images. On pourrait vérifier ainsi un certain nombre de conditions nécessaires pour que l'estimation soit pertinente. Cependant, la démarche du RANSAC permet d'ignorer ces limites et évite d'avoir à construire une liste de vérifications. En effet, si le modèle est calculé à partir d'une paire de points invalides, alors le nombre de points cohérents sera plus faible que si on utilise une paire de points valides. Il n'est donc pas nécessaire de prendre un modèle plus complexe. Par exemple, ces problèmes pourraient être évités en prenant des triplets de points et donc obtenir les angles sans ambiguïté ou hypothèse supplémentaire. En contrepartie le nombre

d'itérations et le temps de calcul du modèle sont plus grands, si bien qu'il n'est pas avantageux d'utiliser cette technique.

On note E l'ensemble des paires de points appariés. On note M_{max} le meilleur modèle calculé par le RANSAC, et M le modèle calculé à chaque itération. On note $\mathscr{P}$ et $\mathscr{P}_{max}$ l'ensemble des paires de points cohérents avec M et M_{max} respectivement. On note N_{points} le nombre total de paires de points. On note n l'indice de l'itération courante du RANSAC. L'Algorithme 1 résume le principe de l'application de la méthode RANSAC.

Algorithme 1 Estimation de l'orientation de la requête

Paramètres : c taux de points cohérents initial, p probabilité de sélection de points cohérents

Entrée : E l'ensemble de paires de points appariés.
Sortie : $(\omega_x, \omega_y, \omega_z)$ rotations de la requête, c score (taux de points cohérents), $\mathscr{P}_{max}$ ensemble des points cohérents

On note $\mathscr{P}$ l'ensemble des paires cohérentes pour le modèle courant.
while $n < N(c,p)$ **do**
  Calculer le modèle M sur deux paires de points appariés.
  Calculer $\mathscr{P}$ paires de points cohérents.
  if $|\mathscr{P}| > |\mathscr{P}_{max}|$ **then**
    $\mathscr{P}_{max} \leftarrow \mathscr{P}$ et $M_{max} \leftarrow M$
    $c \leftarrow \dfrac{|\mathscr{P}_{max}|}{N_{points}}$
  end if
  $n \leftarrow n+1$
end while
Raffiner M_{max} à partir d'un calcul sur $\mathscr{P}_{max}$, par exemple avec une minimisation au moindre carrés.

On obtient donc un modèle final qui contient une estimation des orientations relatives sur les trois axes, ainsi qu'une classification des points cohérents ou non.

La Figure 3.8(a) montre un exemple de résultat de RANSAC sur un appariement dans un cas de rotation pure. Les paires de points appariés cohérentes sont dessinées en vert, tandis que les incohérentes sont indiquées en rouge. On voit que les paires cohérentes correspondent bien à une déviation globale cohérente, ici une rotation autour de l'axe Z. Les paires incohérentes correspondent ici à de faux appariements, sur des points probablement ambigus en termes de descripteurs.

La Figure 3.8(b) montre un exemple de résultat du RANSAC sur un cas où les hypothèses de départ sont moins vérifiées : le robot a avancé entre les deux prises d'images. Les points qui ne sont pas situés à l'infini ont donc un mouvement qui est dû au changement d'échelle et non purement à la rotation (ce qui brise l'hypothèse $||z'_1 - z'_2|| = ||z_1 - z_2||$). Ces points sont mis en valeur sur la figure par des ellipses bleues : ils sont situés sur un meuble qui est relativement près du robot (environ 1m). Ils sont considérés comme incohérents par rapport au modèle, même s'ils sont bien appariés, ce qui est le comportement attendu. Au contraire, les points qui ont un

(a) Sur une rotation pure

(b) Avec des hypothèses partiellement fausses

FIGURE 3.8 – Exemples de résultats de RANSAC
Plateforme utilisée : Webots 3.8(a), Pepper 3.8(b)

comportement plus proche de points à l'infini sont cohérents, à savoir ceux qui sont situés sur l'étagère au fond.

On utilisera dans la suite le taux de paires cohérentes comme une mesure de la qualité de l'appariement. Cette qualité est donc sans unité. La qualité maximale est de 1 (tous les points sont cohérents), et la qualité minimale est 0.

3.2.1.3 Estimation de l'orientation absolue

Le paragraphe 3.2.1.2 permet de calculer l'orientation relative entre deux images qui se recoupent partiellement. Pour obtenir une information d'orientation absolue, il faut donc d'abord extraire une image de référence pertinente et calculer ensuite l'orientation de l'image de requête par rapport à celle-ci.

On suppose dans un premier temps qu'on a une unique image de requête. La présence d'images multiples sert uniquement à rendre l'estimation plus robuste, comme décrit à la fin de cette section.

Une première approche naïve consisterait à effectuer le calcul de l'orientation entre l'image de requête et toutes les images de référence, et de sélectionner l'image pour laquelle le score de

points cohérents est le plus élevé. Cependant, cette méthode implique de faire des calculs sur des images non pertinentes de façon systématique, ce qui augmente à la fois le temps de calcul et les erreurs potentielles.

Pour remédier à ce problème, on adoptera ici une approche plus efficace et moins systématique. Cette approche repose sur la constatation suivante : puisque les images de référence se recoupent entre elles, si on compare l'image de requête à l'image de référence la plus proche, alors on doit observer un maximum local du nombre de points appariés sur cette image. Le nombre de points appariés sur les images situées de chaque côté de la référence idéale doit en effet être légèrement inférieur, mais toujours significatif, car les images se superposent légèrement avec leurs voisines.

Une condition nécessaire pour que cette hypothèse soit valide est qu'elle le soit en interne pour le panorama de référence. Pour vérifier cela, on peut calculer une matrice qui effectue des appariements deux à deux entre toutes les images de référence (cette matrice est symétrique). Idéalement, la matrice doit être nulle partout sauf sur la diagonale (où on compare les images avec elles mêmes) et sur les diagonales inférieures et supérieures (où on compare les images à leurs voisines directes). Il est donc nécessaire de vérifier que ces conditions sont vraies avant d'enregistrer un nœud, en tolérant éventuellement quelques points en dehors des diagonales si leur nombre reste négligeable (moins de 5 points par exemple).

En pratique, cette condition est très largement vérifiée. La Figure 3.9 montre un exemple d'une telle matrice calculée sur un panorama typique. Les indices des images sont indiqués en haut et à gauche de l'image, et le nombre de points d'intérêt appariés est indiqué dans chaque case. Les valeurs maximales de points appariés sont indiqués en rouge, et les valeurs minimales en bleu. On voit bien que chaque image est distinguée des autres, tout en ayant quelques points communs avec l'image précédente et l'image suivante. On peut utiliser cette matrice pour diagnostiquer un cas d'ambiguïté en détectant une valeur trop importante en dehors de la diagonale, ou bien une image qui serait trop pauvre en points d'intérêt en comparaison avec les autres. On supposera donc par la suite que le panorama satisfait l'hypothèse de non ambiguïté, avec un comportement proche de celui décrit dans la matrice présentée. Par exemple, un critère peut être : pas plus de 10 points appariés en dehors de la diagonale, et au moins 30 points sur la diagonale. La Figure 3.9 montre notamment qu'il y a une ambiguïté potentielle entre les trois premières images.

Le principe de la méthode consiste donc à sélectionner une image candidate parmi les images de référence qui est la plus proche possible de l'image de requête avant de calculer l'orientation relative. On note I_k l'image de référence d'indice k dans le panorama, et N_k le nombre de points appariés entre I_k et l'image de requête I. On note k l'indice courant de l'image de référence candidate. Les indices sont supposés être ramenés entre 0 et $P-1$, où P est le nombre d'images dans le panorama. L'Algorithme 2 décrit cette recherche : elle consiste à d'abord rechercher un maximum local d'appariements, puis à effectuer le calcul de l'orientation avec l'image correspondante. Pour cela, on part d'une hypothèse préalable, et on cherche une image contenant un taux minimum de points d'intérêt. Cette recherche se fait à partir de l'hypothèse, en s'éloignant progressivement de l'image de départ à gauche et à droite. Une fois une telle image trouvée, on recherche le maximum local le plus proche en vérifiant les images voisines. Toutes les recherches s'effectuent au maximum sur la totalité des images, en évitant de faire plusieurs tours.

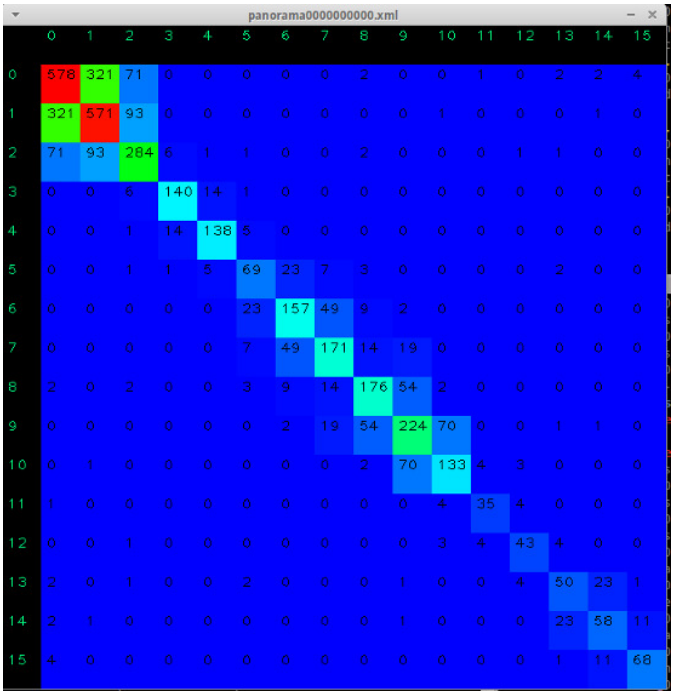


FIGURE 3.9 – Exemple de matrice de confusion d'un panorama (bleu : pas de points d'intérêt appariés, rouge : nombre maximal de points d'intérêt appariés)

On voit que cette méthode, sous une hypothèse simplificatrice, permet une recherche plus efficace que l'approche naïve. En effet, le calcul de l'orientation proprement dite n'est fait qu'une seule fois, après la sélection d'une image candidate appropriée. De plus, la recherche s'arrête dès qu'un maximum local de points appariés est détecté, ce qui évite de faire une recherche complète. Plus l'hypothèse de départ est proche, plus la recherche sera rapide. Cette recherche est aussi rapide que la recherche naïve dans le pire des cas (le cas où l'hypothèse de départ est diamétralement opposée à la position réelle de la meilleure image). En revanche, on prend le risque de sélectionner une fausse image candidate. Expérimentalement, on a montré que ces cas étaient en réalité très rares (voir section 4.2.1). Si on sélectionne une mauvaise image de référence, alors le calcul échoue ou donne une estimation fausse.

Les cas d'échec possibles proviennent de la présence d'un autre maximum local ou d'un trop faible nombre de points appariés. On a vu que le premier cas pouvait être limité par l'étude de la matrice de confusion du panorama. La section 4.2.1 montre notamment que le cas des maximums locaux ambigus se produit très peu en pratique. Le second dépend du seuil fixé pour accepter une image candidate, qui doit être réglé pour que le résultat du calcul de l'orientation soit significatif. On peut également considérer le taux de points aberrants en sortie de RANSAC comme un indicateur supplémentaire de qualité de l'appariement.

Pour renforcer la qualité de l'estimation, notamment pour compenser d'éventuelles erreurs de l'algorithme précédent, il est possible de prendre plusieurs images de requête. Dans ce cas, on utilise l'algorithme précédent pour chacune des images de requête, puis on cherche à combiner

Algorithme 2 Calcul de l'orientation de la requête par rapport à un nœud

Paramètres : n nombre minimal de paires appariées.

Entrée : I requête, $I_0, \ldots, I_P$ images de référence. k_0 hypothèse de départ pour l'indice de la meilleure image.

Sortie : I_k meilleur appariement, $(\omega_x, \omega_y, \omega_z)$ rotations correspondantes.

Initialiser $k_g = k_0$ et $k_d = k_0 + 1 \pmod{p}$ indices de recherche (gauche et droite).
Rechercher une image qui contient au moins n paires appariées.
while $N(k_g) < n$ ou $N(k_d) < n$ et $k_g \not\equiv k_d \pmod{P}$ **do**
  $k_g \leftarrow k_g - 1 \pmod{P}$ (décalage vers la gauche)
  $k_d \leftarrow k_d + 1 \pmod{P}$ (décalage vers la droite)
end while
if $k_d \equiv k_g \pmod{P}$ **then**
  Tour complet réalisé sans trouver assez de points.
  return Échec : pas assez d'appariements
end if
Soit k_i tel que $N(k_i) > n$.
Soit k l'indice potentiel de la meilleur image. On initialise $k = k_i$.
On modifie k pour que $N_{k-1 \pmod{P}} < N_k$:
while $N_{k-1 \pmod{P}} > N_k$ et $k \not\equiv k_i \pmod{P}$ **do**
  $k \leftarrow k - 1 \pmod{P}$ (décalage vers la gauche).
end while
On modifie k pour que $N_{k+1 \pmod{P}} < N_k$:
while $N_{k+1 \pmod{P}} > N_k$ et $k \not\equiv k_i \pmod{P}$ **do**
  $k \leftarrow k + 1 \pmod{P}$ (décalage vers la droite).
end while
if $k \equiv k_i \pmod{P}$ **then**
  On a fait un tour complet sans trouver de maximum local.
  return Échec : pas de maximum local.
end if
k est l'indice d'une image qui présente un maximum local de paires appariées.
Calculer l'orientation entre I_k et I à l'aide de l'Algorithme 1

les estimations. On commence par éliminer les éventuelles mesures aberrantes, en éliminant celles qui sont trop éloignées des autres. On calcule ensuite la moyenne des valeurs restantes, pondérée par le taux de points cohérents.

On obtient donc une mesure de l'angle courant de la base du robot par rapport à un nœud. Cette méthode peut être calculée en temps réel (voir la section 4.2.1 pour davantage de détails).

3.2.1.4 Précision et limites de l'approche

La précision de l'estimation du cap de robot dépend de plusieurs paramètres. Certains sont inhérents à la plateforme, d'autres dépendent des hypothèses simplificatrices.

La résolution angulaire de l'image est un premier facteur crucial. En effet, la précision minimale du déplacement relatif d'une image à l'autre correspond à l'ouverture angulaire d'un pixel. Par exemple, pour une résolution VGA sur NAO, un pixel correspond à une précision 0,095°, et pour une résolution QQVGA à 0,76°. On ne peut donc pas espérer une précision meilleure. On remarque tout de même que même pour la résolution la plus basse, la précision est largement meilleure que celle fournie par l'odométrie.

Une autre source d'imprécision est la position des points d'intérêt. Il est notamment difficile de situer un point d'intérêt avec précision dans une image floue. Même si le point d'intérêt est retrouvé dans l'image de requête et apparié, il se peut que sa position ait changé. Ces imprécisions sont difficiles à estimer a priori, et le RANSAC tente de s'y accommoder en trouvant les meilleurs compromis sur des données bruitées.

Comme précisé dans la paragraphe 3.2.1.3, le calcul de l'orientation absolue repose sur une hypothèse simplificatrice pour accélérer la recherche de la meilleure image pour l'orientation. En cas de mauvais appariement, on peut obtenir une valeur aberrante. Une solution pour minimiser les occurrences de ce cas consiste à prendre plusieurs images pour s'assurer qu'elles sont cohérentes entre elles.

La source principale d'imprécision vient du calcul de l'orientation elle même. En effet, lorsque le robot se décale par rapport à sa position initiale, les hypothèses ne tiennent plus que si les points sont à l'infini. Or cette hypothèse n'est pas garantie dans un environnement de taille réduite. Les hypothèses de calcul sont donc de moins en moins vraies, ce qui fait que l'erreur entre le cap réel du robot et le cap calculé s'agrandit. En pratique, dans un environnement de bureaux, on obtient une précision satisfaisante dans un cercle de 2 mètres de rayon. Cependant, le principe du RANSAC permet d'éviter d'avoir des valeurs aberrantes, puisqu'il se base uniquement sur des points qui restent cohérents avec le modèle.

3.2.2 Approche globale plus précise : la corrélation

On a vu dans la section 3.2.1 qu'il est possible d'obtenir une estimation rapide du cap du robot dans un nœud, avec une méthode parcimonieuse, dont la précision se dégrade avec la distance. Il est donc nécessaire de trouver une méthode complémentaire qui compense ces inconvénients.

Beaucoup d'approches de localisations sont basées sur des méthodes de points d'intérêt parcimonieuses, même celles qui tentent de capturer une information globale comme par exemple (Chapoulie et al., 2011). (Courbon et al., 2008) utilise toutefois une mesure dense autour de descripteurs locaux parcimonieux.

Une approche dense et globale a plusieurs avantages. Cela permet notamment d'être robuste au flou, puisque les imprécisions sur un pixel en particulier sont compensées par la cohérence globale de la zone. Une approche globale se prête aussi particulièrement bien à de basses résolutions, voire en bénéficie, puisque des basses résolutions capturent uniquement la structure globale de l'image.

Parmi les inconvénients possible d'une approche dense, on pourrait citer le coût supplémentaire en calcul, la complexité dépendant des aires considérées. Cependant, on peut se permettre de baisser la résolution de l'image, mais aussi de restreindre la zone de recherche grâce à l'orientation calculée par la boussole par exemple (voir section 3.2.4).

Le principe de la méthode consiste à trouver la zone de l'image de requête la plus corrélée à une zone de l'image de référence. Pour cela, on utilise la mesure de Zero Normalized Cross-Correlation (ZNCC). L'idée est d'utiliser une partie de l'image de référence comme modèle (qui sera par la suite notée T pour *template*), et de calculer la corrélation pour chaque position d'une fenêtre glissante sur l'image de requête de la taille du modèle. On construit ainsi une matrice qui regroupe les différentes valeurs de corrélation pour chaque position de la fenêtre glissante. Notons qu'on considère ici la valeur en niveau de gris de l'image, mais qu'il serait possible de considérer les trois canaux de couleur RVB séparément.

3.2.2.1 Définition de la ZNCC

La mesure de ZNCC est définie comme suit. Pour une image ou une matrice M, on note $M(x,y)$ la valeur stockée à la position (x,y). On note w_M la largeur de la matrice et h_M sa hauteur. On note I l'image de requête et T la partie de l'image de référence utilisée comme modèle. On note C la matrice qui contient les valeurs de ZNCC pour chaque position (x,y) admissible. On suppose $w_I > w_T$ et $h_I > h_T$ (ce qui est nécessaire pour que le principe de la fenêtre glissante ait du sens). Pour simplifier les notations, on considère que les indices des sommes sont toujours compris entre 0 et h_T et entre 0 et w_T respectivement. Alors on définit la mesure selon l'Équation 3.1 :

$$\begin{aligned}
&C(x,y) = \frac{\sum_{i,j} \hat{T}(i,j)\hat{I}(x+i,y+j)}{\sqrt{\sum_{i,j} \hat{T}(i,j)^2 \sum_{i,j} \hat{I}(x+i,y+j)^2}} \\
&\text{où :} \\
&\hat{T}(i,j) = T(i,j) - \frac{1}{w_T h_T} \sum_{k,l} T(k,l) \\
&\hat{I}(i,j) = I(i,j) - \frac{1}{w_T h_T} \sum_{k,l} I(i+k,i+l)
\end{aligned} \tag{3.1}$$

Cette mesure est donc similaire sur le principe à la corrélation classique (Équation 3.2), à ceci près qu'on retranche en chaque point la valeur moyenne de l'image sur une fenêtre de la taille du modèle. On doit notamment tenir compte de la taille du modèle par rapport à celle l'image de requête pour pouvoir considérer les valeurs sur un bon voisinage : cela implique donc de retirer une partie de l'image de requête, trop proche du bord. La Figure 3.10 illustre le principe de la fenêtre glissante, en mettant en valeur la zone dans laquelle on ne peut pas calculer la corrélation à cause de la taille du modèle.

Les valeurs obtenues sont normalisées, comme le nom de la mesure l'indique : on obtient des valeurs comprises entre -1 et 1. Ce calcul s'interprète de la manière suivante : on fait glisser sur l'image de requête une fenêtre de la taille du modèle de référence. Pour chaque position, on calcule la corrélation entre la région de la requête définie par la fenêtre et le modèle.

Pour rappel, la formulation la plus classique de la corrélation est le produit de convolution, formulé comme suit :

$$C_{\text{convol.}}(x,y) = \sum_{i,j} T(i,j) I(x+i,y+j) \tag{3.2}$$

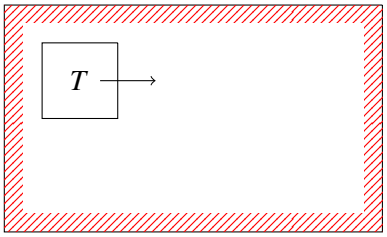


FIGURE 3.10 – Fenêtre glissante et zone sans corrélation (hachurée rouge)

ou encore sa version normalisée :

$$C_{\text{convol. norm.}}(x,y) = \frac{\sum_{i,j} T(i,j) I(x+i, y+j)}{\sqrt{\sum_{i,j} T(i,j)^2 \sum_{i,j} I(x+i,y+j)^2}} \tag{3.3}$$

La ZNCC a plusieurs avantages par rapport à ces mesures. Par rapport à l'Équation 3.2, la normalisation permet tout d'abord de comparer les valeurs de corrélation de façon pertinente, au sein de la même image ou entre plusieurs images. De même, il est plus facile de définir un seuil absolu de corrélation et de l'interpréter avec des valeurs absolues. La mesure de l'Équation 3.3 est normalisée (avec des valeurs comprises entre 0 et 1), mais sa formulation implique un biais pour les valeurs de pixels élevés. Par exemple, une zone entièrement blanche (et donc avec une valeur de pixel élevée) va induire des scores de corrélation élevés par rapport à une zone grise par exemple. La ZNCC au contraire évite ce problème en considérant l'écart à la moyenne, ce qui élimine l'influence de telles zones monochromes.

3.2.2.2 Recherche du maximum de corrélation

Le but est ici de positionner l'image de requête dans l'image de référence. Pour cela, on recherche le maximum de corrélation entre un modèle tiré de la référence et la requête.

Le modèle de référence correspond à une zone angulaire donnée extraite du nœud de référence. Comme cette zone n'est pas a priori située à l'intérieur d'une seule image caméra, on reconstruit cette zone en combinant les images à partir de leur positionnement dans le nœud. Étant donné que le modèle doit être plus petit que l'image de requête, et pour éviter des effets de bord, le modèle ne doit pas prendre la totalité du champ de vision horizontal ou vertical. Plus le modèle est grand, plus on est robuste à des bruits ou des changements locaux, mais plus il est difficile de trouver la totalité du modèle dans l'image de requête. De plus, la complexité du calcul est proportionnelle à l'aire du modèle. La taille du modèle doit donc être adaptée pour obtenir un compromis entre robustesse et temps de calcul.

Une fois la ZNCC calculée, on obtient une matrice qui contient les valeurs de corrélation, de taille $(h_I - h_T, w_I - w_T)$, normalisées. On représentera par la suite ces matrices de corrélation par une image en fausses couleurs, où le rouge représente le maximum de corrélation et le bleu le minimum. On peut alors déduire de la position du maximum la position de la meilleure fenêtre : si on note (x_{max}, y_{max}) la position du maximum dans C, alors la position du coin supérieur droit

de la meilleure position dans l'image est $(x_{max} + w_T/2, y_{max} + h_T/2)$. Il est ensuite possible de convertir la position relative en pixels en une position absolue (voir paragraphe 3.1.2.1).

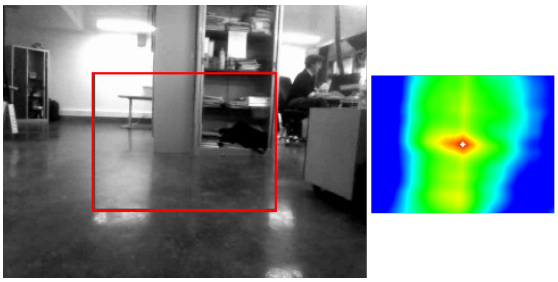


FIGURE 3.11 – Exemple d'auto-corrélation (modèle et maximum de corrélation en rouge)
Plateforme utilisée : NAO

La Figure 3.11 montre un exemple d'auto corrélation d'image. Le modèle est entouré en rouge, et la matrice de ZNCC est représentée en dessous. Le maximum de corrélation est marqué par un point blanc, sur un pic ici bien défini. La Figure 3.12 montre la corrélation entre une image de référence et une autre prise après s'être déplacé légèrement dans sa direction. Ces images correspondent à la concaténation d'images successives prises pendant l'acquisition d'un nœud. On voit que la position du maximum de corrélation et donc de la meilleure fenêtre correspond bien à la position effective du modèle dans l'image de requête, malgré quelques différences dans l'image.

Pour éviter des faux positifs ou des positionnements de mauvaise qualité, on rajoute quelques critères à cette recherche de maximum. Le but est d'éviter des faux positifs, notamment parce qu'il n'y a pas de garantie que le modèle de référence se retrouve dans l'image de requête. Le critère le plus simple est de rajouter un seuil minimal de corrélation, facilité ici par le fait que la

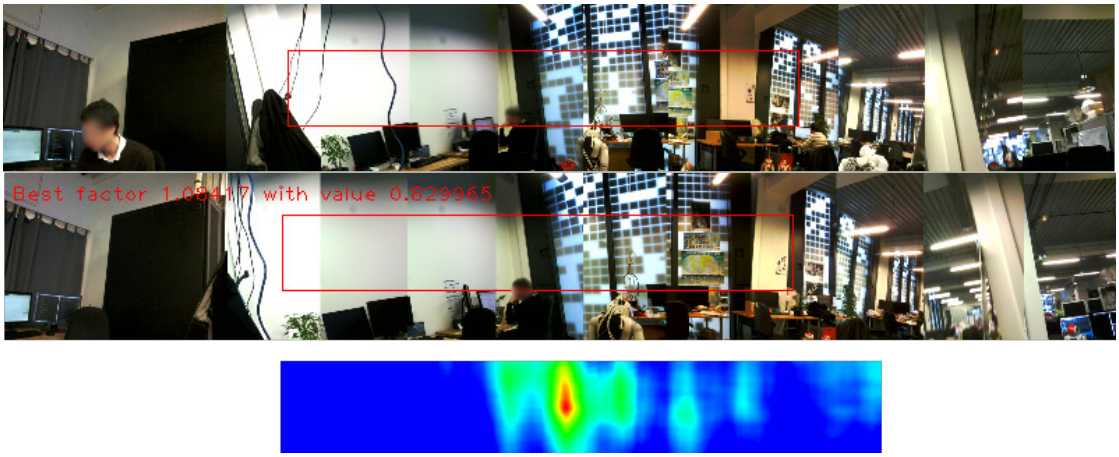


FIGURE 3.12 – Exemple de corrélation : image de référence (en haut), de requête (au milieu) et matrice de corrélation (en bas)
Plateforme utilisée : Pepper

ZNCC est normalisée, mais il est nécessaire d'en rajouter d'autres pour plus de robustesse.

Pour filtrer les faux positifs, on ne considère comme valides que les matrices de corrélation qui présentent un seul pic de corrélation distinct. Cela implique de rejeter les images qui possèdent une structure périodique, mais en pratique cela ne représente pas une contrainte. La zone angulaire considérée est large, si bien qu'il est difficile dans un environnement normal de trouver une telle redondance. En revanche, cette méthode permet d'éliminer des cas de fausses détections, où on voit des pics multiples de faible intensité. On voit par exemple sur la Figure 3.13 plusieurs zones disjointes où la corrélation est forte. La détection de pics multiples peut se faire par exemple à l'aide d'un seuillage de l'image et d'une détection de "blobs" : la partie inférieure de l'image montre le résultat d'un tel seuillage, qui met bien en évidence la présence de deux pics distincts.

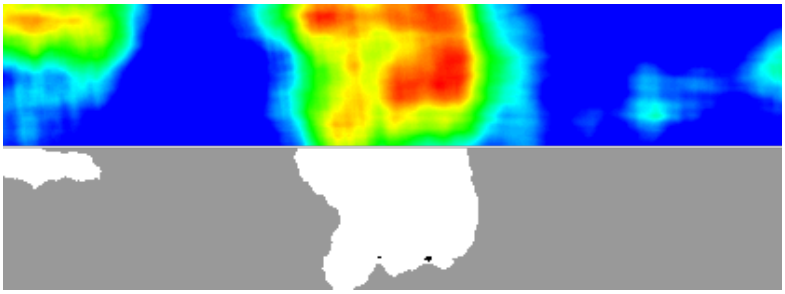


FIGURE 3.13 – Exemple de matrice de corrélation invalide (en haut) et du seuillage correspondant

On peut également rajouter un critère sur l'aspect du pic de corrélation. Dans un cas parfait, le pic doit être bien marqué, mais si l'image est ambiguë ou que la corrélation est de mauvaise qualité, le maximum de corrélation sera plus diffus. C'est le cas par exemple dans la Figure 3.13 : on voit que le pic de gauche est particulièrement large (à comparer notamment à l'aspect de la matrice de la Figure 3.12). Ici aussi, on pourrait éliminer des cas valides si l'environnement est ambigu ou périodique, mais cela permet de rejeter des cas de mauvaise corrélation.

Ces critères sont construits pour éliminer les cas qui sont des sources d'imprécisions potentielles. Cela permet d'augmenter la précision de la méthode, mais implique de baisser le taux de rappel, c'est-à-dire le taux de corrélation effectivement détectée dans une image de requête qui contient bien le modèle.

Prise telle quelle, cette méthode permet d'être précis tout en étant robuste au flou ou à des changements d'environnement légers. Cependant, on ne couvre pas ici une des sources d'imprécision de la boussole visuelle décrite en section 3.2.1 : l'éloignement du robot par rapport à la position du nœud de référence. En effet, s'éloigner du robot implique des changements d'échelle, auxquels la ZNCC n'est pas robuste directement. Pour cette raison, une adaptation multi-échelle a été réalisée : on détecte et on sélectionne la meilleure échelle possible entre l'image de référence et l'image de requête, et on calcule la corrélation sur cette échelle. Les détails de la méthode sont décrits dans la section 3.3.2, puisque la robustesse aux changements d'échelle est particulièrement utile dans le cas de la navigation.

On obtient donc au final la position de la zone de l'image de requête la plus corrélée à l'image

de référence. On peut donc facilement se ramener à la position relative de l'image de requête par rapport à l'image de référence. On obtient également un facteur d'échelle entre l'image de référence et l'image de requête qui permet de quantifier l'éloignement relatif par rapport au nœud de référence. On a donc une information à la fois plus précise et plus complète que le résultat de la boussole visuelle.

3.2.3 Raffinement du positionnement : les données 3D

Comme expliqué dans la partie 1.2, il est parfois possible d'avoir un capteur 3D embarqué sur le robot. Dans ce cas, il est possible d'envisager d'avoir un positionnement métrique du robot par rapport au nœud de référence, en suivant donc un formalisme topométrique hybride. Cette section décrit la méthode employée, et les adaptations dues aux bruits du signal et à la présence éventuelle d'obstructions et de changements.

On utilise la même approche que pour l'acquisition des images de la caméra classique : on enregistre une série d'images de profondeur (qu'on peut ensuite ramener à un nuage de points) qui sont positionnées par rapport à la position initiale lors de la prise du nœud. On suppose que pendant une localisation, on a acquis plusieurs images sur un champ de vision élargi (au moins 180°). Pour des raison de temps de calcul, on réduit ici le nuage de points construit pendant l'acquisition du nœud à un profil en deux dimensions. Pour cela, on sélectionne les points sur une coupe horizontale à hauteur des yeux du robot. On sous échantillonne également le signal pour réduire le nombre de points. Dans la suite, on désignera les points ainsi obtenus par le terme *profil*.

3.2.3.1 Description de l'ICP

La méthode par Iterative Closest Points (ICP) est une méthode communément utilisée pour l'alignement de nuages de points (Chen and Medioni, 1992). Elle consiste à aligner itérativement chaque point d'un nuage sur son plus proche voisin, en calculant une transformation optimale pour tenter d'aligner le nuage. Cette méthode est très communément employée, et possède notamment des variantes à la fois pour des nuages de points à deux et trois dimensions.

On note $T = \begin{pmatrix} x \\ y \end{pmatrix}$ et θ l'angle de rotation la transformation candidate. On note P_{ref} le profil de référence et $P(T,\theta)$ le profil de requête auquel on a appliqué la transformation (T,θ). On note $D(P,P_{ref})$ la distance entre un profil P et le profil de référence P_{ref}. L'Algorithme 3 décrit la méthode employée.

On voit qu'on a besoin de deux outils : une mesure de distance entre deux profils donnés, et une méthode pour trouver un alignement itératif entre deux profils.

Pour donner une mesure de distance entre deux profils, on utilise une méthode efficace basée sur la transformée de distances du profil de référence. La transformée de distance d'un contour donné consiste à stocker pour chaque point d'une image la distance minimale au contour. Cette image se construit facilement pour la distance euclidienne à l'aide de méthode de morphologie mathématique. La Figure 3.14 montre un exemple de transformée de distances sur un profil d'un nœud.

Algorithme 3 ICP pour le recalage de deux profils

Paramètres : s distance minimale pour la convergence, δ changement minimal entre deux étapes.

Entrée : T_0, θ_0 hypothèses préalables de translation et rotation, P profil de requête et P_{ref} profil de référence.
Sortie : T, θ translation et rotation entre P et P_{ref}.
Initialiser T_0 et θ_0.
Initialiser $\Delta D = +\infty$.
while $D(P(T,\theta), P_{ref}) > s$ ou $\Delta D > \delta$ **do**
  Apparier chacun des points de $P(T,\theta)$ avec leur plus proche voisin dans P
  Calculer ΔT, $\Delta\theta$ pour aligner les points de $P(T+\Delta T, R_{\theta+\Delta\theta})$ avec P_{ref}
  $\Delta D \leftarrow D(P(T+\Delta T, \theta+\Delta\theta)) - D(P(T,\theta))$
  $T \leftarrow T + \Delta T$ et $\theta \leftarrow \theta + \Delta\theta$
end while

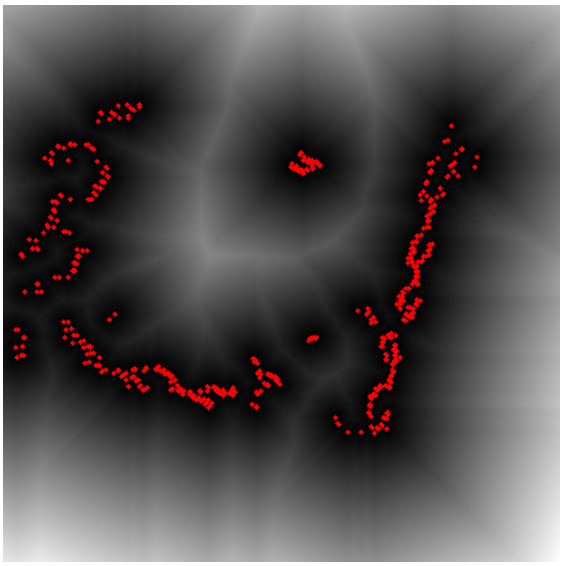


FIGURE 3.14 – Exemple de transformée de distance pour un profil donné (blanc : distance maximale, noir distance minimale, rouge profil de référence)
Plateforme utilisée : Pepper

Pour connaître la distance d'un point au profil, il suffit donc d'accéder au pixel correspondant dans l'image de la transformée. Celle-ci est construite une fois pour toutes après l'acquisition du nœud. Une fois cette distance calculée, on peut l'utiliser pour calculer une mesure classique dans le cas d'alignements de profils : la distance de Chamfer. Celle-ci est définie par l'Équation 3.4. On note $p = \begin{pmatrix} x \\ y \end{pmatrix}$ un point quelconque du profil P, et π un point du profil P_{ref} (repérés par

leur coordonnées planaires).

$$CD(P,P_{ref}) = \frac{1}{|P|}\sum_{p\in P}\min_{\pi\in P_{ref}}||p-\pi|| \tag{3.4}$$

Pour connaître les plus proches voisins de chaque point, on utilise une recherche dans un index qui stocke les positions des points du profil de référence sous la forme d'un *Kd-Tree* (Bentley, 1975), pour une recherche plus efficace.

À partir de cet appariement, on doit trouver la rotation d'angle θ et la meilleure translation. On représente la rotation par la matrice R suivante :

$$R = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

et la translation T par :

$$T = \begin{pmatrix} x \\ y \end{pmatrix}$$

On pose $\mu(R,T)$ et μ_{ref} les barycentres de $P(R,T)$ et de P_{ref} respectivement. On définit l'ensemble des points appariés $S = \{(p,\pi)|p \in P, \pi \in P_{ref}, \text{appariés}\}$ On cherche à minimiser l'erreur suivante :

$$E(R,T) = \sum_{(p,\pi)\in S}||\pi - Rp - T||^2 \tag{3.5}$$

qui correspond à la somme des distances entre les points de référence et leur équivalents appariés dans le profil transformé par R et T.

On peut montrer que la solution optimale provient de la décomposition en valeurs singulières (SVD) de la matrice $C = \sum_{(p,\pi)\in S}(\pi-\mu_{ref})(p-\mu)^T$. La décomposition en valeurs singulières est définie comme suit :

$$C = U\Sigma V^T$$

où U et V sont des matrices unitaires, et Σ est une matrice diagonale $\begin{pmatrix} \sigma_0 & 0 \\ 0 & \sigma_1 \end{pmatrix}$

On peut montrer que la solution optimale est alors :

$$\begin{cases} R = UV^T \\ T = \mu_{ref} - R\mu \end{cases}$$

Dans le cas parfait où les deux profils se correspondent exactement et où tous les points sont appariés, une seule étape suffirait. En pratique, les profils sont différents, en raison du bruit et de changements dans l'environnement. Il n'est pas garanti que l'appariement avec les points les plus proches soit pertinent. Pour cette raison, plusieurs itérations sont souvent nécessaires pour atteindre un alignement satisfaisant. La décomposition en valeurs singulières est obtenue en utilisant la bibliothèque d'algèbre linéaire Eigen.

L'ICP est sensible aux minima locaux. En pratique, si on ne fournit pas une hypothèse de départ raisonnable, la convergence se fait sur une valeur non pertinente, notamment en rotation. Or il n'est pas toujours possible d'avoir une hypothèse sur la position, en particulier dans le cas du robot capturé. Pour traiter ce cas, on cherche à construire une première hypothèse grossière mais rapide, à l'aide de la forme du profil. Pour cela, on utilise le descripteur Shape Context (Belongie et al., 2000). Celui-ci cherche à caractériser la forme d'un profil en subdivisant l'espace autour d'un point donné selon l'orientation (voir Figure 3.15). On représente ensuite cette subdivision par un histogramme, qui recense les points du profil situés dans chaque tranche d'angle. Le descripteur est rendu invariant par rotation en fixant l'origine de l'angle sur la tranche d'angle qui contient le plus de points. On normalise également l'histogramme.

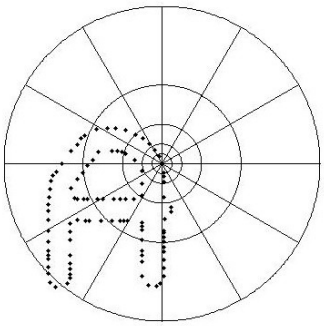


FIGURE 3.15 – Exemple de calcul du descripteur Shape Context
Source : en.wikipedia.org

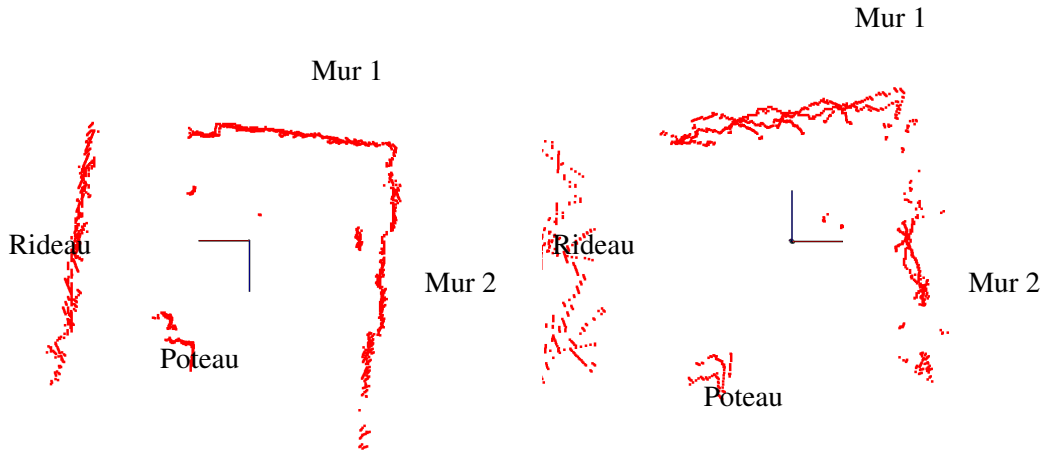
On calcule les descripteurs sur un sous-échantillonage arbitraire du profil, mais en ne retenant que les plus significatifs : on privilégie ceux qui correspondent à une discontinuité dans le profil (par exemple un coin), c'est-à-dire qui ont plus d'une direction notable.

On apparie ensuite les descripteurs en utilisant une mesure de distance entre deux histogrammes, et en faisant une recherche de plus proche voisin. On peut utiliser par exemple une mesure de corrélation entre les histogrammes, ou toute autre mesure adaptée. Une fois ces appariements obtenus, on estime une transformation à l'aide d'un RANSAC qui calcule la transformation de façon analogue à la paragraphe 3.2.1.2. Cette transformation est en général très approximative et rajoute un temps de calcul non négligeable, mais permet d'améliorer nettement la convergence par rapport à une approche sans hypothèse préalable.

Pour calculer la position de l'ICP, on procède donc de la façon suivante. Si on a une hypothèse préalable, alors on calcule le déplacement avec la méthode de l'ICP. Si jamais la convergence de cette méthode n'est pas satisfaisante ou s'il n'existe pas d'hypothèse préalable, alors on calcule l'hypothèse approximative à l'aide des descripteurs Shape Context, et on recommence le calcul de l'ICP avec cette hypothèse. Le calcul échoue donc si les deux approches n'ont pas donné une qualité de convergence satisfaisante. Celle-ci peut être quantifiée grâce à la mesure de la distance finale entre le profil de référence et le profil aligné.

3.2.3.2 Compensation du bruit et des déformations

En pratique, les profils sont significativement différents entre la requête et la référence. Ces différences proviennent de plusieurs facteurs : les plus importants sont l'environnement dynamique et le bruit de mesure.



(a) Sans lentilles (b) Avec lentilles

FIGURE 3.16 – Déformation du signal sur un profil pris par la caméra 3D sans lentilles ou avec lentilles
Plateforme utilisée : Pepper

La principale source de bruit provient des lentilles montées sur les yeux de Pepper. Ces lentilles sont contraintes en forme par des considérations esthétiques (forme arrondie obligatoire dans le design), et en matériau et qualité pour des raisons de coût. Une procédure a été engagée pour tenter à la fois d'améliorer les lentilles et d'investiguer des possibilités de calibration logicielle., mais il n'est pas envisageable de retirer les lentilles ou de les modifier à court terme. Pour cette raison, on utilise le signal sans davantage de traitements.

La Figure 3.16 montre un exemple de profil acquis face à un coin de murs (en haut à droite) d'une part, un rideau à gauche et un poteau en bas. La partie supérieure correspond au profil acquis par le robot en l'absence de lentilles, et la partie inférieure avec des lentilles utilisées actuellement sur le robot. On voit que le bruit introduit est très important, de l'ordre de 50cm d'amplitude pour le mur situé à 2m du robot et de 1m sur le rideau situé à un peu plus de 3m du robot. Les déformations s'accentuent avec la distance, en particulier lorsque la distance dépasse les 2 ou 3 mètres. Ces déformations sont particulièrement gênantes pour l'ICP, puisqu'elles modifient l'aspect du profil fonction de la distance.

Pour compenser les effets de ces différences, on considère seulement les appariements de points qui sont au delà d'une distance minimale. Cela limite la portée de l'appariement, mais permet de filtrer rapidement certains points aberrants et de ne pas tenir compte de ceux qui

auraient disparu ou changé d'un profil à l'autre. Ce filtrage améliore les performances de l'ICP en évitant de considérer des points aberrants.

De même, pour l'évaluation de la distance globale entre le profil aligné et le profil de référence, on ignore les points qui sont situés à une distance trop grande du profil, tout en surveillant le taux de points ainsi rejetés pour éviter de privilégier des profils très éloignés. Cela permet par exemple de ne pas pénaliser un alignement correct où la différence provient d'un nouvel obstacle rajouté à la référence. Par exemple, on voit sur la Figure 3.17 qu'une partie du profil est masqué par un obstacle, en l'occurrence une personne venue se placer devant le robot. Le profil de référence est indiqué en rouge : on y distingue notamment deux murs entiers et un pilier. Le profil courant aligné en indiqué en vert. On peut y voir trois obstacles qui n'existaient pas dans le profil de départ (en l'occurrence, des personnes placées autour du robot). On voit également qu'une partie du champ de vision est masquée par ces obstacles (une partie du mur à droite de l'obstacle). Malgré ces différences, le profil est aligné correctement et on lui associe un score relativement élevé.

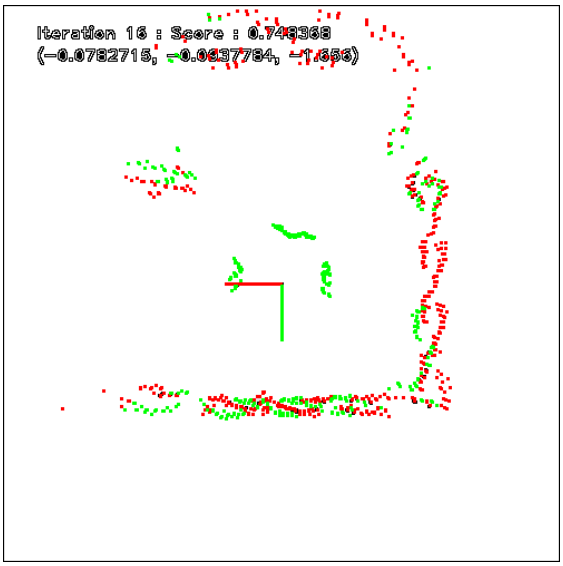


FIGURE 3.17 – Exemple d'alignement réussi (profil vert) en présence d'un obstacle supplémentaire sur le profil de requête (en rouge)
Plateforme utilisée : Pepper

3.2.4 Renforcement par la structure en cascade

3.2.4.1 Description et justification de la structure

On a vu dans les sections précédentes qu'il existe des méthodes qui fournissent des informations de localisation partielles. Cependant ces méthodes sont progressivement de plus en plus lourdes à mettre en place, mais elles ont une meilleure précision et une complexité plus faible si elles disposent d'une hypothèse indicative préalable. Il est donc naturel d'utiliser ces méthodes

de façon hiérarchique, c'est-à-dire que les différents algorithmes utilisent les résultats des précédents pour renforcer leurs résultats et diminuer leur complexité.

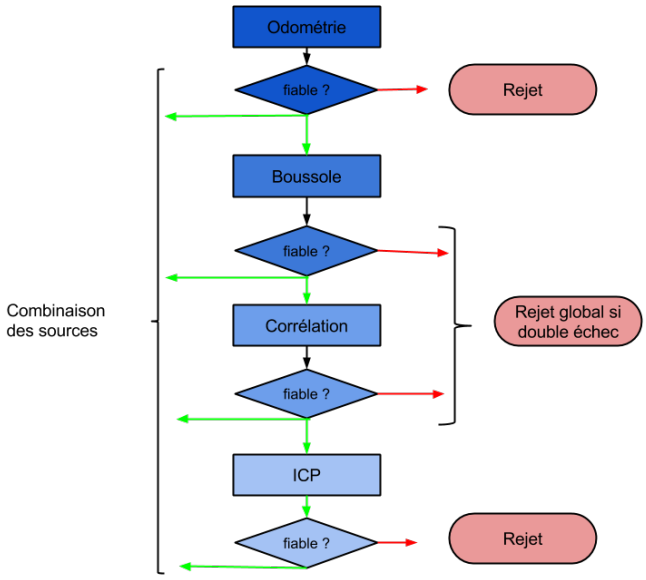


FIGURE 3.18 – Représentation schématique de la structure en cascade de la localisation (flèche rouge : échec, verte : succès)

La Figure 3.18 représente cette structure en cascade. On y voit qu'on enchaîne chaque source de localisation (rectangle bleu) avec un test (losange associé). Si la source calcule une information avec succès (flèche verte), l'information est passée aux sources suivantes et finalement combinée avec les autres. En cas d'échec (flèche rouge), l'information est rejetée. Le principe de cette cascade est que chaque source utilise les informations des sources précédentes si elles sont disponibles, et dans le cas contraire calcule ses informations sans hypothèses préalable. En cas de succès, l'information de la source est transférée aux sources suivantes. Les sources sont arrangées par ordre de précision, et chaque source quantifie la précision du résultat donné.

La cascade est basée sur la mesure de l'odométrie. En effet, il s'agit de la méthode la plus simple et la plus rapide, mais aussi de la moins précise. Sa précision se dégrade avec le temps, en raison des glissements du robot. La localisation du robot est estimée en ajoutant le déplacement relatif estimé par l'odométrie depuis la dernière localisation connue. On obtient alors l'information de position en deux dimensions par rapport au nœud, (x, y, θ). Pour quantifier la fiabilité de l'odométrie, on se base sur le déplacement cumulé depuis la dernière localisation, ce qui permet de distinguer par exemple le cas où le robot est resté immobile de celui où le robot aurait effectué plusieurs tours sur lui même. On voit que dans le premier cas, l'information d'odométrie est fiable, alors que dans le second l'erreur cumulée sur les tours induit des imprécisions. Le cas du robot capturé, c'est-à-dire sans localisation préalable connue, est un cas d'échec. Un autre cas d'échec est lorsque le robot a chuté, a été poussé ou soulevé depuis la dernière localisation. Ces événements perturbent très fortement l'odométrie qui n'est en général plus pertinente après.

Enfin, on peut rajouter un dernier cas d'échec basé sur un seuil appliqué au déplacement cumulé depuis la dernière localisation.

Comme décrit dans la section 3.2.1, la boussole est une source de localisation rapide, plus précise que l'odométrie. Si l'information de l'odométrie est disponible, on utilise l'angle estimé θ comme hypothèse de départ pour la recherche de l'orientation (voir paragraphe 3.2.1.2 en particulier l'algorithme de recherche Algorithme 2). Sinon, on utilise une position de départ arbitraire pour la recherche. Si l'information de l'odométrie n'est pas complètement fausse, alors la recherche sera accélérée et on aura moins de risque de faux positifs dans la recherche de l'image. On obtient donc une information de position angulaire du robot par rapport au nœud, θ. La précision de cette source peut être quantifiée à l'aide du taux de points cohérents. Si la recherche de la meilleure image de référence échoue, ou bien si le taux ou le nombre de points cohérents est trop faible, la source échoue.

La corrélation, comme décrit dans la section 3.2.2, est une source à la fois plus précise et plus complète que la boussole, mais qui sans hypothèse préalable est à la fois plus lourde et possède un rappel plus faible. Si une hypothèse préalable est disponible, alors on peut l'utiliser pour restreindre le champ d'extraction de l'image de référence, et donc réduire la complexité de la méthode. On a également de meilleures chances de retrouver le modèle de l'image de référence dans l'image de requête si la zone est correctement circonscrite. Si possible, la corrélation utilise l'information du compas comme hypothèse préalable, puis l'information d'odométrie sinon, par ordre de précision décroissante. Si aucune hypothèse préalable n'est disponible, on utilise l'ensemble des images. La précision de la corrélation peut être quantifiée à l'aide de la valeur de la ZNCC, qui donne une mesure absolue. Si cette valeur de corrélation est trop faible, ou si le maximum trouvé n'est pas valide, alors on est dans un cas d'échec.

Si les deux sources de localisation issues de la vision, la boussole et la corrélation, échouent toutes les deux, alors on rejette la localisation. En effet, on considère que si l'information de vision contenue dans la requête n'a pas de point commun avec l'information de référence, alors les chances que l'information soit pertinente sont très faibles. Cela signifie que la référence n'est pas visible depuis la requête.

Enfin, l'ICP (section 3.2.3) est placé en fin de cascade. L'hypothèse préalable est ici particulièrement importante, puisqu'elle améliore fortement la convergence, en particulier pour l'orientation. On a plusieurs cas de figure possible :

— la source d'odométrie est disponible. Dans ce cas, on construit une hypothèse (x_h, y_h, θ_h) qui combine l'information de position de l'odométrie et l'information d'angle la plus précise possible ;

— la source d'odométrie n'est pas disponible, mais au moins une des sources de vision l'est. Dans ce cas, la position de départ est $(0, 0, \theta_h)$ où θ_h est l'information d'angle la plus précise possible. Cette hypothèse est a priori fausse en termes de position, mais permet de donner un point de départ pertinent en termes d'orientation.

Plus l'hypothèse préalable est précise, meilleure sera la convergence et moins on risque d'avoir à recourir à la méthode sans hypothèse du Shape Context. Les cas d'échec possibles viennent d'un défaut de la convergence de l'ICP, ou bien d'une trop grande distance entre le profil estimé et le profil de référence.

Cette structure permet donc de rendre les calculs à la fois plus robustes et plus rapides.

L'utilisation des hypothèses est ici simple mais fournit des résultats significativement meilleurs que si toutes les sources de localisation sont prises séparément.

3.2.4.2 Combinaison et renforcement des résultats

Comme décrit dans la section précédente, les différentes sources de localisation et d'information donnent différents résultats, avec une incertitude associée. Il reste à les combiner pour obtenir une estimation finale de la localisation. Pour cela, on utilise un modèle probabiliste très simple. Notons qu'il serait possible d'utiliser un modèle plus complexe.

On assimile le résultat de chaque source de localisation à une Gaussienne sur trois dimensions. On considère que les trois coordonnées (x, y, θ) sont indépendantes. Dans le cas d'une source calculant (x_s, y_s, θ_s), on lui associe donc la loi de probabilité suivante :

$$\begin{aligned} P_s(x,y,\theta) &= P_s(x)P_s(y)P_s(\theta) \text{ grâce à l'indépendance} \\ &= e^{\frac{x_s-\mu_{x_s}}{\sigma_{x_s}}} e^{\frac{y_s-\mu_{y_s}}{\sigma_{y_s}}} e^{\frac{\theta_s-\mu_{\theta_s}}{\sigma_{\theta_s}}} \end{aligned}$$

où $\mu_{x_s}, \mu_{y_s}, \mu_{\theta_s}$ correspondent au résultat calculé par la source et $\sigma_{x_s}, \sigma_{y_s}, \sigma_{\theta_s}$ correspondent à la variance calculée à partir des incertitudes de chaque source. Si une coordonnée n'est pas calculée par la source, alors on lui associe une loi de probabilité uniforme.

On considère également que toutes les sources sont indépendantes. Cela permet donc de calculer les lois de probabilité en multipliant coordonnée par coordonnée. On obtient donc globalement :

$$\begin{aligned} P(x,y,\theta) &= \prod_{s\in S} P_s(x,y,\theta) \\ &= \prod_{s\in S} P_s(x)P_s(y)P_s(\theta) \\ &= \prod_{s\in S} P_s(x) \prod_{s\in S} P_s(y) \prod_{s\in S} P_s(\theta) \end{aligned}$$

Pour estimer l'incertitude de la combinaison, on utilise la fonction de répartition. Celle-ci est définie par :

$$\begin{aligned} F(x,y,\theta) &= P(X <= x, Y <= y, \Theta <= \theta) \\ &= \int_{x=-\infty}^{x=+\infty} \int_{y=-\infty}^{y=+\infty} \int_{\theta=-\pi}^{\theta=+\pi} P(x,y,\theta)dxdyd\theta \\ &= \int_{-\infty}^{+\infty} P(x)dx \int_{-\infty}^{+\infty} P(y)dy \int_{-\pi}^{+\pi} P(\theta)d\theta \end{aligned}$$

Il est également utile de considérer sa réciproque :

$$Q(p) = (x,y,\theta) \text{ tel que } F(x,y,\theta) = p$$

La fonction de répartition et sa réciproque ne peuvent pas être exprimée de façon analytique. On utilise donc une approximation en échantillonnant la loi de probabilité et en approchant la fonction de répartition correspondante et sa réciproque sur un voisinage fixé. Les paramètres

d'échantillonnage et la taille du voisinage sont choisis en fonction de la portée et de la précision voulues.

On considère que le résultat de la combinaison (x_f, y_f, θ_f) correspond à l'antécédent du maximum de la loi de probabilité. Pour estimer l'incertitude associée, on utilise la fonction de répartition et sa réciproque, pour déterminer $(x_{lim}, y_{lim}, \theta_{lim})$ tel que :

$$P(x_f - x_{lim} < X < x_f + x_{lim}, y_f - y_{lim} < Y < y_f + y_{lim}, \theta_f - \theta_{lim} < \Theta < \theta_f + \theta_{lim}) = p_{lim}$$

où p_{lim} est une probabilité choisie a priori en paramètre.

Il est important de noter que les hypothèses sont très simplificatrices. En particulier, l'indépendance des coordonnées entre elles pour une même source est fausse, par exemple dans le cas de l'odométrie : si le robot tourne sur place, cela induit également un léger bruit sur sa position. Ces effets sont ici négligés pour plusieurs raisons. Il est difficile d'estimer un modèle fiable, car il dépend fortement des sols sur lequel le robot circule : il ne suffit donc pas d'identifier les paramètres du robot. De plus, on considère que l'échelle de temps entre deux localisations est suffisamment faible pour pouvoir négliger ces effets, ce qui est vrai en pratique. De même, l'hypothèse d'indépendance des sources peut être discutable, surtout dans le cas de la structure en cascade. Cependant, les différentes sources utilisent les informations précédentes uniquement comme une hypothèse, et sont en particulier robustes à une fausse hypothèse, si bien qu'il n'est pas déraisonnable de les considérer comme indépendantes. Dans le cas où il serait possible de modéliser ces dépendances, le modèle est suffisamment générique pour permettre ces améliorations.

Notons au passage le cas particulier de l'angle. Pour calculer les lois de probabilité sans avoir de problème de modulo, on arrange les angles calculés par les différentes sources de manière à ce qu'ils soient tous le plus proche possible. L'Algorithme 4 décrit cet arrangement. On considère que l'écart-type est suffisamment faible pour négliger le fait qu'une Gaussienne n'est pas une représentation exacte pour un intervalle fini.

Algorithme 4 Ajustement des angles préalable au calcul

Entrée : $\theta_0, \ldots, \theta_n$ angles calculés par la boussole.
Sortie : $\theta_0, \ldots, \theta_n$ angles placés dans le même modulo.

```
Sélectionner θ_0.
for all θ_i do
    if |θ_i − θ_0| > 2π then
        θ_i ← θ_0 + (θ_i − θ_0 (mod 2π))
    end if
end for
Considérer la Gaussienne sur l'intervalle ]θ_0 − π, θ_0 + π[.
```

Cette combinaison hiérarchique des résultats est utilisée également pour tenter de minimiser la prise d'images pour la localisation. Pour cela, on utilise les mêmes étapes que pour construire un nœud (voir paragraphe 3.1.2.1) en les décomposant. On procède de la manière suivante. On commence par acquérir les 8 premières poses du nœud. On tente de calculer la localisation

du robot sur ces 8 images. Si la localisation réussit, il est inutile de faire continuer le robot davantage. Si elle échoue, c'est-à-dire si aucune des sources basées vision ne réussit, que la combinaison des Gaussiennes échoue ou que l'incertitude est trop grande, alors on complète en effectuant la seconde moitié d'un nœud. On considère alors la totalité des images pour la localisation.

Cela permet de préciser les estimations si jamais l'incertitude préalable était trop grande, en étendant le champ de vision considéré. Par exemple, pour le cas de la boussole visuelle, on peut effectuer le calcul de l'orientation sur davantage d'images, et raffiner la moyenne pondérée. Le plus grand champ de vision considéré permet aussi de compenser l'effet d'obstructions sur la première moitié. Dans le cas des données 3D, cela peut aussi être utile par exemple si une partie de l'environnement est hors de portée du capteur et donc ne contient pas d'information pour la localisation. Enfin, cette méthode a l'avantage de chercher à minimiser le nombre d'images prises, et donc le temps d'acquisition et de calcul. Idéalement, on pourrait même effectuer le calcul incrémentalement à chaque image, en parallèle de l'acquisition, et arrêter l'acquisition dès qu'une localisation est atteinte. Cependant, à part la boussole où une seule image est éventuellement suffisante, il faut acquérir plusieurs images pour que le champ de vision soit assez large.

3.2.5 Identification du nœud courant

Les sections précédentes décrivent comment il est possible de construire une information de localisation par rapport à un nœud. Il reste à résoudre le problème de l'identification du nœud courant. Celui-ci se pose dans le cas du robot capturé, ou tout simplement en cas d'incertitude trop forte.

Pour identifier le nœud courant, on utilise l'approche par sac de mots. Comme on a l'a vu dans la paragraphe 2.3.1.2, les systèmes de localisation topologique basés sur des caméras utilisent couramment cette approche. On cherche donc ici à adapter ces techniques aux descripteurs et aux contraintes de temps de calcul.

Contrairement à la plupart des approches classiques, on utilise des descripteurs binaires. On s'inspire donc de l'approche de (Galvez-Lopez and Tardos, 2012) qui construit un sac de mots hiérarchique basé sur des descripteurs BRIEF. Cette approche présente deux adaptations notables par rapport à l'approche la plus classique.

La première adaptation vient de la méthode d'agrégation des descripteurs d'origine pour construire un nombre donné de mots du vocabulaire. Pour se ramener à la méthode d'agrégation classique du *k-means*, on plonge les descripteurs binaires de l'espace $\{0,1\}^{256}$ dans l'espace réel $\mathbb{R}^{256}$. On effectue ensuite l'algorithme du k-means comme pour descripteurs normaux, et on obtient des descripteurs réels dans $[0,1]^{256}$, que l'on ramène ensuite dans l'espace de départ en arrondissant. L'algorithme classique est décrit par l'Algorithme 5. Il suffit de remplacer la norme L_2 utilisée ici par la distance de Hamming. La distance de Hamming est une distance adaptée aux vecteurs binaires : il s'agit du nombre de bits différents entre deux vecteurs binaires. Comme il s'agit de vecteurs binaires, la distance est équivalente à la norme L_1 pour les réels.

L'autre adaptation utile consiste à construire le vocabulaire de façon hiérarchique. En effet, la complexité du k-means dépend du nombre de centres recherchés, et un vocabulaire est typiquement constitué de quelques milliers de mots. La méthode employée ici consiste donc à

Algorithme 5 Méthode classique du k-means

Paramètres : k nombre de centres désirés, d déplacement minimal entre deux itérations.

Entrée : S ensemble des descripteurs.
Sortie : $m_0, ..., m_k$ centres et $V_0, ..., V_k$ descripteurs associés.
Initialiser $m_0^0, ..., m_k^0$ et $V_0^0, ..., V_k^0$.
Initialiser index itération $n = 1$.
while $\exists i \in [0,k], ||m_i^{n-1} - m_i^n|| > d$ **do**
Calculer les k cellules de Voronoi $V_i^n = \{x \in S | \forall n \in [|0,k|], ||x - m_i|| \leq ||x - m_n||\}$ associées à $m_0^n, ..., m_k^n$
for all $V_i^n \in (V_0^n, ..., V_k^n)$ **do**
$$m_i \leftarrow \frac{1}{|V_i^n|} \sum_{x \in V_i^n} x$$
end for
$n \leftarrow n + 1$
end while
$m_0, ..., m_k \leftarrow m_0^n, ..., m_k^n$, $V_0, ..., V_k \leftarrow V_0^n, ..., V_k^n$

construire un arbre d'arité k et de profondeur p (Figure 3.19). Les k^p feuilles de l'arbre constituent le vocabulaire final.

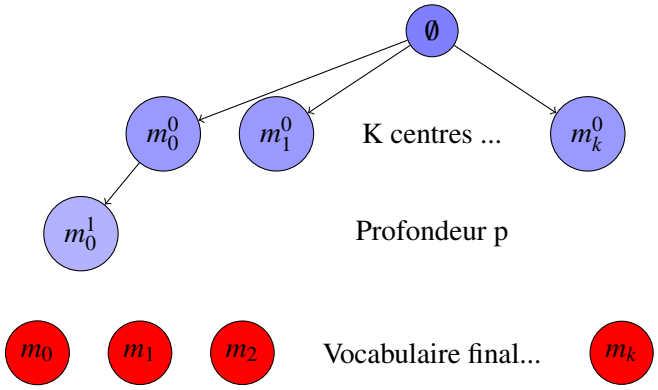


FIGURE 3.19 – Structure du vocabulaire hiérarchique

La construction du vocabulaire est récursive, et décrite dans l'Algorithme 6. On y construit des mots intermédiaires, pour limiter le nombre de centres demandés à k, sur des ensembles intermédiaires de plus en plus petits. À chaque mot intermédiaire calculé, on associe les descripteurs de l'ensemble original dont le mot intermédiaire est le plus proche voisin. Cette construction se fait naturellement pendant l'heuristique du k-means. On a donc à la fois l'avantage de travailler sur un nombre plus faible de centres, mais aussi sur un nombre plus faible de descripteurs.

En ce qui concerne l'appariement d'un descripteur de requête à un mot du vocabulaire, la structure hiérarchique est également utile. Pour apparier un descripteur donné, on parcourt

Algorithme 6 Construction du vocabulaire

Paramètres : p profondeur et k arité de l'arbre.

Entrée : D ensemble des descripteurs originaux.
Sortie : $m_0, \ldots, m_{k^p}$ mots du vocabulaire et arbre associé.
On note M_n l'ensemble de mots contenus dans un étage n.
On note V_m l'ensemble des descripteurs originaux associés à un mot m.
On note m_i^n le i-ème mot de l'étage n.
On initialise $M_0 = \emptyset$ et D_0 à l'ensemble des descripteurs originaux.
for $n \in [0, p[$ **do**
  for $m_i^n \in M_n$ **do**
    Construire $\{m_{ki}^{n+1}, \ldots, m_{k(i+1)-1}^{n+1}\}$ les k centres de $V_{m_i^n}$.
    Stocker les ensembles $V_{m_{ki}^{n+1}}, \ldots, V_{m_{k(i+1)-1}^{n+1}}$ associés.
    $M_{n+1} \leftarrow M_{n+1} \cup \{m_{ki}^{n+1}, \ldots, m_{k(i+1)-1}^{n+1}\}$
  end for
end for

l'arbre en cherchant pour chaque nœud le plus proche voisin du descripteur parmi ses enfants, et en continuant récursivement à partir de ce voisin. On ramène donc l'appariement à pk tests : k comparaisons par étage pour p étages.

Cette méthode a l'avantage d'être rapide à calculer et de tirer parti des mêmes informations que les sources de localisation. Le vocabulaire de mots visuels est rapide à construire grâce à la structure hiérarchique, et l'appariement aux mots visuels également. On peut alors appliquer une approche d'identification par sac de mots classique, par exemple en utilisant la mesure de TF-IDF (Angeli et al., 2008) ou de TS-IDS (Chapoulie et al., 2011).

On a donc une méthode d'identification du nœud courant. Cette identification peut être effectuée dans le cas du robot capturé (redémarrage du robot par exemple), ou tout simplement sur les nœuds voisins si la localisation par rapport au nœud supposé courant échoue.

3.2.6 Conclusion

On a montré ici qu'il était possible, en respectant les contraintes du problème, d'extraire des informations de localisation à la fois quantitatives par rapport à un nœud du graphe et qualitatives en identifiant le nœud courant du graphe.

Pour obtenir la localisation du robot par rapport à un nœud donné du graphe, on utilise une méthode progressive à la fois en termes d'acquisition d'images et de structure de calcul. On utilise une série de sources de localisation organisées dans une structure en cascade, par ordre croissant de complexité et de précision : cela permet d'augmenter la robustesse et de diminuer le coût de chaque source en tirant parti des informations précédentes pour construire une hypothèse préalable. On combine les résultats de chaque source de façon probabiliste en approchant le comportement des sources par des Gaussiennes et en faisant des approximations d'indépendance.

Les sources de localisations sont basées sur les capteurs disponibles du robot.

— L'**odométrie** est la plus simple des sources de localisation. Elle est sujette à une forte dérive et non significative dès que la marche du robot est perturbée, elle a l'avantage d'être facile à calculer et de fournir au moins une restriction dans le calcul des localisations suivantes. Pour avoir une information plus précise, il est nécessaire d'exploiter l'information donnée par les caméras couleur.
— La **boussole visuelle** est une approche parcimonieuse basée sur des points d'intérêt, qui consiste à estimer l'orientation de l'image de requête par rapport au nœud de référence en considérant les déplacements de points appariés, et donc à fortiori le cap du robot. Cette méthode est relativement rapide à exécuter, et beaucoup plus précise que l'odométrie dans un domaine proche du nœud. Une hypothèse préalable sur l'angle du robot permet de restreindre la recherche d'appariement et donc d'accélérer le calcul. Cependant, sa précision se dégrade avec l'éloignement au nœud.
— La **corrélation** d'images grand angle est une méthode dense qui permet de retrouver des modèles extraits du nœud dans les images de requête, et d'en tirer à la fois la position angulaire et une estimation d'un facteur d'échelle entre les images de référence et de requête. Cette méthode permet d'être robuste au flou et au changement d'échelle, et vient donc compléter avantageusement la boussole. Elle utilise d'éventuelles hypothèses préalables pour restreindre le modèle de référence recherché.
— Enfin, si un capteur 3D est disponible (par exemple sur ROMEO), on peut en tirer parti à l'aide d'un **alignement de profils** basé sur une méthode d'ICP. Cette approche permet d'estimer une position métrique par rapport au nœud de référence. La convergence de cette méthode est dépendante de l'hypothèse préalable, à la fois en termes de vitesse et de précision.

Les informations pour la localisation sont acquises progressivement. On interroge régulièrement l'odométrie, et on s'en contente tant que le bruit estimé reste dans des bornes acceptables. On tente ensuite une localisation à partir d'une série d'images prises en laissant la base du robot fixe. Si cette information n'est pas suffisante, on la complète avec un demi-tour puis une autre série d'images.

Pour la localisation qualitative du robot dans le graphe, on utilise un système de sac de mots basés sur les descripteurs binaires extraits pour la boussole visuelle. Cela permet d'obtenir un système d'identification de nœud à bas coût.

On a donc construit un système de localisation en identifiant les informations qui peuvent être extraites des capteurs limités. L'accent est mis sur l'adaptation et l'optimisation de méthodes classiques pour en faire des méthodes originales adaptées aux fortes contraintes de la plateforme. Les résultats quantitatifs de la localisation sont présentés dans la partie 4.2.

3.3 Transitions d'un nœud à l'autre

On a construit un système de localisation par rapport à un nœud donné, et d'identification du nœud courant. Il reste alors à effectuer et à contrôler les transitions d'un nœud à l'autre. Or le contrôle de trajectoire (en position et en vitesse) proposé par défaut sur le robot est basé sur l'odométrie qui est elle même en boucle ouverte, et donc soumise à de fortes dérives. La dérive est particulièrement visible sur les rotations du robot. On met donc en place deux méthodes pour

contrôler les transitions entre nœuds : la première est focalisée sur le contrôle de la direction du robot pendant son déplacement, l'autre sur la distance parcourue.

3.3.1 Contrôle de trajectoire

La marche du robot est implémentée pour être stable, en particulier en cas de petits obstacles et d'irrégularités dans le sol. Elle peut être contrôlée soit en position et orientation (on donne une consigne (x, y, θ) au robot), soit en vitesse. Dans le dernier cas on peut contrôler indépendamment la vitesse angulaire et la vitesse en translation suivant les axes X et Y. Cependant, ces paramètres sont estimés en boucle ouverte à partir des mesures de position des MRE situés dans les jambes, et sont donc sujets à des bruits et aux simplifications du modèle. La plus grand incertitude de la marche porte sur le cap du robot. En particulier, le robot NAO n'est pas capable de marcher droit. Les paramètres ne peuvent pas être estimés a priori car ils dépendent de chaque robot, et du sol sur lequel le robot évolue. Les glissements sont par exemple très différents lorsque le robot évolue sur de la moquette ou sur un sol lisse. On doit donc trouver un moyen extérieur d'estimer le cap du robot relativement à la position de départ de la base du robot.

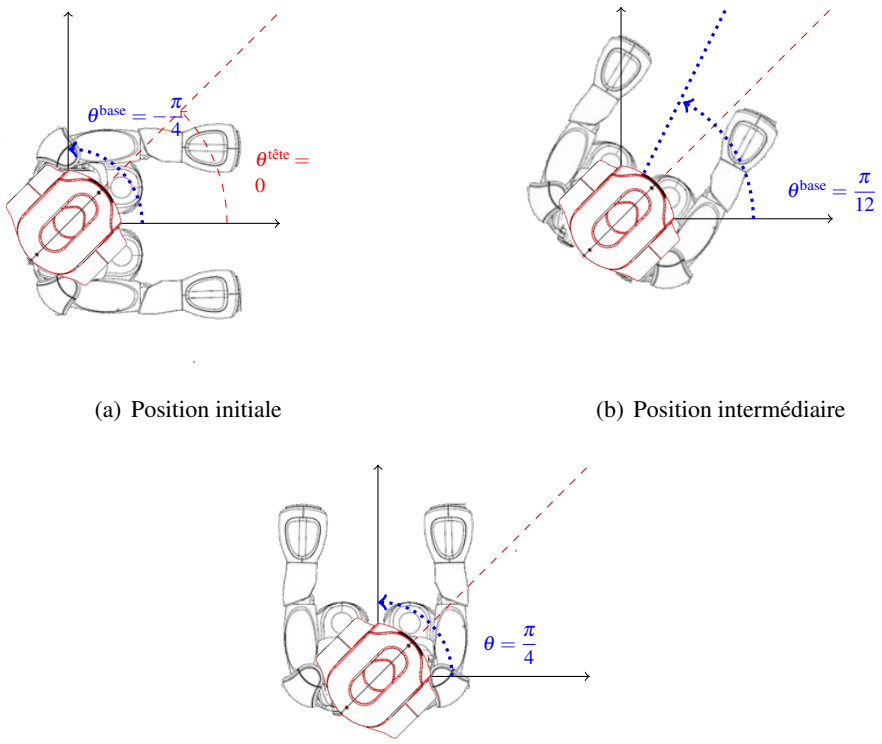
Pour cela, on utilise la méthode de calcul de l'orientation relative de l'image décrite dans la paragraphe 3.2.1.2. Le principe consiste à définir l'image de référence à la volée au lieu de l'extraire d'un nœud. Une fois cette image de référence définie, on contrôle le cap du robot pour effectuer soit une rotation sur place (autour de l'axe Z), soit une translation en ligne droite. On peut alors contrôler la rotation du robot autour de l'axe Z en utilisant la mesure de la déviation de l'image vue par la tête du robot. On considère que connaître la position de la tête du robot est équivalent à connaître la position de son torse. En effet, la précision des capteurs de position MRE situés entre la tête et la base du robot est suffisante et la mesure assez rapide pour que l'erreur soit considérée comme négligeable, notamment par rapport à la précision de la mesure de la déviation de l'image. Par exemple, sur le robot NAO, il n'y a que deux MRE sur la chaîne cinématique entre la tête et le torse, soit une précision de $0.1°$.

Le déplacement du robot et le contrôle de sa tête sont considérés ici comme une boîte noire, si bien qu'on doit notamment prendre en compte l'influence du déplacement de la base dans les mouvements de la tête. On peut toutefois noter qu'il existe des formes de contrôle en cascade qui auraient pu permettre d'intégrer directement les contraintes en position de la tête dans la marche du robot.

3.3.1.1 Contrôle d'une rotation

Pour effectuer une rotation sur place, on définit la référence de la manière suivante. On note $\theta \in]-\pi, \pi]$ la rotation à effectuer par rapport à la position initiale de la base du robot. On tourne la tête du robot de façon à ce que sa position angulaire soit $\frac{\theta}{2}$ par rapport à la position initiale du robot. La référence est définie par l'image visible par la tête du robot dans cette position. On note $\theta^{\text{tête}}$ l'angle entre la tête et la direction de référence, et θ^{base} l'angle entre la base et la direction de référence. Alors la consigne pour $\theta^{\text{tête}}$ est $\theta^{\text{tête}}_{\text{ref}} = 0$, avec une position initiale $\theta^{\text{tête}}_{0} = 0$, et la consigne pour θ^{base} est $\theta^{\text{base}}_{\text{ref}} = \frac{\theta}{2}$. La Figure 3.20 monte ces notations et différentes étapes de la

rotation sur place. La direction de référence de la tête est représentée par un trait pointillé rouge, et celle du corps par des points bleus.



(a) Position initiale

(b) Position intermédiaire

(c) Position finale

FIGURE 3.20 – Positions successives pour une rotation de $\frac{\pi}{2}$

On choisit de contrôler indépendamment la position de la tête par rapport au robot et la position de la base du robot. Cela se justifie par le fait que la tête peut être contrôlée beaucoup plus rapidement que la base du robot, qui peut être soumise à des glissements ou des contraintes venant du placement des pas. Le but est de maintenir la tête dans la direction de référence le plus rapidement possible, tout en adaptant la position de la base. Cela permet par exemple d'être robuste au fait que la rotation de la base du robot est gênée par un obstacle.

Pour contrôler la tête et la base, on utilise des commandes en vitesse, calculées avec des contrôleurs PID. Le principe de ce contrôle est décrit dans la Figure 3.21.

On commence par considérer et paramétrer un contrôleur PID pour la tête uniquement, qui permet de maintenir la tête dans une direction donnée indépendamment du corps. On a une mesure de la position de la tête $\hat{\alpha}$ grâce à la méthode de la boussole. Les paramètres sont obtenus par une méthode manuelle.

Ces paramètres ont d'abord été approchés en réalisant des expériences dans le simulateur Webots (voir section 1.2.4 pour plus de détails sur la simulation) puis ajustés sur les robots réels.

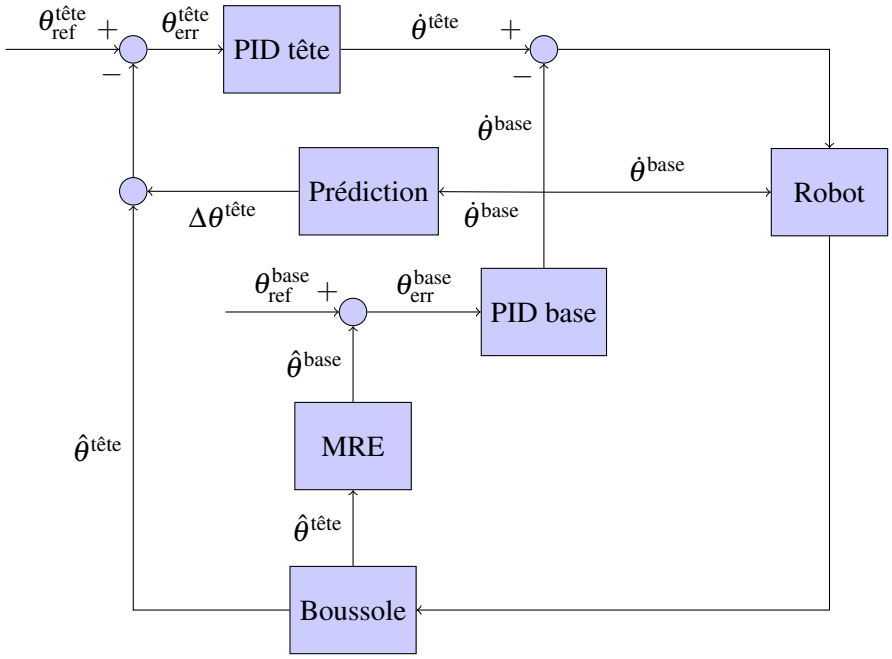


FIGURE 3.21 – Contrôleurs PID de la tête et du corps du robot pour une rotation pure

La section 4.3.1 montre les résultats de convergence obtenus.

On obtient alors un contrôleur pour la tête qui donne une mesure de la vitesse consigne de la tête par rapport à la direction donnée, $\dot{\alpha}$. Une fois ce contrôleur obtenu, on intègre les mouvements de la base. On commence par ajouter un contrôleur PID pour la base. La mesure de l'angle de la base $\hat{\theta}$ se fait grâce à la mesure de l'angle de la tête et des positions des MRE. On se base ensuite sur la constatation simple que la vitesse de la tête par rapport à la référence est la somme de la vitesse de la tête et de celle de la base. Pour tenir compte de cela, on ajoute une prédiction à la mesure de l'angle courant de la tête $\hat{\alpha}$: à partir de la commande en vitesse $\dot{\theta}$ calculée pour la base, on ajoute un angle $\Delta\alpha_p = \dot{\theta}\Delta t$, où Δt est le pas de temps du calcul du PID. Cette mesure modifiée est alors fournie en entrée du contrôleur de la tête. On donne en consigne pour la vitesse de la tête $\dot{\alpha} - \dot{\theta}$. En effet cette consigne est donnée dans le référentiel du robot.

On a donc un contrôleur qui permet de maintenir la direction de la tête et de contrôler celle du corps. Les performances en convergence sont détaillées dans la section 4.3.1. Le contrôle décrit ici n'est fait que pour la rotation autour de l'axe Z vertical. Cependant la tête possède un degré de liberté supplémentaire pour la rotation autour de l'axe Y, et le calcul de l'orientation fournit également une estimation de cette rotation. Il serait donc envisageable de contrôler également ce degré de liberté pour stabiliser la tête. La rotation autour de l'axe X correspond au mouvement de balancier latéral du robot pendant la marche. Elle peut être aussi calculée par la boussole, mais ne peut pas être compensée car la tête du robot ne possède pas ce degré de liberté, et qu'il n'est pas possible de le compenser par exemple avec les hanches sans compromettre la stabilité du déplacement.

Lorsque le robot est en mouvement, la tête du robot n'est pas stabilisée. L'image présente

donc des niveaux de bruit et de flou importants, renforcés par l'effet de rolling shutter de la caméra. Cela implique que la fréquence des images utilisables est beaucoup plus faible que la fréquence théoriquement disponible sur la caméra. Pour contourner et compenser cet effet, on peut jouer sur plusieurs paramètres. Une première approche possible serait de prendre les images uniquement lorsque les deux pieds du robot sont au sol, pour que la tête ne soit pas en mouvement. Cependant la fréquence d'image s'en trouve trop réduite, puisque la fréquence typique de la marche du robot est de un ou deux pas par seconde. Une meilleure solution consiste à réduire la hauteur et la fréquence des pas du robot, ainsi que les vitesses maximales (en rotation et en translation). De cette manière, l'amplitude et la vitesse des mouvements de la tête sont réduits, sans compromettre la stabilité de la marche mais au prix d'une réduction de la vitesse du robot.

Pour détecter des images floues et éviter de calculer une orientation sur cette image, on peut aussi comparer le niveau de flou de l'image courante à celui de l'image de référence, qui est nette car le robot et sa tête sont à l'arrêt au début du mouvement. Un critère rapide pour cela consiste à comparer les valeurs maximales du Laplacien de chaque image. Plus simplement, on peut comparer le nombre points d'intérêt dans l'image courante avec celui de l'image de référence : puisque l'image change peu au cours de la rotation, une décroissance significative du nombre de points signifie que le niveau de flou est trop important.

3.3.1.2 Contrôle d'une translation

Pour effectuer une translation, la référence est simplement définie par l'image prise dans la position de départ de la base : on amène la tête du robot dans la même direction que la base. On réutilise le contrôleur de rotation pour maintenir le cap du robot, en changeant les consignes de la base. Dans ce cas, on a $\alpha_0 = 0$ et $\alpha_f = 0$ pour la tête, et pour la base $\theta_0 = 0$ et $\theta_f = 0$. On rajoute ensuite une vitesse de translation suivant l'axe X, en plus du contrôleur. Pour déterminer l'arrêt de la translation, on se base sur l'estimation de la distance parcourue par l'odométrie. Celle-ci est surestimée à cause des glissements, mais un peu plus précise que l'estimation de position directe.

L'inconvénient de cette approche est que contrairement au cas de rotation pure, les points doivent être à l'infini pour respecter les hypothèses de calcul de l'orientation d'image (voir paragraphe 3.2.1.2). Dans un environnement de bureau ou domestique, il est difficile de trouver des points respectant réellement cette hypothèse. En revanche, on peut tout de même approcher le comportement de points situés à une distance réduite tant que le changement d'échelle reste modéré. On peut également mesurer l'évolution de la qualité de l'orientation avec le taux de points aberrants du RANSAC. On adapte donc le contrôle en surveillant la qualité du calcul, et en rafraîchissant l'image de référence avant qu'elle ne se dégrade trop. À chaque rafraîchissement, on prend un nouvelle image qui contient des points situés plus loin et avec une échelle adaptée par rapport à la position courante du robot. On définit pour cela un taux seuil de points cohérents T_S, tel que la qualité est encore suffisante pour avoir un calcul d'orientation fiable, mais suffisamment basse pour éviter des rafraîchissements trop fréquents.

L'Algorithme 7 décrit la méthode utilisée. Elle consiste à arrêter le robot lorsque la qualité du calcul commence à se dégrader, avant qu'elle ne devienne inutilisable. Dès que le robot est à nouveau aligné avec la direction de départ, on rafraîchit l'image de référence, puis on donne

Algorithme 7 Algorithme de rafraîchissement de la référence pour la translation

Paramètres : T_S taux limite de points cohérents, $\dot{x}_{ref}$ vitesse de translation souhaitée, ε rotation maximale

Entrée : I_r image de référence, I image courante, $T(I, I_r)$ taux de paires cohérentes, D distance à parcourir.

Sortie : Translation en ligne droite.

Initialiser $I_r = I_0$ image de référence initiale.
On note d la distance parcourue.
$\dot{x} \leftarrow \dot{x}_{ref}$
while $d < D$ **do**
  $\dot{x} \leftarrow \dot{x}_{ref}$
  if $T < T_S$ **then**
    if $|\theta| < \varepsilon$ **then**
      $I_r \leftarrow I$ (renouvellement référence)
    else
      $\dot{x} \leftarrow 0$ (arrêt translation)
    end if
  end if
  if $|\theta| > \varepsilon$ **then**
    $\dot{x} = 0$ (arrêt translation).
  end if
  Ramener $\theta = 0$ à l'aide du contrôle en rotation de la paragraphe 3.3.1.1.
end while

à nouveau une vitesse en translation au robot. Cela permet de contourner l'inconvénient du changement d'échelle, au prix toutefois d'une accumulation d'erreurs à chaque arrêt et d'interruptions régulières de la marche. L'avantage de cette approche est que la fréquence de rafraîchissement est adaptative, en fonction des points visibles et de l'environnement. Par exemple, il est tout à fait possible qu'une seule image de référence suffise à la totalité de la trajectoire, ou que le robot soit capable d'évoluer dans un environnement changeant. Le critère d'un seuil sur la qualité est ici assez simple, mais il peut avantageusement être raffiné. Par exemple on pourrait utiliser une moyenne glissante sur les valeurs de qualité, ou bien en filtrer le signal pour détecter des baisses significatives. Le but ici est de trouver un compromis entre robustesse et rapidité de déplacement du robot, même si en pratique les phases d'alignement pur sont très brèves.

3.3.1.3 Atteindre une position générique

On peut donc contrôler une trajectoire d'un robot en contrôlant le cap avec précision. Pour atteindre une position cible donnée (x, y, θ), on décompose le mouvement en trois étapes successives, pour que le mouvement soit uniquement constitué de rotations pures ou de translations sur l'axe X. On décompose le mouvement en une première rotation θ_0 dirigée vers la position cible, une translation d jusqu'à la position cible, et une deuxième rotation θ_1 pour aligner le robot dans

la direction désirée. La Figure 3.22 détaille ces notations. On a alors :

$$\theta_0 = atan(y,x)$$
$$d = \sqrt{x^2+y^2}$$
$$\theta_1 = \theta - \theta_0$$

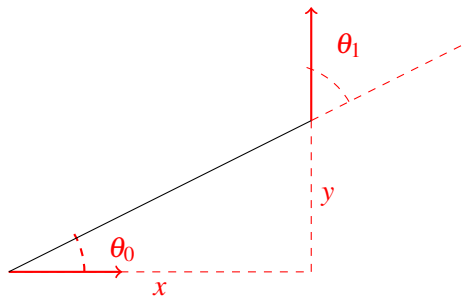


FIGURE 3.22 – Décomposition d'un déplacement en rotations et ligne droite

On perd donc la capacité du robot de se déplacer de façon holonome, mais en y gagnant une précision accrue, notamment en termes de cap. La précision de cette méthode en termes de cap est largement supérieure à l'odométrie, comme détaillé dans la section 4.3.1.

On peut utiliser ce contrôle de trajectoire pour effectuer les transitions d'un nœud à l'autre dans le graphe de localisation. Si les nœuds sont suffisamment proches relativement pour que l'information de positionnement soit fiable, alors on peut stocker dans les transitions l'estimation de l'odométrie (soit une position relative (x,y,θ)), et utiliser le contrôleur directement pour aller d'un nœud à l'autre. Pour rendre cette estimation plus fiable, on peut éventuellement recourir à une méthode de relaxation des contraintes sur les mesures d'odométrie du graphe, comme dans (Angeli et al., 2009). On peut aussi supposer que tous les déplacements du robot sont contrôlés par la boussole, y compris pendant l'apprentissage : la direction entre deux nœuds est donc connue avec précision, mais leur distance est connue approximativement.

Comme la distance reste mal connue, il est nécessaire d'avoir une façon de déterminer une condition d'arrêt de la translation. Pour cela, on pourrait arrêter régulièrement le robot pour procéder à l'identification du nœud en suivant la méthode décrite dans la section 3.2.5. Cela aurait l'inconvénient de ralentir fortement la progression du robot. Pour éviter cela, il faut une mesure de distance qui soit possible pendant la marche et ne nécessite pas forcément d'augmenter le champ de vision du robot. Cette mesure est obtenue à l'aide de la corrélation, décrite en section 3.3.2.

3.3.2 Mesure de l'éloignement par la corrélation

Comme décrit dans la section 3.2.2, un des buts de la localisation par corrélation d'image est de devenir robuste au flou et aux changements d'échelle. Ces deux caractéristiques sont particulièrement utiles pour effectuer des transition : elles sont nécessaires pour être encore utilisables alors que le robot est en mouvement.

Habituellement, la corrélation est sensible aux changements d'échelle. En effet le calcul repose sur le fait que le modèle de référence se retrouve à la même échelle dans l'image de requête. Il est donc nécessaire de se placer au préalable à cette échelle pour obtenir un pic de corrélation fiable et significatif. Le principe consiste à définir un facteur d'échelle donné α et à construire deux pyramides d'images parallèles à partir de l'image de référence d'une part et de requête d'autre part, tel que α est le facteur d'échelle pour passer d'une étape à l'autre. Cette double pyramide est visible dans la Figure 3.23. Plus α est proche de 1, plus la détermination de l'échelle sera précise. On note k le nombre d'étapes de chaque pyramide, indexées de 0 à k. Pour α, k donnés, la pyramide permet donc d'explorer des échelles d'images comprises entre $(\frac{1}{\alpha})^{k-1}$ et α^{k-1}. On prend ici par exemple $\alpha = 0.98$ et $k = 10$.

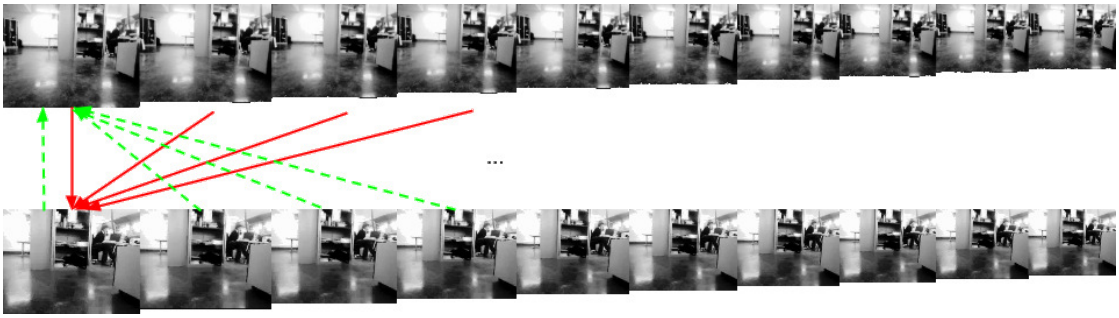


FIGURE 3.23 – Pyramide d'images pour la corrélation : référence en haut, requête en bas
Plateforme utilisée : NAO

On note R_i l'image d'index i dans la pyramide de référence, et I_i l'image d'index i dans la pyramide de requête. Pour rechercher la meilleure échelle, on utilise $R_0, ..., R_{k-1}$ comme référence en les comparant à I_0, et on calcule pour chaque étape le maximum de corrélation, s'il est valide (voir section 3.2.2 pour les critères de validité). Cette étape est représentée par les flèches rouges sur la Figure 3.23. Symétriquement, on utilise $I_0, ..., I_{k-1}$ comme référence en les comparant à R_0. On sélectionne ensuite l'échelle et la corrélation maximale pour l'ensemble des possibilités considérées. Cette recherche est décrite dans l'Algorithme 8.

Cette recherche permet d'être robuste aux changements d'échelle, et de calculer le facteur d'échelle idéal, à condition que le coefficient α soit suffisamment fin. En effet, si l'image de requête I_0 contient R_0 à une échelle inférieure à 1, alors il faut réduire R_0 pour l'y retrouver, et donc une des images de la pyramide de référence aura une valeur de ZNCC haute et significative. Inversement, si I_0 contient R_0 à une échelle supérieure à 1, alors en réduisant I_0 on se ramène à la même échelle que R_0, et donc une des images de la pyramide de requête aura un coefficient de corrélation fort.

On utilise l'image issue d'une pyramide comme référence plutôt que l'image originale pour des raisons pratiques. En effet le modèle de référence doit être plus petit que l'image de requête pour évaluer une ZNCC donnée, ce qui a plus de chances de se produire si on sélectionne l'image de la pyramide. A priori, même si les images à comparer n'ont pas de raisons d'avoir les mêmes tailles, elles ont des dimensions du même ordre, en particulier en hauteur.

Comme on calcule la corrélation sur deux pyramides séparées en inversant temporairement

Algorithme 8 Recherche de l'échelle et de la position de l'image de requête

Paramètres : k taille de la demi pyramide, α facteur d'échelle entre deux étapes, $V(C)$ fonction de validité d'une matrice

Entrée : $R_0, ..., R_k$ pyramide référence, $I_{-k+1}, ..., I_{-1}$ pyramide image requête
Sortie : C matrice de corrélation, f facteur d'échelle, $(x_{r,i}, y_{r,i})$ position du modèle dans la référence, (x_i, y_i) position du modèle dans la requête

```
On note 𝒞 l'ensemble des couples (corrélation, échelle) calculés.
for i ∈ [0, k[ do
    Calculer C(R_i, I_0).
    if V(C(R_i, I_0)) then
        𝒞 ← 𝒞 ∪ (C(R_i, I), α^i)
    end if
end for
for i ∈ ]−k+1, −1] do
    Calculer C(I_i, R_0).
    if V(C(I_i, R_0)) then
        𝒞 ← 𝒞 ∪ (C(I_i, R), α^i)
    end if
end for
On note c_max la valeur maximale de corrélation.
for (C_i, α^i) ∈ 𝒞 do
    if max(C_i) > c_max then
        c_max ← max(C_i)
        C ← C_i, f ← α^i
    end if
end for
```

requête et référence, la position du modèle dans R_0 n'est pas connue a priori. On note $(x_{r,i}, y_{r,i})$ la position du centre de la zone de corrélation dans R_i et (x_i, y_i) la position du centre de la zone de corrélation dans I_i. On note (x_r, y_r) et (x, y) leurs équivalents dans les images originales R et I. Alors on a :

$$(x_r, y_r) = \begin{cases} \alpha^{-i}(x_{r,i}, y_{r,i}) & \text{si } i > 0 \\ (x_{r,i}, y_{r,i}) & \text{sinon} \end{cases}$$

$$(x, y) = \begin{cases} (x_i, y_i) & \text{si } i > 0 \\ \alpha^{-i}(x_i, y_i) & \text{sinon} \end{cases}$$

La précision et la portée de cette méthode dépendent directement des paramètres α et k. Les critères de validité d'une corrélation entre deux images données sont ici essentiels pour limiter les fausses détections éventuelles qui nuiraient à la recherche de la meilleure corrélation. Comme la complexité pour une image carrée $n \times n$ est $O(n^2)$, alors la complexité de la nouvelle méthode

est :

$$\sum_{i=-k+1}^{k-1} (n\alpha^i)^2 = \frac{\alpha^{-k+1} - \alpha^k}{1-\alpha} n^2$$

soit une complexité exponentielle en k.

Cette technique peut être employée pour passer d'un nœud à un autre, en utilisant le calcul du facteur d'échelle pour quantifier l'éloignement au nœud suivant. Pour cela, on construit l'image de référence à partir des indications de position relative des nœuds : on utilise la direction du nœud cible et un angle d'ouverture donné pour la construire.

Cette zone de référence doit être visible depuis le nœud de départ. Dans ce cas, la corrélation obtenue doit donner un facteur d'échelle inférieur à 1, puisque le robot est situé plus loin que sa position cible. Il est intéressant de noter que cette approche est quantitative mais non métrique. En effet, elle dépend de la distance entre la position de référence du robot et l'objet de référence observé. Pour obtenir une information métrique, il faudrait par exemple les données de profondeur associées aux images. Ceci n'est possible que dans le cas de robots équipés de capteurs adaptés, mais cela pourrait constituer une extension possible.

La méthode peut être exécutée alors que le robot est en mouvement. Il est possible notamment de prendre l'image courante de la caméra comme image de requête, sans tourner la tête. Il suffit de prendre une image de référence plus large, pour compenser les mouvements latéraux de la tête par exemple. Des expériences ont montré que le suivi de l'image était robuste même en mouvement (voir section 4.3.2). Lorsque le robot avance vers la cible, il suffit alors d'utiliser l'image courante comme image de requête et de contrôler le facteur d'échelle obtenu. Lorsque le facteur d'échelle devient proche de 1, alors la cible a été atteinte. Cela permet d'obtenir un critère d'arrêt pour les translations d'un nœud à l'autre, comme décrit dans la section 3.3.1.

Il serait également possible d'utiliser ce suivi de corrélation pour contrôler la trajectoire de la même manière que la boussole. En effet, on obtient ici aussi l'orientation relative entre l'image de référence et de requête, si bien qu'il suffirait de réutiliser cette mesure dans les contrôleurs définis dans la section 3.3.1. On éviterait également d'avoir à modifier la marche, car cette méthode supporte des niveaux de flou et d'effet de rolling shutter plus important. De plus, l'image de référence pourrait être extraite du nœud cible plutôt que d'être prise à la volée. Cependant, calculer la corrélation est plus coûteux que d'extraire les points d'intérêt. Pour compenser cela, une extension possible serait de restreindre l'échelle de la pyramide pendant la progression du robot, pour explorer uniquement les échelles vraisemblables. Par exemple, il est inutile de considérer les échelles inférieures à 1 lorsque le robot se rapproche de la cible.

3.4 Conclusion

On a construit un système de localisation topologique. Les lieux sont représentés par des nœuds d'un graphe, et reliés par des transitions potentielles. Cette structure est particulièrement adaptée au cas de figure considéré, puisque les informations disponibles sont rares et partielles. Chaque nœud est construit à partir d'une série d'images prises de façon panoramique autour d'une position initiale du robot, à l'aide d'une caméra monoculaire et éventuellement d'une caméra 3D. On repère les pixels ainsi enregistrés par leurs coordonnées angulaires.

On peut alors identifier le nœud courant à l'aide d'une approche par sac de mots, basés sur des descripteurs ORB extraits des images. Une fois le nœud courant identifié, on l'utilise comme référence pour calculer des informations de localisation. On utilise alors une structure hiérarchique qui organise les différentes sources d'information par ordre croissant de complexité, de précision et de temps de calcul. Les résultats partiels de chaque source sont fournis aux suivant, de façon à donner une hypothèse préalable pour simplifier éventuellement les calculs.

— L'odométrie du robot constitue la base de cette structure, car elle est très rapide mais aussi soumise à une forte dérive et sensible aux perturbations.
— La boussole visuelle permet d'estimer l'orientation du robot en se basant sur des appariements de points d'intérêt, sous des hypothèses simplificatrices qui diminuent sa précision avec la distance à la position initiale.
— La méthode par corrélation permet de retrouver un modèle tiré des plusieurs images du nœud dans les images courant. On en déduit à la fois l'orientation du robot et une notion de facteur d'échelle. On utilise les hypothèses d'orientation pour restreindre le modèle et la région courante de recherche.
— Un recalage par ICP de profil en deux dimensions extraits de nuages de points 3D permet de calculer une position du robot. Le recalage utilise l'orientation calculée précédemment pour améliorer la convergence.

On fusionne ensuite les résultats avec un formalisme probabiliste simple. Cette structure permet d'obtenir des meilleurs résultats plus rapidement, et est suffisamment flexible pour pouvoir être étendue facilement. On arrive donc à rassembler suffisamment d'informations pour avoir une localisation utilisable.

Pour effectuer les translations entre les nœuds, on met en place un contrôle de trajectoire basé sur la boussole visuelle. Celle-ci permet de connaître et donc de contrôler précisément l'orientation du robot, même en mouvement. On l'utilise donc pour effectuer des rotations précises et des translations en ligne droite. Ce contrôle peut donc être utilisé pendant la construction de la carte et ensuite pour effectuer des translations d'un nœud à l'autre, en se servant de l'information de position relative d'un nœud par rapport à l'autre. La mesure du facteur d'échelle de la corrélation peut également servir de critère d'arrêt.

Le chapitre 4 présente des résultats expérimentaux de l'application des algorithmes présentés dans ce chapitre.

CHAPITRE 4

Résultats expérimentaux et validation

Ce chapitre est consacré à la présentation de différents résultats expérimentaux qualitatifs et quantitatifs. La partie 4.1 décrit les dispositifs expérimentaux employés. La partie 4.2 détaille les résultats de la localisation par rapport à un nœud, et la partie 4.3 ceux des transitions. Enfin, la partie 4.4 décrit les applications de la localisation et leur intégration sur les robots d'Aldebaran.

4.1 Dispositifs expérimentaux

Cette section décrit les dispositifs expérimentaux employés pour qualifier les algorithmes de localisation et de navigation. Ils comprennent à la fois une partie effectuée sur simulateur (section 4.1.1) et une partie effectuée dans un environnement réel (section 4.1.2). La première est parfaitement contrôlée mais aussi nettement plus simple qu'un environnement réel, mais il est plus difficile d'avoir une vérité terrain précise dans le cas de la deuxième.

4.1.1 Utilisation du simulateur Webots

Comme décrit dans la section 1.2.4, on dispose d'un simulateur qui comprend toutes les plateformes d'Aldebaran. On a construit un environnement virtuel de test. Celui-ci représente un environnement intérieur de façon schématique, avec une seule pièce qui contient des modèles d'objets du quotidien (tables, chaises etc). Pour obtenir des informations visuelles, on rajoute un certain nombre d'images texturées sur les murs de la pièce, comme des posters ou des tableaux. Les murs sont situés à une distance minimale de 2m et maximale de 5m de la position de départ du robot, située au milieu de la pièce. Ce monde virtuel est visible dans la Figure 4.1.

Le simulateur dispose d'un superviseur, qui est capable de connaître la position réelle du robot. Les expériences réalisées sur simulateur disposent donc toutes d'une vérité terrain exacte.

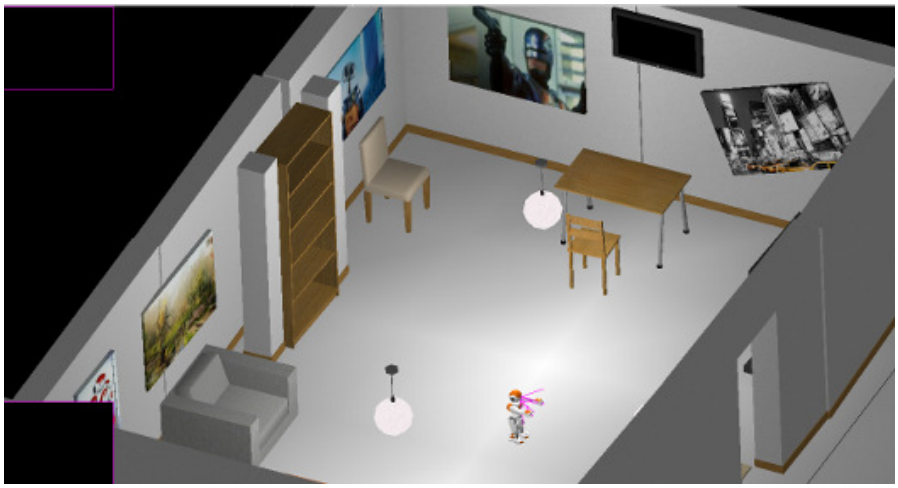


FIGURE 4.1 – Monde virtuel sous Webots utilisé pour les expérimentations

En contrepartie, les images couleur obtenues ne sont pas sensibles au flou ou aux mouvements du robot, et le signal 3D n'est pas bruité. Même s'il est possible d'introduire artificiellement un bruit sur les capteurs, les conditions ne sont pas aussi difficiles que dans un environnement réel dynamique. L'odométrie, à la fois sur NAO et sur Pepper, présente une dérive très forte. Cette erreur est due aux imprécisions de la modélisation des contacts par le moteur physique de simulations ODE utilisé par Webots.

Les expériences réalisées sous Webots servent de validation théorique et pour effectuer des tests à long terme sans risque pour le robot ou l'environnement. Les données recueillies sont enregistrées puis représentées a posteriori.

4.1.2 Acquisition d'images en conditions réalistes

En plus des expériences effectuées sur simulateur, on acquiert une base de données importante de nœuds pris dans un environnement réel. Ces données ont été acquises sur des robots dans un contexte réaliste : les personnes présentes n'avaient pas de consignes particulières, et étaient libres de se déplacer entre les acquisitions ou même pendant l'acquisition.

Les acquisitions sont réalisées par séries. Pour chaque série, on acquiert des nœuds espacés d'environ 0.5m autour d'un nœud central, sur une zone d'environ 1.5m de rayon (adaptée selon la configuration de l'environnement). Pour des raison pratiques, l'orientation de départ choisie est constante. La vérité terrain est construite de façon approximative, c'est-à-dire que le robot est déplacé à la main pour atteindre approximativement la position voulue. La Figure 4.2 montre un exemple d'une telle série : le nœud central est indiqué en rouge et les autres en bleu, avec l'orientation de départ indiquée par une flèche.

Des exemples de données acquises sont présentés dans l'Annexe B. Les nœuds ont été acquis dans plusieurs contextes différents :

— à l'étage R&D d'Aldebaran Robotics (Figure B.1) . Il s'agit d'un environnement de bureaux, de type *open space*, c'est-à-dire qu'on y trouve au maximum deux murs visibles

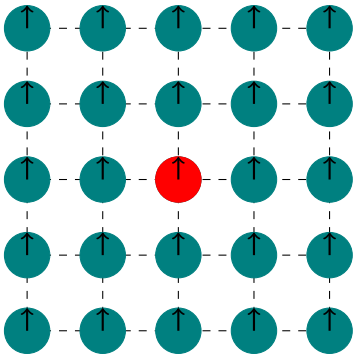


FIGURE 4.2 – Structure d'une série d'acquisitions de nœud

par la caméra 3D. L'environnement est texturé, et contient des éléments mobiles (des personnes et des chaises). Ce cas de figure est donc utile pour tester la robustesse dans des conditions d'environnement dynamique ;

— dans la salle de tests des applications d'Aldebaran Robotics (Figure B.2) . Cette salle est conçue pour tester les robots dans des conditions variées d'illuminations. Les murs sont texturés seulement par endroits (présence de posters sur les murs, murs blancs sinon), et certaines parties comportent des miroirs, y compris des miroirs situés face à face. Ces séries permettent donc de qualifier la robustesse dans des environnements difficiles ;

— dans des boutiques à Tokyo (Figure B.3). Ces expérimentations ont été réalisées de nuit dans une boutique dont les ouvertures sur l'extérieur avaient été masquées car le projet était à ce moment confidentiel. Il ne s'agit donc pas exactement des conditions réelles, car il n'y a pas de clients en mouvement, mais l'environnement est exactement le même, et les conditions d'éclairage quasiment identiques à une utilisation normale (puisqu'en intérieur il s'agit surtout de lumière artificielle). Les acquisitions ont été réalisées à plusieurs étages d'une boutique, située dans le quartier de Shibuya à Tokyo.

Chaque série a pour but de tester la robustesse des algorithmes de localisation dans différentes conditions. Pour cela, on utilise un système de tests automatiques qui suit le protocole suivant. Pour chaque série de nœuds, on prend le premier nœud comme référence, puis on tente de localiser les nœuds de la série par rapport à ce nœud en simulant une acquisition à la volée. On teste deux types de scénario : dans un premier temps, on réalise la localisation dans le cas du robot capturé, c'est-à-dire sans indice de position préalable. Dans un second temps, on calcule la localisation avec une hypothèse approximative issue de la position du nœud dans la série. On applique du bruit et une incertitude à cette hypothèse pour simuler une information provenant de l'odométrie. Après chaque calcul de localisation, on vérifie que la localisation obtenue est suffisamment proche de la vérité terrain. Comme la vérité terrain est approximative, on choisit des seuils de l'ordre de 0.2m en position et 10° en orientation. On peut donc présenter les résultats sous la forme suivante : succès du calcul, orientation suffisamment précise, position suffisamment précise.

Pour représenter et quantifier les résultats, on enregistre les données, puis on rejoue l'algorithme en différé sur un ordinateur séparé. Cependant, les algorithmes sont strictement identiques

à ceux exécutés sur le robot réels : les paramètres sont dimensionnés pour être utilisables sur le robot, en particulier les itérations de l'ICP ou du RANSAC.

4.2 Localisation au sein d'un nœud

Cette section est dédiée aux résultats de la localisation. On considérera d'abord la localisation de façon indépendante pour chacune des sources de localisation, puis on étudiera l'influence de la structure en cascade sur le résultat global. On s'intéressera également aux résultats d'identification des nœuds par sac de mots.

4.2.1 Estimation de l'orientation par la boussole visuelle

Cette section traite les résultats de la boussole visuelle décrite en section 3.2.1. On évalue d'abord la précision de la méthode à l'aide du simulateur Webots, puis on s'intéresse à des résultats typiques de calcul de l'orientation, et enfin aux performances de l'algorithme en embarqué.

4.2.1.1 Précision : résultats sous simulateur

Pour estimer la précision du calcul de la boussole, on effectue l'expérience suivante : sous le simulateur Webots, on fait effectuer au robot une série de rotations sur place. Ces rotations sont constituées de plusieurs rotations incrémentales de 30°, espacées de 10 secondes chacune. Pendant ces rotations, on échantillonne régulièrement les valeurs d'angles données par le superviseur du simulateur, qui correspondent donc à la vérité terrain, ainsi que les valeurs de la boussole et de l'angle estimé par l'odométrie du robot. L'expérience dure au total 15 minutes.

La Figure 4.3 montre l'évolution des valeurs d'angles échantillonnées au cours du temps. Les angles sont donnés dans l'intervalle $]-\pi,\pi]$ pour plus de lisibilité, si bien que les discontinuités observées correspondent en réalité à l'effet d'un modulo. La vérité terrain donnée par le superviseur est indiquée en bleu (*Ground truth*) : on voit qu'elle évolue bien de façon périodique au cours du temps. Les valeurs d'angles estimées par l'odométrie sont indiquées en rouge, et celles de la boussole en vert. On voit bien que les valeurs d'angles de l'odométrie dévient progressivement, au fur et à mesure que les erreurs s'accumulent, tandis que les valeurs de la boussole restent dans un voisinage très proche de la vérité terrain.

La Figure 4.4 représente l'évolution de l'erreur d'estimation au cours du temps. L'erreur de l'odométrie est représentée en vert, et celle de la boussole en bleu. On voit ici aussi que l'erreur de l'odométrie augmente de façon linéaire avec le temps, ce qui correspond à l'accumulation des erreurs d'estimation de la rotation avec le temps. L'erreur d'odométrie dans le simulateur est due aux glissements du robot sur le sol : elle est systématique car le sol sous le robot est d'aspect constant. En revanche, l'erreur de la boussole a une amplitude bornée, de l'ordre de 0.2 radians soit 3°. On remarque notamment l'absence de dérive avec le temps, contrairement à l'odométrie.

On voit donc qu'on peut non seulement obtenir une précision d'estimation très satisfaisante, mais aussi s'affranchir des dérives de l'odométrie, même si celle-ci est très marquée comme dans le cas du simulateur. Cependant, cette estimation est limitée ici à une zone restreinte autour du nœud d'origine, puisque le robot effectue des rotations sur place.

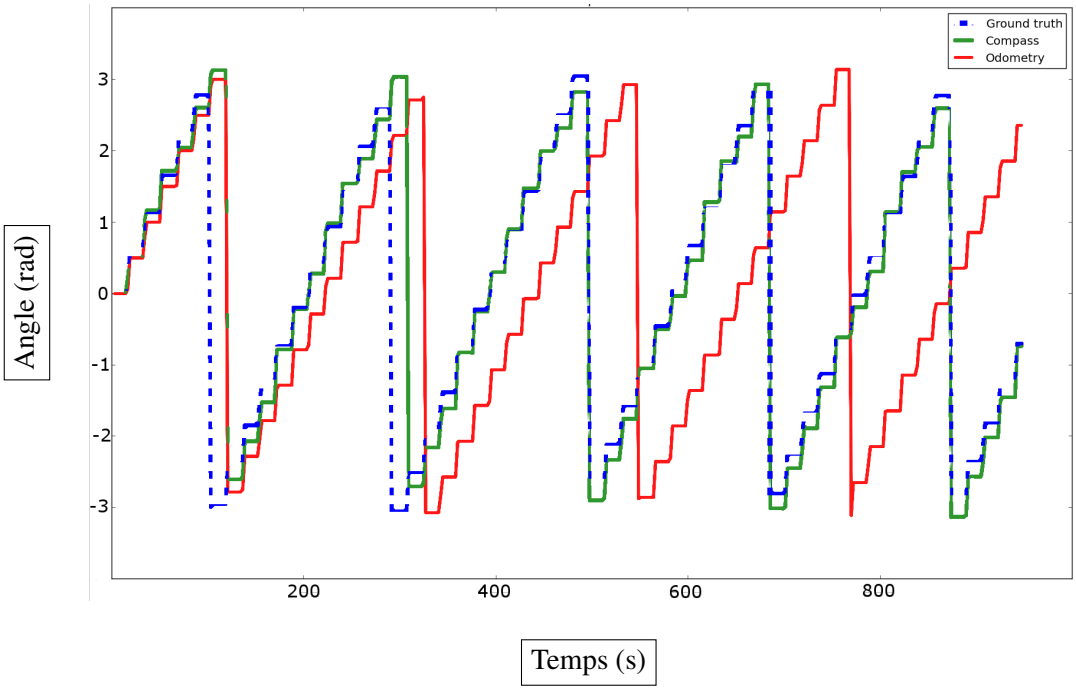


FIGURE 4.3 – Évolution des angles estimés par l'odométrie (en rouge) et la boussole (en vert) par rapport à l'angle réel (en bleu) au cours d'une expérience de rotation sous Webots

4.2.1.2 Résultats typiques en essais réels

On présente ici quelques résultats typiques en conditions réelles de la boussole visuelle dans différents cas de figure. Ces résultats sont obtenus à partir des séries d'images décrites dans la section 4.1.2.

La Figure 4.5 montre des exemples d'appariements réussis entre des images de référence et des images prises pour une localisation. Malgré certains faux appariements, le RANSAC met en valeur des résultats cohérents et pertinents. Sur la Figure 4.5(a), la densité des points d'intérêt est très importante, notamment sur les fenêtres, qui sont couvertes de panneaux solaires avec des formes régulières et marquées. Il est intéressant de noter que certains appariements sur ces panneaux sont faux (entre la partie supérieure gauche de la référence et l'inférieure droite de la requête), probablement à cause de l'ambiguïté des motifs. Ils sont alors éliminés par le RANSAC car ils ne sont pas cohérents. Dans la Figure 4.5(b) les points d'intérêt appariés sont nettement moins denses. En effet l'environnement est assez peu texturé, en particulier les murs blancs au fond : la majorité des points se situent sur les tables. On remarque aussi la présence d'un miroir (sur le poteau central), qui ne perturbe pas les calculs du RANSAC : les points détectés dans l'image reflétée sont correctement appariés et intégrés au modèle.

La Figure 4.6 montre un exemple où l'image de requête est partiellement occultée par une personne présente dans le champ de vision du robot. On voit que sur l'image de référence, les points d'intérêts situés dans la zone que recouvre la personne sont appariés de façon aberrante, et considérés comme tels par le RANSAC. Les points correctement appariés sont situés en de-

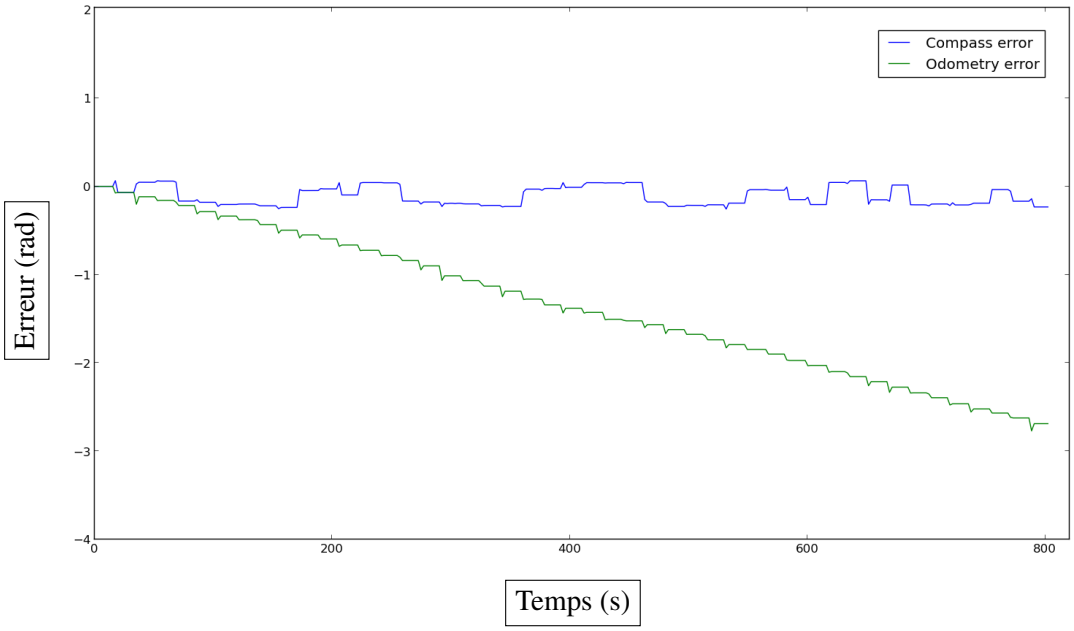


FIGURE 4.4 – Évolution de l'erreur par rapport à l'angle réel de l'odométrie (en vert) et de la boussole (en bleu) au cours d'une expérience de rotation sous Webots

hors de l'occultation et sont effectivement classés comme cohérents et intégrés dans le modèle. L'approche par RANSAC permet d'être robuste à ce genre d'occultations et d'erreurs : il existe suffisamment de points cohérents sur les côtés de l'image pour que le modèle final calculé soit correct. On remarque notamment que le taux de points incohérents est très important.

4.2.1.3 Performances

Le Tableau 4.1 montre l'influence de la résolution sur la boussole visuelle. Les mesures sont effectuées sur un robot pour deux images typiques. On compare ici deux images proches (sans la recherche du meilleur appariement), et on mesure le nombre de points détectés dans chaque image, le nombre de points appariés et le temps d'exécution.

Résolution	Points référence	Points requête	Nombre d'appariements	Extraction (ms)	Appariement et RANSAC (ms)
VGA	903	704	137	44.6	123
QVGA	464	476	88	16.6	60.0
QQVGA	15	20	11	2.18	16.3

Tableau 4.1 – Influence de la résolution sur la boussole (en embarqué sur un robot)

Le nombre de points détectés diminue avec la résolution. Cela vient du fait que les textures les plus fines disparaissent lorsque la résolution baisse. Le nombre d'appariements en revanche

(a) Dans l'environnement R&D (b) Dans la salle de tests

FIGURE 4.5 – Exemple d'appariements entre images de référence et images courantes
Plateforme utilisée : Pepper

représente toujours approximativement 25% du nombre de points trouvés. Il s'agit donc des appariements uniques, selon le critère décrit dans la paragraphe 3.2.1.2. À la résolution minimale (80x60), le nombre d'appariements devient très faible, probablement trop faible pour être robuste à des erreurs. Comme on a affaire ici à un cas facile, tous les appariements sont corrects et donc considérés comme cohérents par le RANSAC. Le temps d'exécution diminue fortement avec la résolution. Cette diminution n'est pas exactement quadratique, en raison de la phase de RANSAC, mais les temps diminuent significativement. En pratique, on utilise la résolution QVGA pour la boussole visuelle en mouvement (voir aussi section 4.3.1) et la résolution VGA pour la localisation dans un nœud.

On a vu dans la paragraphe 3.2.1.3 que la recherche de l'image à apparier dans le nœud était simplifiée : on recherche un maximum local dans le nombre de points appariés plutôt que le maximum global. En théorie, cela expose au risque de sélectionner des images qui ne sont pas les plus appropriées. En pratique, des tests ont été réalisés pour vérifier l'impact de cette approximation. Pour cela, on prend un nœud de référence, et on simule une requête en sélectionnant une image du nœud, puis en recherchant le meilleur appariement. On teste la recherche dans les cas suivants :

— on indique une hypothèse d'orientation proche de l'orientation réelle ;
— on ne donne aucune hypothèse d'orientation ;
— on donne une hypothèse d'orientation fausse (diamétralement opposée à la position

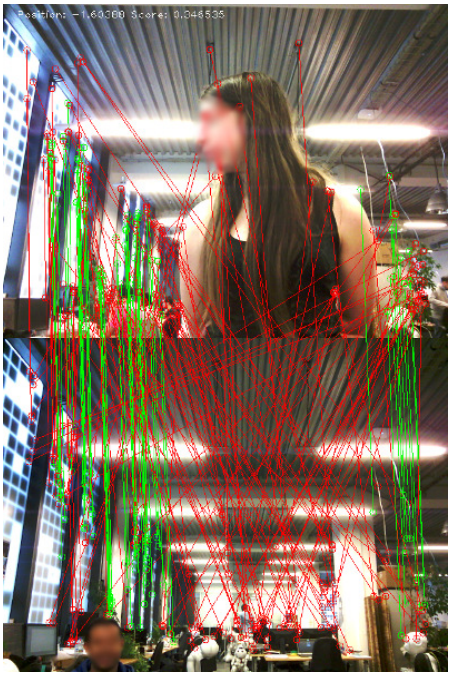


FIGURE 4.6 – Exemple d'appariement avec occultation de l'image de requête
Plateforme utilisée : Pepper

réelle).

Les deux derniers cas sont donc les plus difficiles. On effectue le test pour chaque image du nœud, et pour plusieurs environnements différents. On a ensuite plusieurs cas de figure :

— le maximum global et le maximum approché sont les mêmes ;
— le maximum global et le maximum approché sont différents, mais l'estimation d'orientation est correcte ;
— le maximum global et le maximum approché sont différents et l'estimation d'orientation est mauvaise ;
— les recherches de maximum ont échoués.

Le cas le pire est donc le troisième, puisque la recherche a induit la boussole en erreur.

Le Tableau 4.2 résume les résultats obtenus pour ces tests. Pour chaque condition de test (sans hypothèse : $\emptyset$, hypothèse proche : $\checkmark$ et fausse : $\times$), on recense en pourcentage les occurrences de chaque cas. Les tests portent sur trois types d'environnement : locaux R&D, boutique et salle de tests. Pour chaque environnement, on effectue le test sur trois nœuds, soit 48 par environnement. On voit ici qu'en pratique le cas où la recherche approximative donne un résultat faux est extrêmement rare, et se produit uniquement dans le cas d'une hypothèse complètement fausse. Parfois, l'image optimale n'est pas sélectionnée, mais l'orientation finalement calculée est correcte, si bien que l'impact final est négligeable (un peu moins de précision sur le calcul). On peut donc conclure que la recherche utilisée est une bonne approximation, et que le taux d'erreur reste suffisamment faible pour pouvoir par exemple être compensé par le calcul de l'orientation sur d'autres images.

Hypothèse	R&D			Boutique			Salle de tests		
	$\emptyset$	✓	×	$\emptyset$	✓	×	$\emptyset$	✓	×
Global = approx	89.6	100	93.7	97.9	97.9	95.8	93.7	97.9	97.9
Global $\neq$ approx, estimation juste	10.4	0.0	0.0	0.0	0.0	2.1	4.2	0.0	0.0
Global $\neq$ approx, estimation fausse	0.0	0.0	6.3	0.0	0.0	0.0	0.0	0.0	0.0
Échec recherche	0.0	0.0	0.0	2.1	2.1	2.1	2.1	2.1	2.1

Tableau 4.2 – Comparaison entre les approches exhaustive et approchée pour la boussole visuelle

4.2.2 Estimation de l'orientation par la corrélation

On s'intéresse ici aux résultats obtenus pour l'estimation de l'orientation par la méthode de corrélation. Les images sont présentées à la résolution effectivement utilisée par l'algorithme : on utilise pour chaque image une résolution QQVGA (80x60 pixels). Cette diminution permet à la fois de réduire le temps de calcul (qui est quadratique avec la taille de l'image) et d'être plus robuste aux changements à petite échelle (bruits sur la caméra, flou cinétique etc). La référence est située au dessus de la requête, et la matrice de corrélation est représentée en fausses couleurs. La valeur du maximum de corrélation et le facteur d'échelle associé sont indiqués en haut à gauche de l'image.

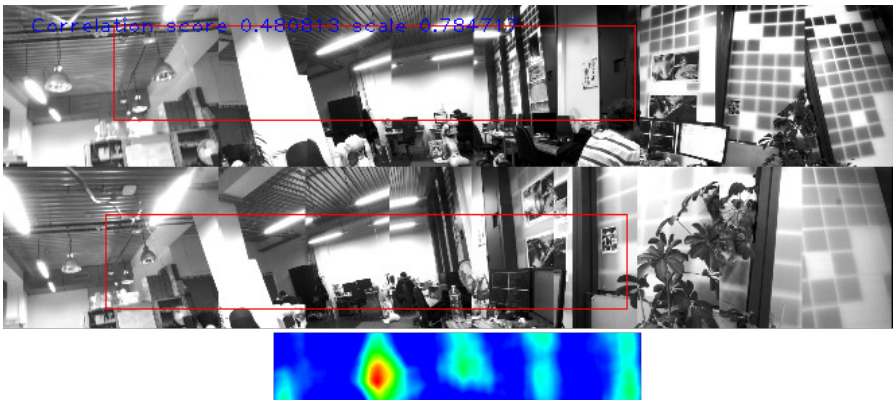


FIGURE 4.7 – Exemple typique de corrélation (avec facteur d'échelle)
Plateforme utilisée : Pepper

La Figure 4.7 présente un exemple typique de corrélation réussie entre un nœud de référence et une requête. Le modèle est replacé précisément dans la requête, et ce malgré un changement d'échelle important et un point de vue légèrement différent (notamment une rotation autour de l'axe Y).

Les Figure 4.8 et Figure 4.9 montrent des exemples de corrélation en présence de personnes. On voit que les différences ne perturbent peu ou pas le calcul de position, même si les per-

sonnes représentent une partie significative de la zone considérée et que l'environnement est assez pauvre en textures. Ceci est dû au fait que la corrélation est considérée de façon globale sur l'ensemble du modèle.

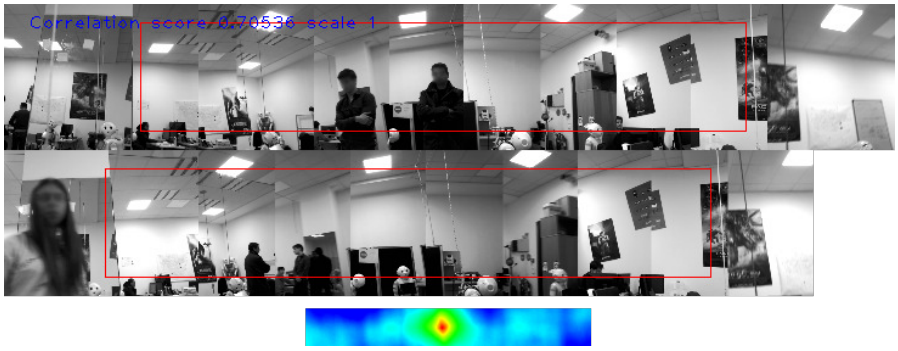


FIGURE 4.8 – Corrélation en présence d'obstacles (salle de tests)
Plateforme utilisée : Pepper

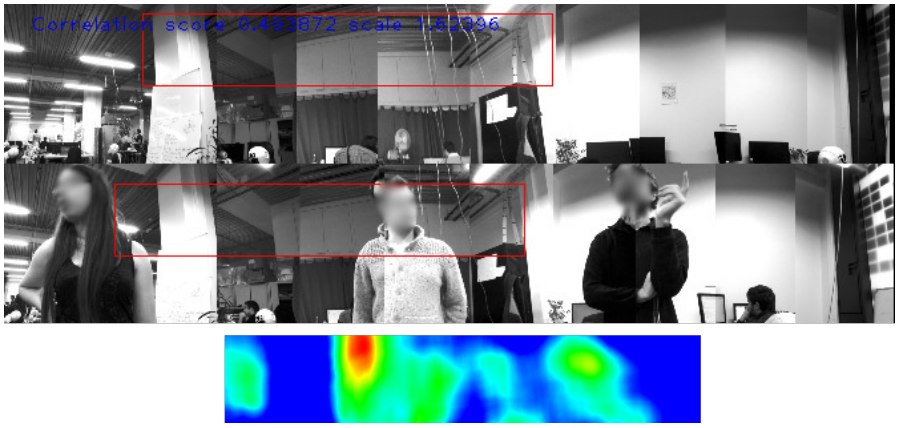


FIGURE 4.9 – Corrélation en présence d'obstacles (R&D)
Plateforme utilisée : Pepper

Sur la Figure 4.8, les obstacles représentent une proportion assez faible de l'image, et les points de vue sont très proches. On se trouve donc dans un cas simple mis à part la présence des obstacles. La matrice de corrélation présente un pic bien marqué, sans ambiguïté, ce qui montre que la présence de personnes supplémentaires n'a pas gêné le calcul. Pour la Figure 4.9, le résultat est moins clair. En particulier, le pic de corrélation est moins marqué que pour le cas de figure précédent, et la corrélation sur-estime le facteur d'échelle, en manquant notamment d'inclure une partie du modèle de base à gauche. La position est cependant satisfaisante.

Il arrive que la corrélation calculée soit erronée, malgré les critères de validité décrits dans la section 3.2.2. La Figure 4.10 montre un exemple de fausse corrélation, qui remplit pourtant ces conditions : valeur de corrélation maximale raisonnable et matrice de corrélation présentant un pic marqué. Cette erreur est probablement due au fait qu'on considère ici la corrélation sans

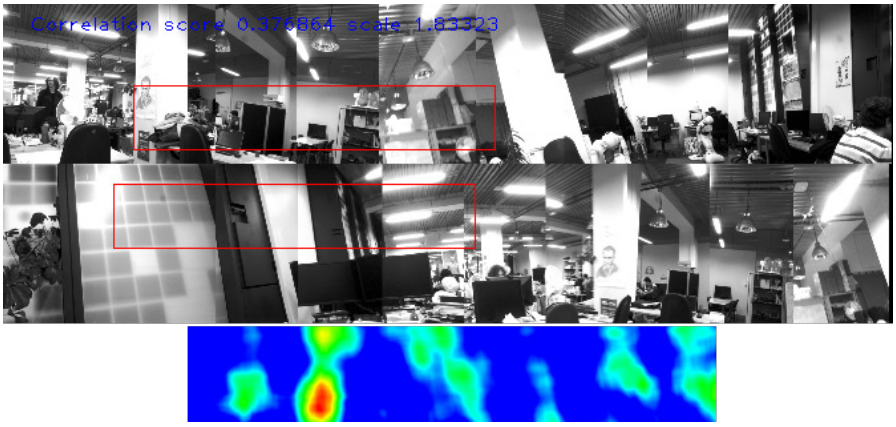


FIGURE 4.10 – Exemple de corrélation erronée
Plateforme utilisée : Pepper

indice préalable, sur une image demi-panoramique, si bien que la partie qui correspond effectivement est très réduite. En pratique, dans ce cas, l'orientation calculée du robot était finalement fausse, ce qui a empêché la convergence correcte de l'ICP, et a donc amené à compléter le nœud de requête. Dans cette seconde partie, la corrélation et la boussole ont calculé l'orientation avec succès (la Figure 4.7 correspond à la corrélation calculée sur la seconde partie).

4.2.3 Estimation de la position avec l'ICP

On s'intéresse ici à la méthode d'estimation de la position par ICP sur les profils 3D de l'environnement, décrite en section 3.2.3. On évalue d'abord la précision sur le simulateur Webots, puis on s'intéresse à quelques résultats typiques dans des situations communes, en particulier en cas d'occlusion, et enfin on considérera les performances de l'algorithme en temps de calcul et en taux de réussite.

4.2.3.1 Précision : résultats sous simulateur

Pour quantifier la précision du calcul de position par l'ICP, on réalise l'expérience suivante. On définit quatre positions cibles par rapport à un nœud donné, de coordonnées $A = (0,0), B = (0,1), C = (1,0)$ et $D = (-1,0)$. On fait alors cycler le robot entre ces quatre positions de la manière suivante : on calcule la position, puis on effectue en boucle ouverte à l'odométrie le déplacement vers la cible suivante correspondante. La position est estimée soit par l'odométrie, soit par la méthode d'ICP qui utilise l'angle de la boussole visuelle comme hypothèse préliminaire. Après chaque mouvement du robot, on enregistre les positions calculées par l'odométrie, l'ICP et le superviseur.

La Figure 4.11 présente les trajectoires obtenues : la trajectoire réelle du robot est représentée en vert, celle estimée par l'ICP en bleu, et l'odométrie en rouge. On peut noter que l'odométrie diverge très rapidement de la position réelle du robot, en raison de l'accumulation des erreurs sur le cap et sur les déplacements. Seule la forme des mouvements est conservée, mais les positions

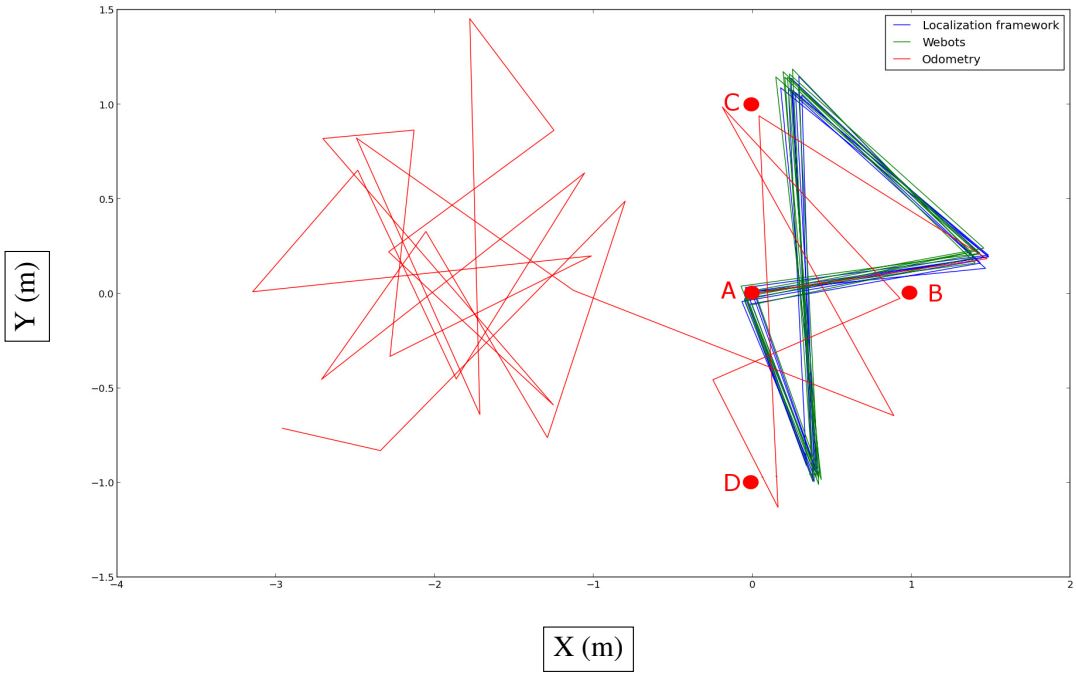


FIGURE 4.11 – Comparaison des trajectoires estimées par l'odométrie (rouge) et l'ICP (bleu) en comparaison avec la trajectoire réelle (vert)

réelles n'ont rapidement plus rien à voir avec les positions estimées. En revanche, l'estimation de l'ICP reste très proche de la position réelle, et ceci malgré un grand nombre de cycles, ce qui met en valeur l'absence de dérive.

On peut remarquer que les positions clés du cycle ne sont pas atteintes exactement. Ceci est dû au fait que même si la position est calculée à chaque étape, les transitions sont ici effectuées en boucle ouverte à l'odométrie. On voit clairement sur la figure cette incertitude sur chacune des transitions : on voit que l'estimation du cap du robot est biaisée et que les distances sont fausses (en particulier entre les étapes A et B). Malgré cela, la position calculée par l'ICP reste cohérente. On peut noter que les étapes sont répétables, ce qui est dû au fait que l'environnement est répétable dans le simulateur.

Il est important de noter que les données 3D du simulateur ne sont pas bruitées ou déformées comme elles le sont sur le robot réel. En particulier, le signal ne se dégrade pas avec la distance, et le profil garde une forme constante quelle que soit la distance réelle de l'environnement. On élimine donc une source très forte d'incertitude et d'erreur.

4.2.3.2 Résultats typiques

La Figure 4.12 et la Figure 4.13 présentent des exemples de recalage de profil avec l'ICP. On y représente quelques unes des itérations nécessaires à la convergence. Le profil de référence est indiqué en rouge, et la requête en cours de recalage en bleu et la position finale en vert. La première itération correspond au positionnement effectué avec l'hypothèse préalable d'orienta-

tion, calculée à l'aide de la boussole et de la corrélation. On n'utilise pas ici l'odométrie pour initialiser la convergence. L'échelle et la position des axes initiaux est indiquée pour le profil de référence (axe rouge pour X, vert pour Y).

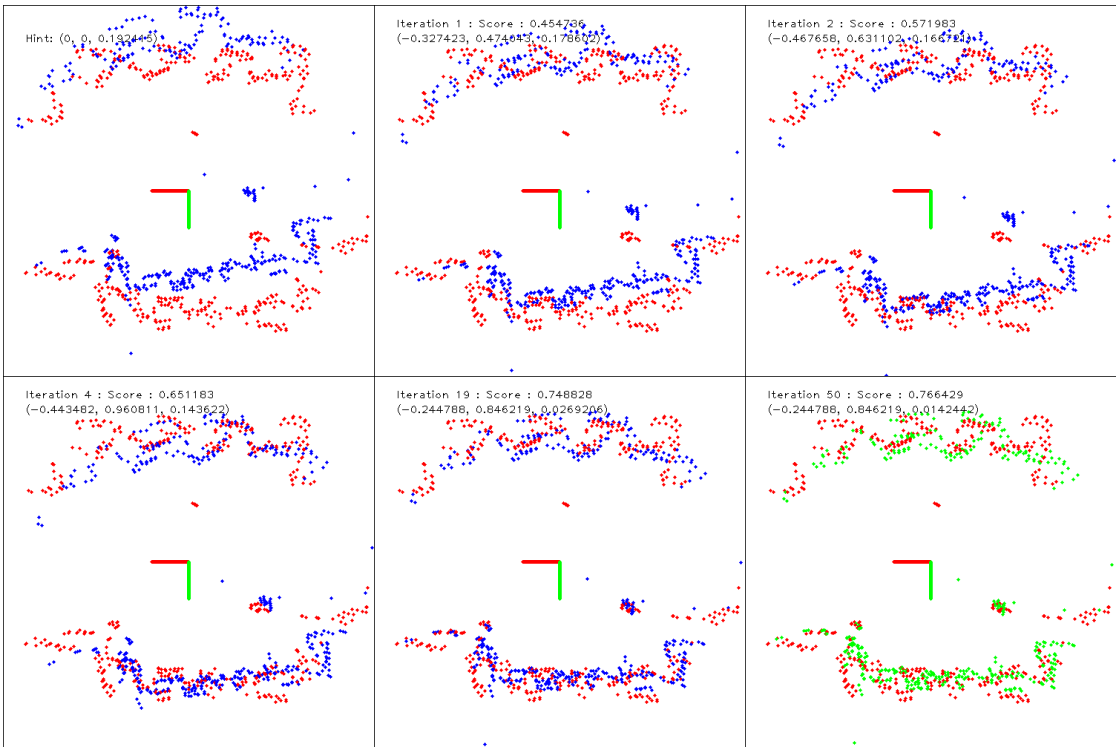


FIGURE 4.12 – Recalage d'un profil par ICP (référence : rouge, alignement en cours : bleu, alignement final : vert)
Plateforme utilisée : Pepper

Dans la Figure 4.12, on voit que l'angle à l'initialisation est quasiment parfait. On voit également que le robot s'est rapproché du mur visible en bas de l'image. Étant donné que le bruit des lentilles dépend de la distance à l'obstacle réel, on peut voir notamment que la morphologie du mur sur le profil a changé : les ondulations dues au bruit ont une amplitude beaucoup moins importante. De même, l'amplitude du bruit sur le mur du haut est plus important. Ces changements rendent le recalage précis nettement plus difficile. Dans la Figure 4.13, on observe le même phénomène, notamment avec le mur du haut. On peut voir également les effets d'une perte de portée, puisque la partie inférieure droite du profil est absente sur la requête.

On obtient malgré tout un recalage satisfaisant. Étant donné le bruit sur le signal de la caméra 3D, on peut estimer la précision du recalage à 0.5m dans le pire des cas (en pratique, l'erreur moyenne se situe plutôt autour de 0.25-0.3 même dans le pire des cas).

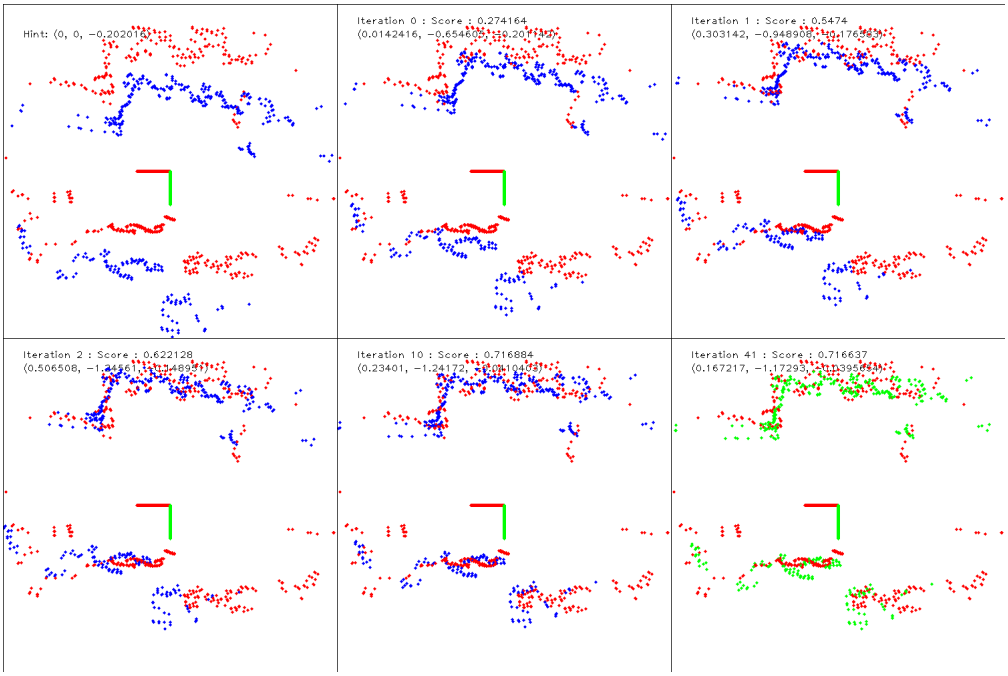


FIGURE 4.13 – Exemples de recalage par ICP d'un profil
Plateforme utilisée : Pepper

4.2.4 Structure en cascade

On s'intéresse ici aux résultats globaux. On montrera d'abord quelques exemples représentatifs des résultats en conditions réelles, puis on quantifiera les résultats obtenus dans les différents environnements et l'influence de la structure en cascade dans les performances.

4.2.4.1 Résultats typiques en conditions réelles

La Figure 4.14 montre un exemple de résultats pour un calcul de localisation dans l'environnement R&D. Les nœuds de référence et de requête sont visibles en Figure 4.14(a) et Figure 4.14(b) : entre les deux, le robot a avancé vers le rideau du fond. Les points d'intérêt sont mis en valeur en rouge. La localisation est faite sans hypothèse a priori (robot capturé).

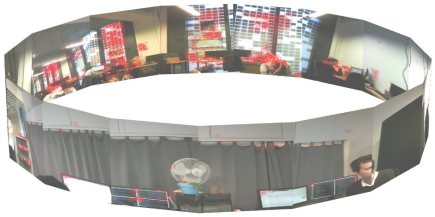
La Figure 4.14(c) montre les résultats de la boussole visuelle sur 9 images successives (qui correspondent à une première moitié de nœud). On voit qu'une seule image (la première) présente un faux appariement, qui sera éliminée comme aberrante. Les dernières images présentent assez peu de points d'intérêt, et auront donc un poids moins importants que les images plus riches.

La Figure 4.14(d) montre la corrélation obtenue entre deux images extraites des nœuds autour de l'angle indiqué par la boussole. On voit que malgré le facteur d'échelle important et le changement de point de vue, la corrélation s'effectue correctement.

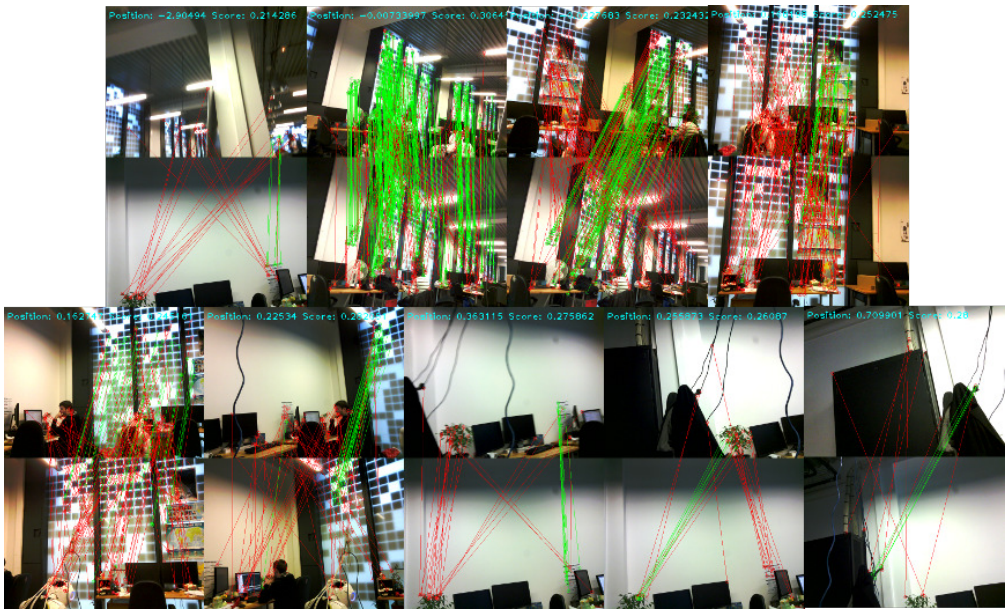
Enfin, la Figure 4.14(e) et Figure 4.14(f) illustrent la convergence de l'ICP. L'état initial est obtenu à l'aide de l'orientation calculée par les méthodes de boussole et de corrélation, mais sans

hypothèse sur la position du robot. La convergence s'effectue facilement, car les déformations sont modérées et l'environnement ne change pas significativement.

(a) Nœud de référence

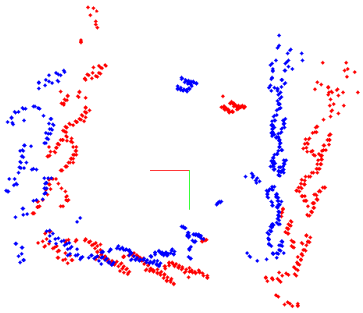


(b) Nœud de requête

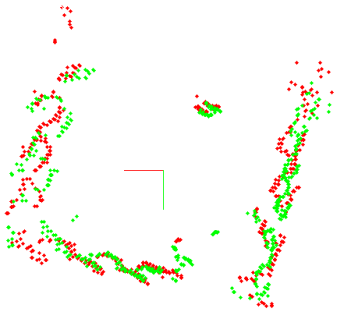


(c) Résultats de la boussole visuelle. Orientation estimée : 0.18 radians

(d) Résultats de la corrélation. Orientation estimée : 0.23 radians



(e) État initial de l'ICP (référence rouge, requête bleu), hypothèse orientation uniquement



(f) État final de l'ICP (référence rouge, requête vert). Position estimée : x : -0.785, y : 0.615

FIGURE 4.14 – Séquence de localisation dans l'environnement R&D. État final (x : -0.785, y : 0.615, θ : 0.19).
Plateforme utilisée : Pepper

La Figure 4.15 présente un autre exemple, réalisé dans la salle de tests. Les conditions sont identiques à l'expérience précédente (pas d'hypothèse préalable), ainsi que les résultats présentés (nœuds, boussole visuelle, corrélation et ICP).

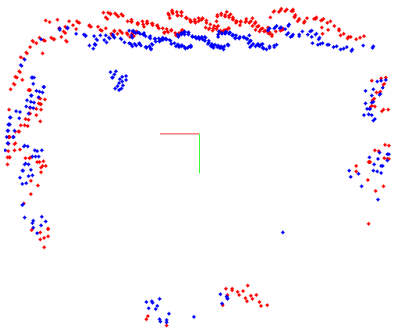
On peut constater ici que les miroirs ne sont pas gênants ni pour la boussole visuelle ni pour la corrélation. On peut observer aussi des personnes en mouvement (directement ou dans les miroirs), qui n'affectent pas les calculs d'orientation.

(a) Nœud de référence

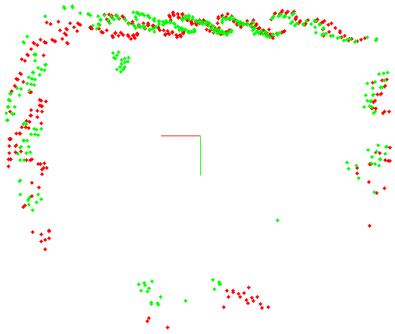
(b) Nœud de requête

(c) Résultats de la boussole visuelle. Orientation estimée : -0.10 radians

(d) Résultats de la corrélation. Orientation estimée : -0.07 radians



(e) État initial de l'ICP (référence rouge, requête bleu), hypothèse orientation uniquement



(f) État final de l'ICP (référence rouge, requête vert). Position estimée : x : 0.035 y : -0.445

FIGURE 4.15 – Séquence de localisation dans la salle de tests. État final (x : 0.035 y : -0.445, θ : -0.10)
Plateforme utilisée : Pepper

4.2.4.2 Influence de la structure en cascade

On examine ici l'influence de la structure en cascade dans le calcul de la localisation. Pour cela, on quantifie sur les séries d'images les résultats obtenus en supprimant progressivement certaines des sources de la structure.

Le Tableau 4.3 présente les résultats de tests effectués sur les 3 types de nœuds acquis (à l'étage R&D d'Aldebaran, dans la salle de tests, dans une boutique). On définit arbitrairement un nœud de chaque série comme référence. On calcule ensuite la localisation des nœuds de la série par rapport à ce nœud, en simulant une localisation normale. On commence donc par considérer la première moitié du nœud, puis la seconde si l'information est insuffisante. Pour chaque nœud de la série, on effectue deux fois le test en changeant le nœud de référence. Les tests effectués sont les suivants :

— on vérifie que le calcul de localisation se fait avec succès. En effet, tous les nœuds acquis sont théoriquement dans la zone de validité du nœud de référence ;
— si la localisation est réussie, on vérifie que la position calculée est suffisamment proche de la position indiquée en vérité terrain. Étant donné que cette position est approximative (mesurée rapidement à l'odométrie ou à la main pour des raisons de temps limité d'acquisition), les limites sont assez permissives, mais restent dans l'ordre de grandeur de la précision finale demandée : 50cm de rayon autour de la position supposée, et 10° autour de l'orientation.

Pour chaque type d'environnement, les résultats sont donnés en pourcentage sur le nombre total de tests. Les résultats donnés pour les deux derniers tests sont donc calculés sur les localisations réussies.

On réalise une série de tests pour chaque environnement, en rejouant les acquisitions :

— on localise le robot en simulant une mesure d'odométrie. Celle-ci est volontairement rendue imprécise et bruitée pour simuler les incertitudes de l'odométrie et éviter de tester un cas trop favorable ;
— on se place dans le cas du robot capturé : on effectue la localisation sans mesure d'odométrie, en utilisant uniquement les données images et 3D acquises ;
— on se place dans le cas du robot capturé, et on désactive artificiellement la boussole visuelle de la structure de calcul. La corrélation est donc la seule source qui permet de calculer l'orientation du robot.

Le but est de déterminer la robustesse de la méthode dans des environnements de catégorie assez différentes, et de confirmer l'importance de la structure hiérarchique.

Conditions	R&D			Salle de test			Boutique		
	Succès	$x-y$	θ	Succès	$x-y$	θ	Succès	$x-y$	θ
Avec hypothèse	96.8%	97.6%	94.4%	90.9 %	100%	100%	95.7%	94.6 %	82.6%
Sans hypothèse	95%	88.8%	93.6%	88.1%	96.7%	98.3%	85.7%	64.8%	79.1%
Sans boussole	52.2%	50%	52.2%	40%	38.3%	40%	50%	45.2%	50%

Tableau 4.3 – Comparaison des résultats sur différentes séries avec plusieurs conditions

On constate donc naturellement que les résultats sont meilleurs lorsqu'on utilise la totalité de la structure, avec une hypothèse approximative issue de l'odométrie. Notons que cette hypothèse est à la fois imprécise et non suffisante : pour que la localisation soit considérée comme réussie, il faut qu'au moins une des sources visuelles ait réussi, et que le recalage de profil soit suffisamment précis. On voit qu'on obtient dans tous les cas des taux très satisfaisants.

Dans les tests sans hypothèse, on constate une baisse de performance en ce qui concerne le calcul de la position du robot. Cette diminution est particulièrement nette dans le cas de l'environnement de boutique. On peut expliquer cela par la mauvaise qualité du signal 3D : dans cet environnement, on voit surtout des murs latéraux, dont le profil est déformé par les lentilles. À cause de ces déformations, le recalage est parfois ambigu ou difficile, même avec une initialisation correcte en orientation. De manière générale, on se situe dans le cas où l'initialisation de l'ICP se fait uniquement en orientation, ce qui augmente les risques de trouver uniquement un minimum local.

On constate qu'avec ou sans hypothèse, l'environnement le plus favorable est celui de l'étage R&D. En effet, il est à la fois texturé et présente des murs à une distance raisonnable pour le recalage de profils.

Si la boussole est désactivée, le taux de localisations réussies chute fortement. Cependant on peut voir que pour chaque localisation réussie, le calcul de l'orientation est exact. En effet, la corrélation a un taux de rappel (rapport entre les corrélations détectées et celles effectivement présentes) faible, pour éviter les fausses corrélations, mais une précision (rapport entre les corrélations correctes et les corrélations détectées) forte. Cette tendance se retrouve dans tous les environnements considérés. La corrélation, lorsqu'elle est réussie, est donc une source fiable d'informations. Ce taux de réussite est augmenté lorsqu'on fournit une hypothèse préalable pertinente, qui permet de restreindre le champ de recherche. On pourrait également dans ce cas se permettre d'utiliser des conditions de rejet moins contraignantes.

4.2.5 Identification du nœud courant par sac de mots

On s'intéresse ici aux résultats d'identification du nœud courant par sac de mots, décrite dans la section 3.2.5. Pour quantifier les résultats, on réalise l'expérience suivante : on acquiert une série de 12 nœuds. Ces nœuds correspondent à 6 emplacements différents dans l'environnement : pour chaque emplacement, on acquiert deux nœuds, avec des orientations initiales différentes et à des instants différents. Les nœuds sont indexés de telle façon que les nœuds pairs correspondent à un nouvel emplacement, et les nœuds impairs à une acquisition supplémentaire au même endroit. On utilise alors la moitié de ces nœuds (d'index pairs) pour entraîner un sac de mots en calculant un vocabulaire de référence, et l'autre moitié comme base de test (les nœuds d'index impairs). On construit alors un vocabulaire constitué de 10 000 mots à partir de la base d'apprentissage. La Figure 4.16 montre la disposition des différents nœuds. Ceux-ci sont espacés d'environ 2m. Les lignes relient les nœuds visibles réciproquement (par exemple, le nœud 4 est visible depuis le 0 et le 2). Cette expérience a été réalisée dans l'environnement R&D, et n'a pas été étendue à l'ensemble de la base de données par manque de temps.

On teste ensuite l'identification d'un nœud donné de la manière suivante : on apparie tous les descripteurs des points d'intérêt du nœud de requête avec les mots du vocabulaire. On calcule ensuite la somme des valeurs de TF-IDF pour chacun de ces mots du vocabulaire, et on apparie

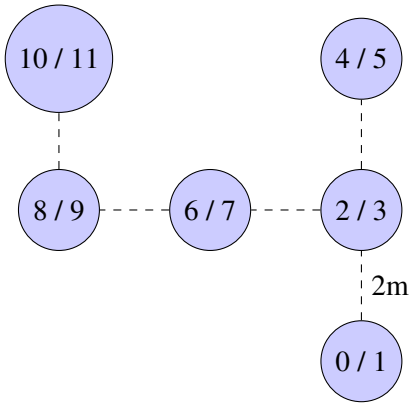


FIGURE 4.16 – Disposition des nœuds tests pour l'identification par sac de mots

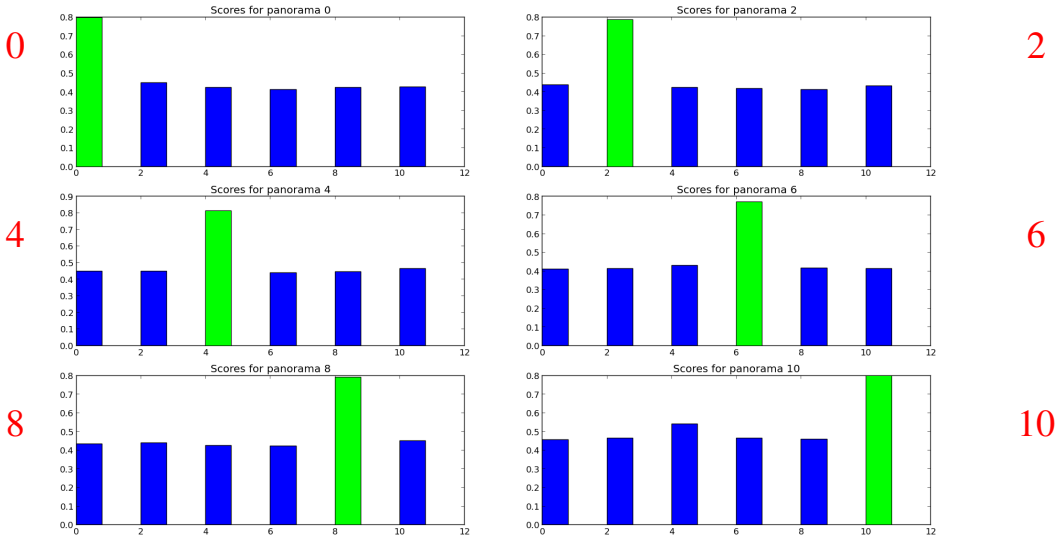


FIGURE 4.17 – Exemple de résultats d'identification sur la base de d'entraînement du sac de mots (10 000 mots)

le nœud de requête avec le nœud de référence présentant le meilleur score. On effectue ce test à la fois sur la base d'apprentissage, pour confirmer la pertinence du vocabulaire, et sur la base de test pour estimer la performance de la méthode. Il est intéressant de noter que certains nœuds sont assez proches des autres (en particulier la série 0-2-4), ce qui fait qu'on retrouve des points d'intérêt de l'un à l'autre. Cette configuration rend l'identification plus complexe.

Les résultats sont présentés de la manière suivante. Pour chaque index de nœud de requête, on construit l'histogramme suivant : les index des nœuds de référence sont indiqués en abscisse, et le score total de TF-IDF donne la hauteur de la barre. Pour plus de lisibilité, le maximum de TF-IDF est mis en valeur en vert. Les résultats pour chaque valeur possible de l'index de requête sont indiqués côte à côte.

La Figure 4.17 présente les résultats d'identification sur la base d'apprentissage. Les résultats

doivent donc être sans ambiguïté. Pour chaque index, le résultat attendu est que le maximum soit atteint pour l'index de référence identique. On voit ici sur la figure que c'est très nettement le cas : tous les scores maximaux représentent approximativement le double des autres scores. Cela montre que l'approche est suffisamment discriminante, malgré le fait que les nœuds soient acquis assez proches les uns des autres, à 2m de distance pour certains par exemple, si bien qu'au moins un autre emplacement est visible depuis chaque nœud.

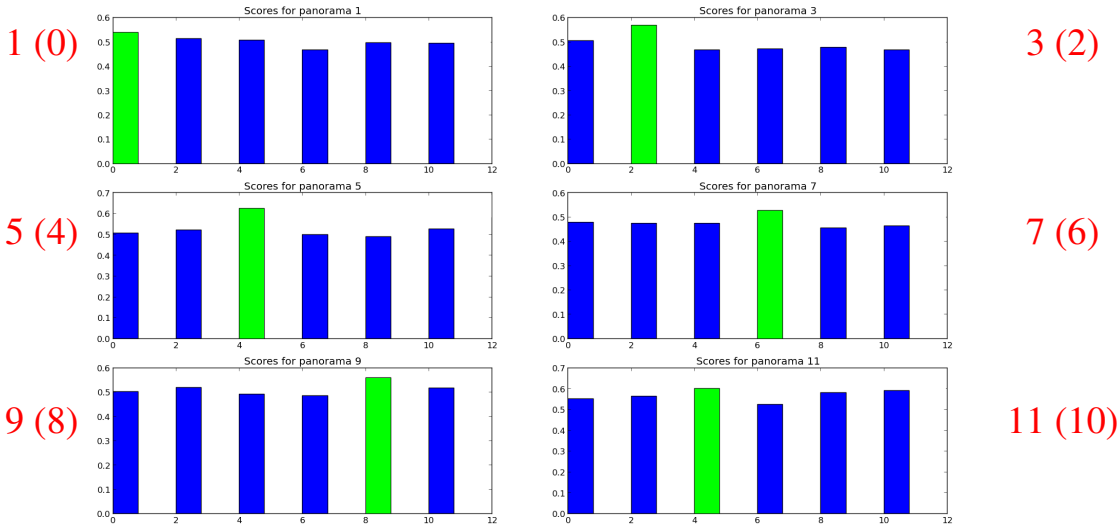
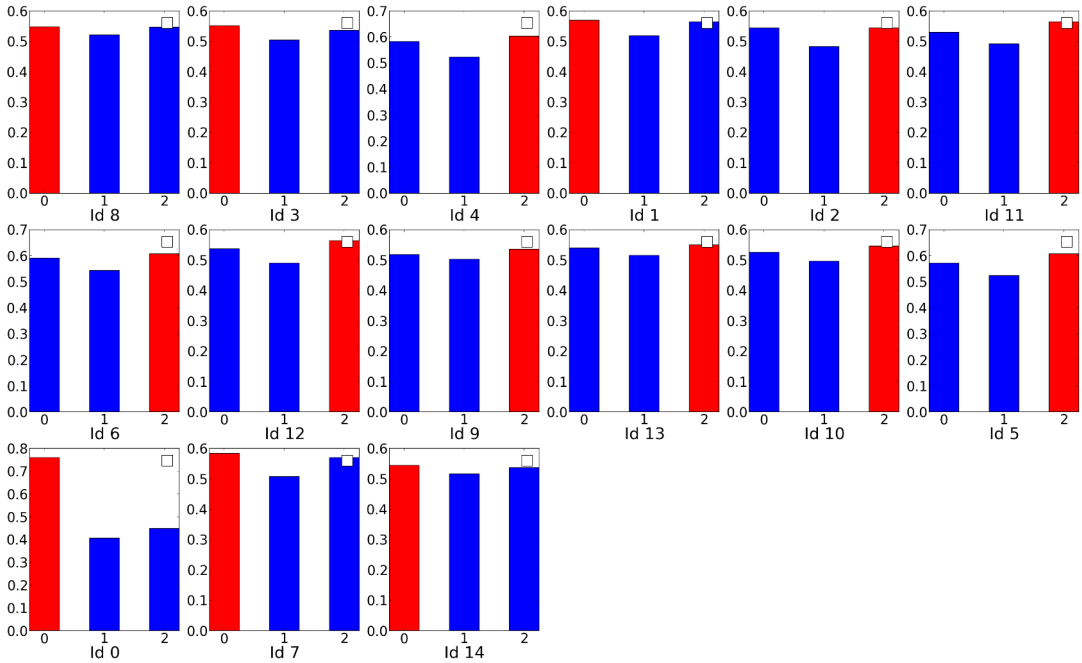


FIGURE 4.18 – Exemple de résultats d'identification sur la base de tests du sac de mots (10 000 mots) : identifiants et références correspondantes

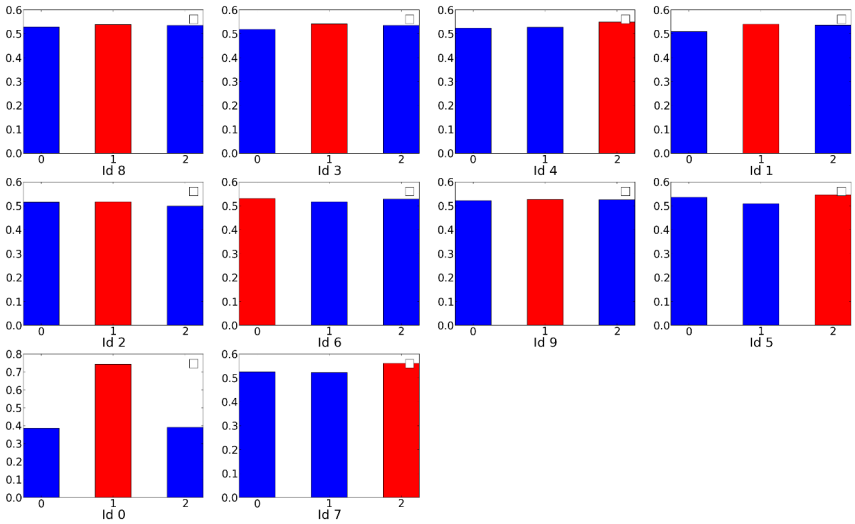
La Figure 4.18 présente les résultats d'identification sur la base de tests. Les nœuds de requête ne sont pas identiques aux nœuds de référence : comme ils n'ont pas été acquis au même moment, et qu'aucune consigne n'a été donnée pour que les gens restent sur place, on peut observer des différences non négligeables dans l'environnement, en particulier au niveau des personnes présentes. On voit que les maximums de score correspondent bien aux index attendus, même s'ils sont nettement moins marqués que pour la base d'apprentissage : pour chaque index impair de nœud de requête $2k+1$, le maximum correspond au nœud d'index $2k$, à l'exception du nœud 11, qui est lui identifié au nœud 4. A noter que le nœud 10 correspond au deuxième meilleur score d'appariement, ce qui laisse penser qu'en ajoutant par exemple des tests supplémentaires pour vérifier les meilleurs scores, il aurait été possible d'identifier le bon nœud de référence.

On a effectué des tests similaires sur la base expérimentale. Dans chaque type d'environnement, les nœuds ont été acquis par série : chacune correspond à une zone d'environ 2m de diamètre autour de la position initiale (et ne contient pas nécessairement le même nombre de poses). Les nœuds sont acquis selon le modèle de la Figure 4.2. Pour construire le vocabulaire, on utilise le premier nœud de chaque série comme référence, puis on le teste en tentant d'identifier chaque nœud par rapport aux origines des séries. En principe, les nœuds devraient être appariés à la série correspondante. On identifie les nœuds série par série (y compris celui qui a

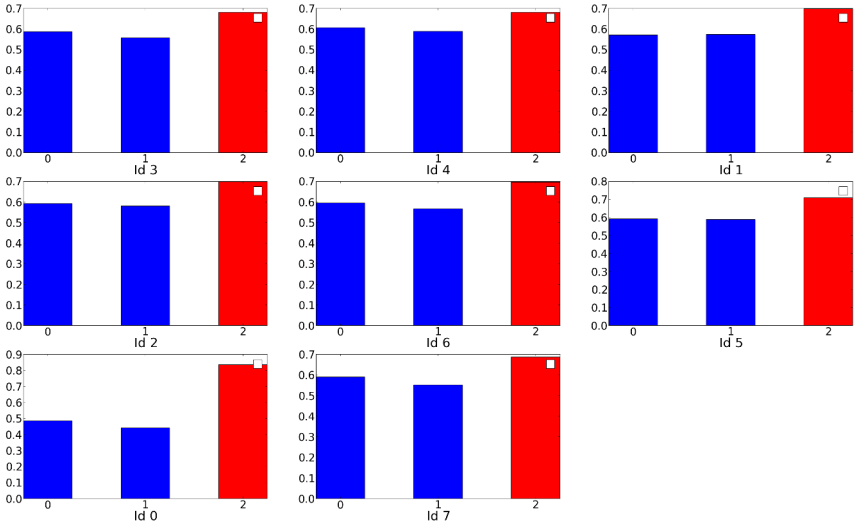
servi à construire le vocabulaire). Pour chaque nœud, on indique les scores pour les trois possibilités et le score maximal est mis en valeur en rouge. Pour que le nœud soit correctement identifié, le score maximal dans correspondre à l'index de la série.



(a) Série 0



(b) Série 1



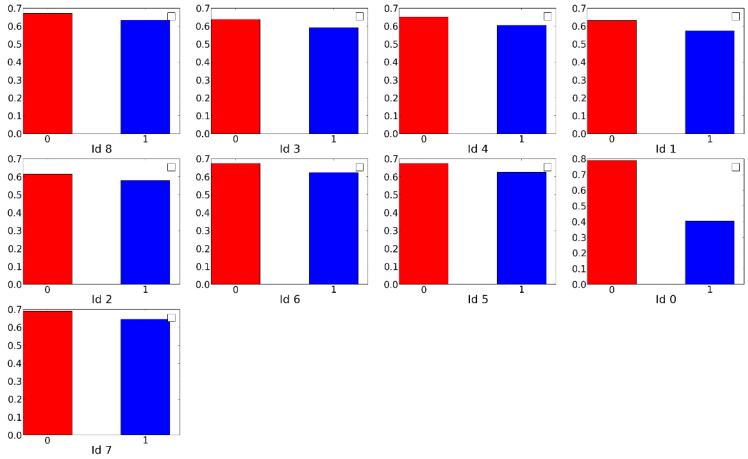
(c) Série 2

FIGURE 4.19 – Résultats d'identification pour l'environnement R&D (10 000 mots)

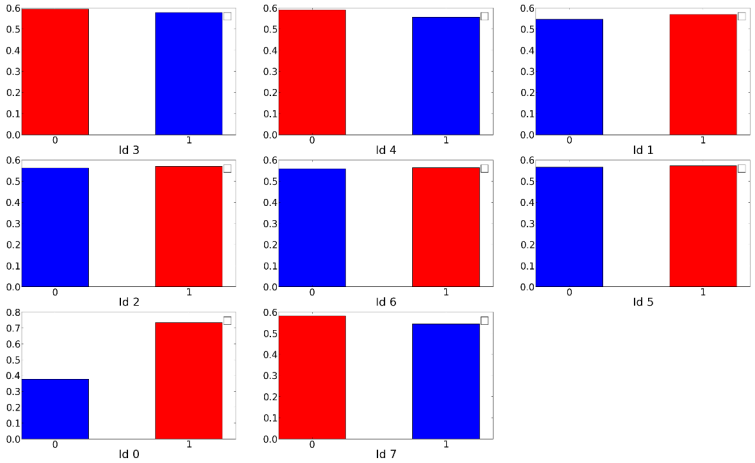
La Figure 4.19 présente les résultats pour l'environnement R&D. On y a sélectionné trois séries, indexées de 0 à 2. Dans tous les cas, le nœud de référence (d'indice 0) est correctement identifié. On observe dans la série 0 une confusion importante entre les séries 0 et 2, et une légère incertitude dans la série 1. En revanche, tous les nœuds de la série 2 sont correctement identifiés. Le Tableau 4.4 représente la matrice de confusion correspondante : pour chaque série, on note les résultats en pourcentage. On obtient sur l'ensemble des séries 63% d'identifications correctes.

	Série 0	Série 1	Série 2	Nombre nœuds
Série 0	40%	0%	60%	15
Série 1	11%	67%	22%	9
Série 2	0%	0%	100%	8

Tableau 4.4 – Matrice de confusion pour les séries R&D



(a) Série 0



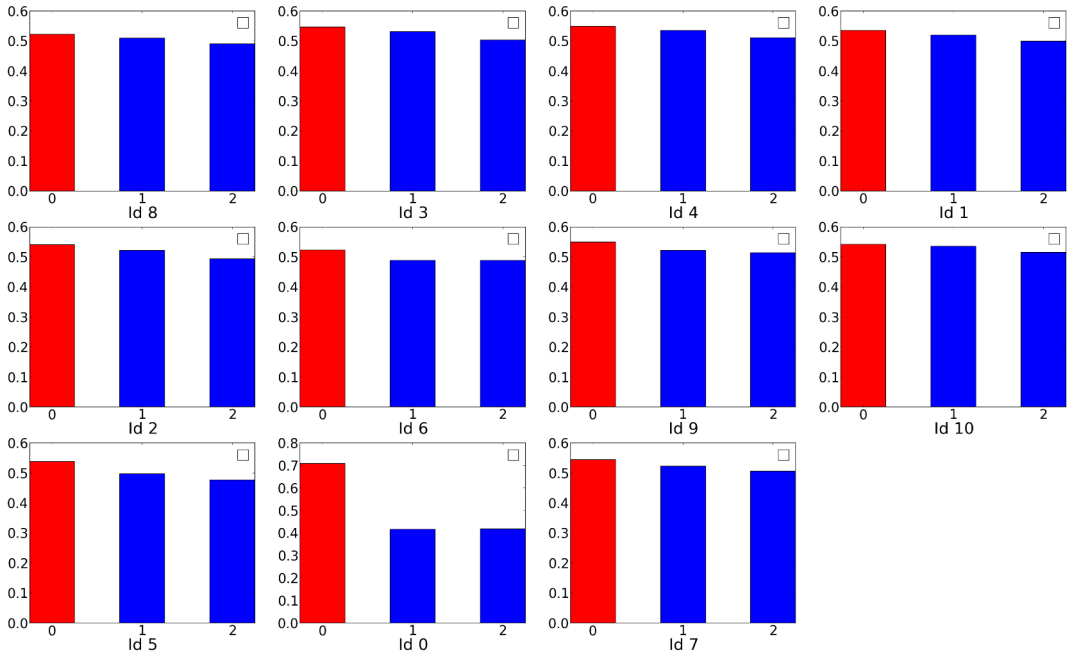
(b) Série 1

FIGURE 4.20 – Résultats d'identification pour l'environnement boutique (10 000 mots)

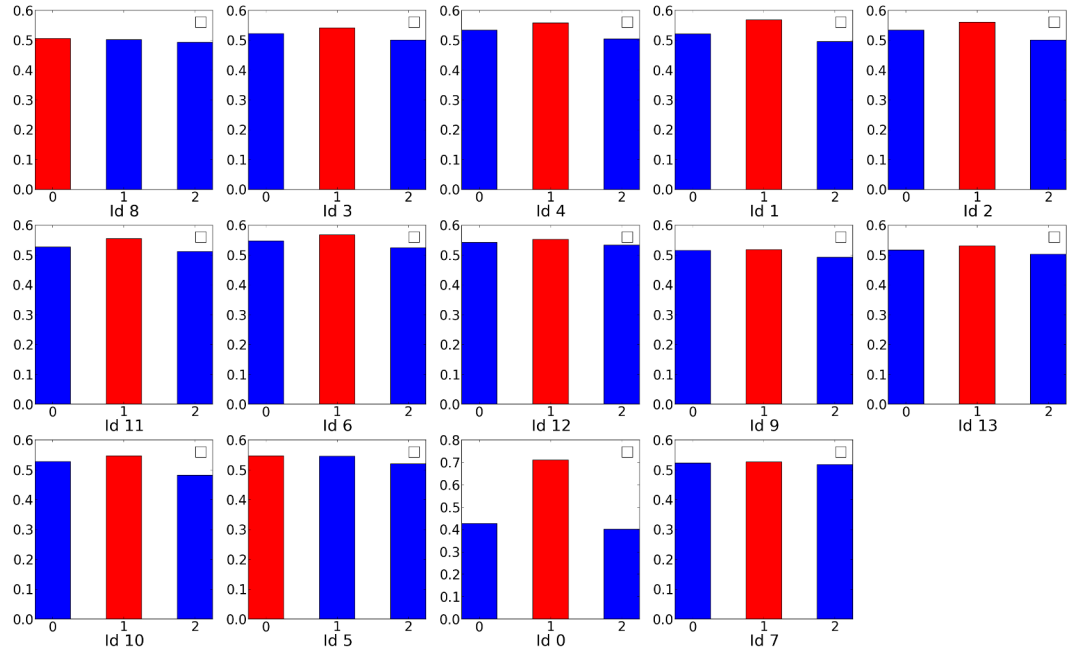
La Figure 4.20 correspond aux résultats de l'environnement boutique, qui contient 2 séries (une à chaque étage). On remarque que l'identification est ici nettement meilleure : seulement deux nœuds de la série 1 sont mal identifiés. On obtient 88% d'identification correcte de l'étage, et la matrice de confusion décrite par le Tableau 4.5 Les séries ont été acquises à deux étages différents, avec des aspects bien distincts, ce qui explique les bons résultats.

	Série 0	Série 1	Nombre nœuds
Série 0	100%	0%	9
Série 1	25%	75%	8

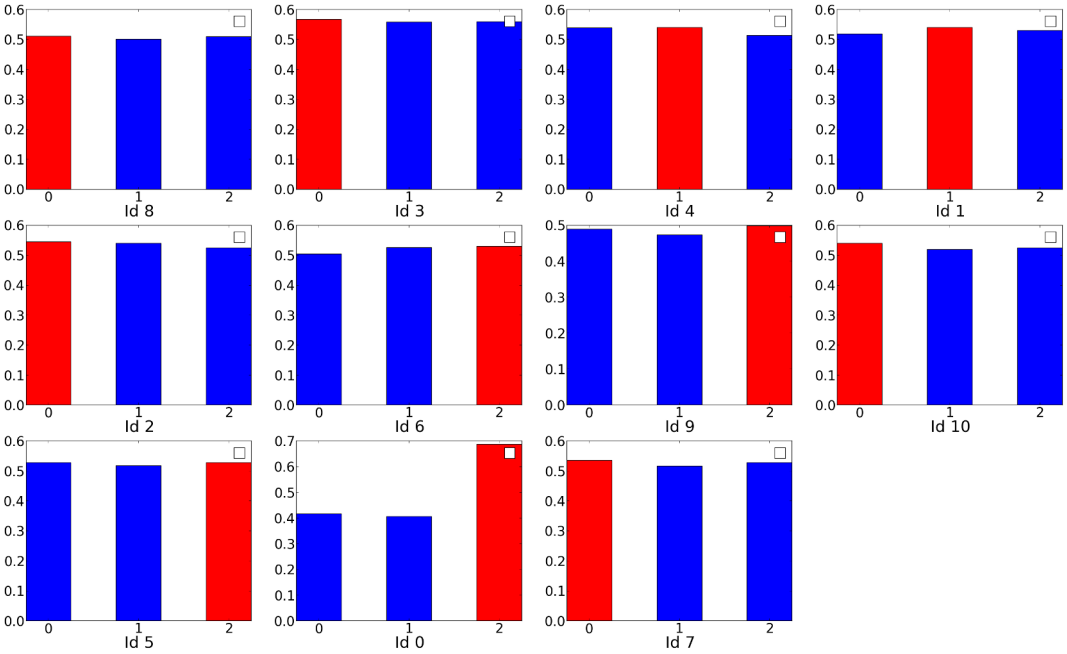
Tableau 4.5 – Matrice de confusion pour les séries boutique



(a) Série 0



(b) Série 1



(c) Série 2

FIGURE 4.21 – Résultats d'identification pour l'environnement salle de tests (10 000 mots)

Enfin la Figure 4.21 présente l'identification dans la salle de tests. Les séries 0 et 1 sont bien identifiées, mais on voit que la série 2 est plus confuse. En pratique, cette série est située entre les deux autres, à un emplacement où elles sont visibles, ce qui pourrait expliquer la confusion. On obtient au total un taux d'identification correcte de 75%, ce qui est plutôt satisfaisant dans un environnement aussi peu texturé et difficile.

	Série 0	Série 1	Série 2	Nombre nœuds
Série 0	100%	0%	100%	11
Série 1	14%	86%	0%	14
Série 2	45%	18%	37%	11

Tableau 4.6 – Matrice de confusion des séries salle de test

On voit donc qu'on peut obtenir une discrimination entre les nœuds, qui atteint ses limites si l'environnement ne change pas suffisamment mais qui peut discriminer correctement deux environnements différents. La marge d'amélioration reste importante.

Pour résoudre les cas d'ambiguïté, il serait possible de rajouter d'autres critères pour discriminer le nœud courant. Par exemple, on pourrait tenter d'effectuer une localisation par rapport aux différents nœuds candidats et vérifier que cette localisation est cohérente, ou bien rajouter des critères de répartition spatiale.

4.3 Transitions

Cette section est dédiée aux résultats de la navigation entre les nœuds. La section 4.3.1 décrit les résultats obtenus pour le contrôle du cap du robot en translation et en rotation. La section 4.3.2 détaille les résultats du calcul d'échelle par la méthode de corrélation.

4.3.1 Correction de trajectoire

Cette section traite des résultats expérimentaux à propos de la correction de trajectoire du robot par le contrôle du cap du robot. On quantifie d'abord la précision de l'approche à l'aide du simulateur Webots, puis on s'intéresse aux aspects temps réel et consommation CPU.

4.3.1.1 Précision : résultats sous simulateur

On s'intéresse ici à la précision du contrôle de trajectoire par la boussole visuelle. Pour cela, on réalise deux expériences successives sur NAO, pour tester l'efficacité en rotation et en translation. Pour la rotation, on effectue deux rotations successives d'un demi-tour dans un sens puis dans l'autre. Pour la translation, on effectue une translation en ligne droite. Dans les deux cas, on échantillonne les positions du simulateur, de la boussole et de l'odométrie, à une fréquence de 15 échantillonnages par seconde.

La Figure 4.22 représente l'évolution de l'angle du robot au cours de l'expérience, telle que donné par le superviseur (pointillé bleu), l'odométrie (pointillé vert) et la boussole visuelle (trait

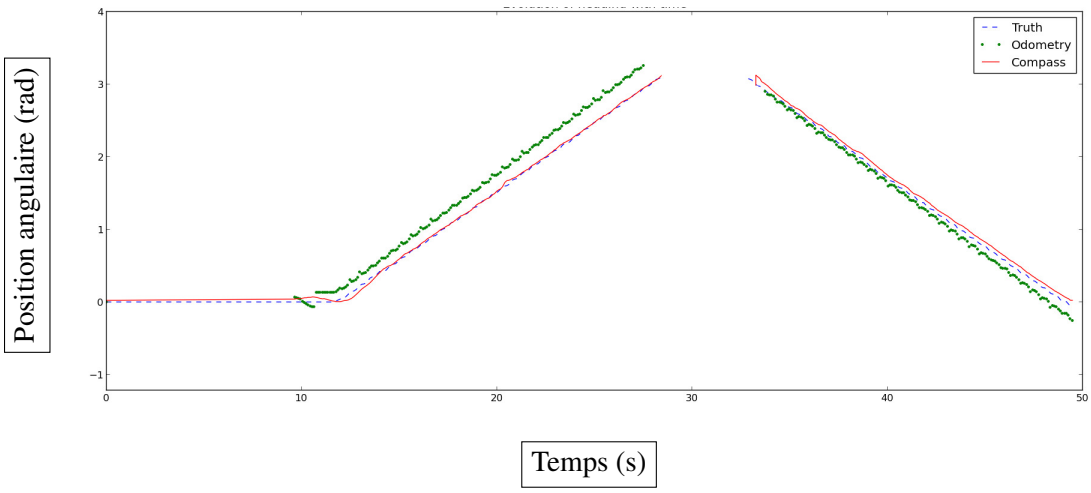


FIGURE 4.22 – Évolution de l'angle du robot calculé par l'odométrie et la boussole visuelle par rapport à l'angle réel pendant deux rotations successives

plein rouge). On voit sur la figure l'évolution linéaire de l'angle pendant les deux phases de rotation. Il est donc visible ici que l'angle calculé par la boussole est quasiment identique à la vérité terrain, alors que celui de l'odométrie dévie progressivement : la pente de la droite estimée par l'odométrie est plus importante que la vérité terrain.

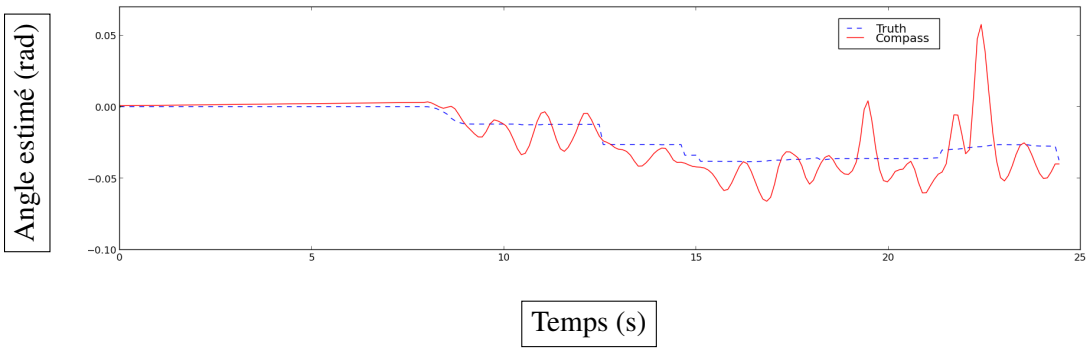


FIGURE 4.23 – Évolution du cap du robot au cours d'une translation en ligne droite calculé par la boussole et la vérité terrain

La Figure 4.23 représente l'évolution du cap du robot pendant une translation. Le contrôle est effectué avec le PID décrit dans la Figure 3.21. On voit que l'angle du robot reste très proche de l'angle initial (l'angle final est de moins de 2°), et que l'angle estimé par la boussole oscille autour de la valeur réelle avec une amplitude de 5°. Au contraire, l'angle estimé par l'odométrie est très différent. Ceci vient du fait que le robot présente une dérive systématique vers la droite, ce qui implique des commandes en vitesse vers la gauche : comme l'odométrie suppose que les commandes sont exactes, cela implique une estimation d'un angle positif.

La Figure 4.24 montre la trajectoire effectuée par le robot sous simulateur pendant un

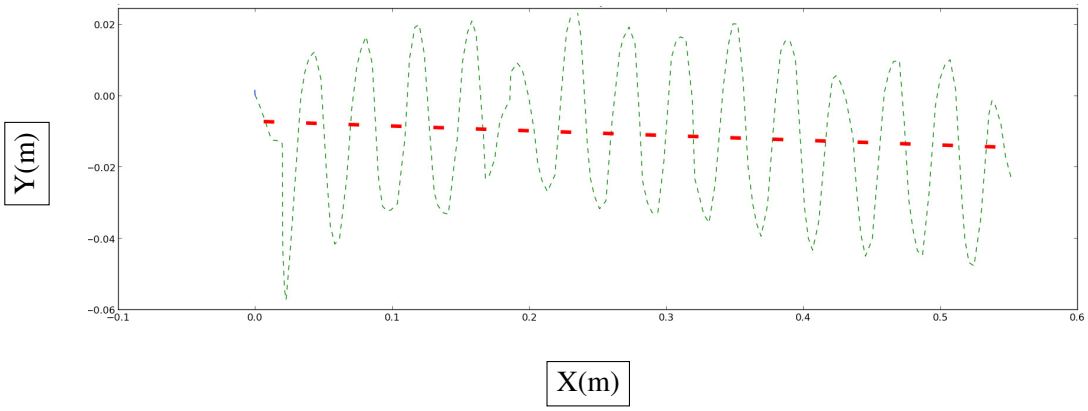


FIGURE 4.24 – Trajectoire du robot pendant une translation (vert pointillé : mouvement du torse, rouge pointillé : mouvement global de la base)

contrôle de trajectoire. Les oscillations sont dues au mouvement de balancier pendant la marche : on voit cependant que la trajectoire du robot est globalement droite, alors qu'elle dévie fortement si elle n'est pas corrigée.

En pratique, les robots NAO réels présentent effectivement une tendance à la déviation systématique. Celle-ci est due notamment à des problèmes de calibration. Chaque robot a donc une tendance à dévier, qui dépend de sa géométrie et qui n'est pas identique d'un robot à l'autre. Comme cette déviation semble très répétable, il serait probablement possible d'utiliser une phase de calibration pour déterminer cette déviation et corriger l'estimation en fonction. Cette correction n'éliminerait pas le besoin d'un contrôle du cap précis en boucle fermée, puisqu'elle tiendrait compte uniquement du robot et non pas des spécificités du sol (matière plus ou moins glissantes, présence d'irrégularités etc).

4.3.1.2 Performances

La méthode de contrôle par la boussole doit être faite en temps réel, avec une fréquence suffisante pour contrôler la trajectoire du robot pendant qu'il se déplace. Étant donnée la vitesse de déplacement de NAO, on peut se permettre d'avoir une fréquence de rafraîchissement assez basse : un contrôle satisfaisant a pu être effectué à une fréquence de 5 Hz. Plus la fréquence est élevée, meilleur sera le contrôle. Il est donc important de quantifier les impacts sur la consommation CPU.

Résolution	160x120	320x240	640x480	1280x960
5 fps	2%	4%	10%	36%
15 fps	6%	11%	31%	NA
30 fps	13%	23%	60%	NA

Tableau 4.7 – Utilisation CPU pour différentes fréquences et résolutions

Le Tableau 4.7 compare les différentes valeurs de consommation CPU en embarqué sur le robot en fonction de la résolution de l'image considérée. On voit ici que l'augmentation est approximativement linéaire avec la fréquence, ce qui est naturel, et approximativement quadratique avec la taille de l'image. La complexité du calcul vient de l'extraction des points d'intérêt, qui est quadratique en fonction de la taille de l'image. Le calcul de l'orientation par le RANSAC est linéaire en fonction du nombre de points. La configuration utilisée en pratique est une fréquence de 15 Hz pour une résolution 160x120, qui est le meilleur compromis en termes de précision contre temps de calcul et impact mémoire.

4.3.2 Estimation de l'orientation et du facteur d'échelle par la corrélation

La Figure 4.25 montre un exemple de corrélation effectuée pendant que NAO était en mouvement. Pour chaque étape, l'image de référence est indiquée à gauche, l'image courante à droite, et la matrice de corrélation en bas. La matrice de corrélation est en fausses couleurs, le rouge correspondant au maximum de corrélation et le bleu au minimum. Le modèle et la zone de corrélation maximale sont indiqués en bleu sur l'image de référence et la requête respectivement. L'image de référence correspond à l'image initiale vue par le robot, à l'arrêt. Le robot effectue une translation en ligne droite (sans contrôle de trajectoire) sur environ un mètre. L'image de référence reste la même pour la durée de l'expérience. Pour des raisons pratiques de visualisation, les images présentées n'ont pas été calculées en embarqué sur le robot, mais à partir d'une vidéo enregistrée. En pratique, l'algorithme peut être porté sur le robot à une fréquence suffisante (on atteint aisément une fréquence de 10Hz).

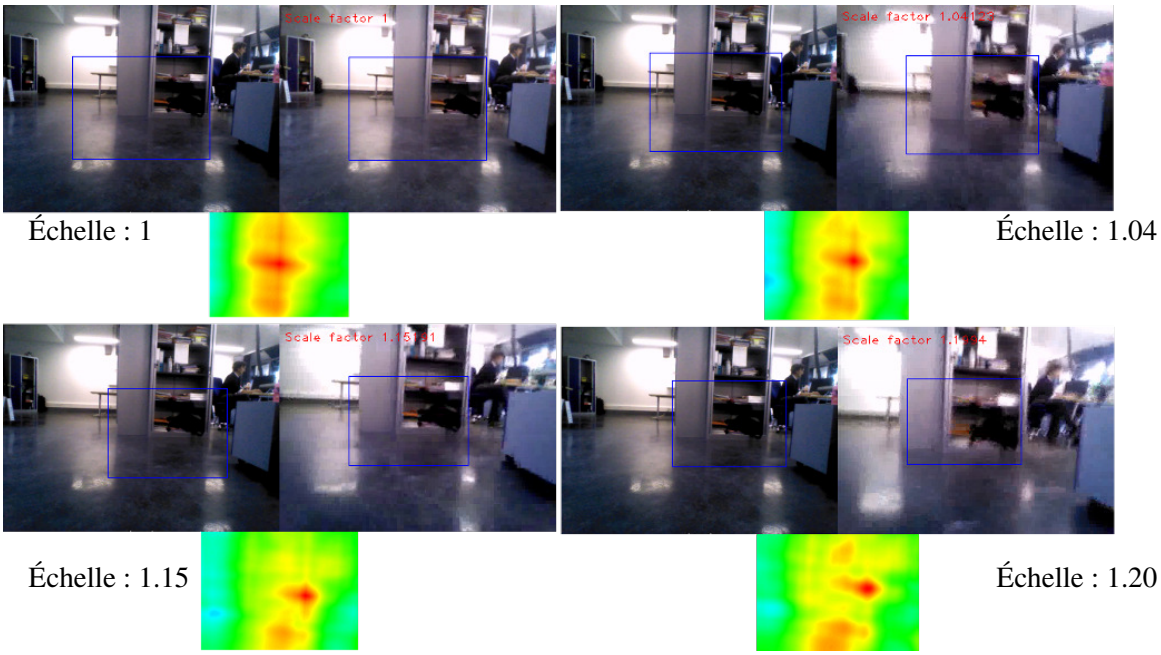


FIGURE 4.25 – Étapes successives pendant une transition
Plateforme utilisée : NAO

On voit sur la Figure 4.25 que la corrélation peut s'effectuer correctement même avec des changements d'échelle et des niveaux de flous importants. En particulier, sur la dernière image, on voit que le flou cinétique est très important, mais le maximum de corrélation est détecté de façon cohérente. On voit donc qu'il est possible d'utiliser une image de la même manière que pour la boussole visuelle, pour contrôler l'orientation du robot, mais aussi pour estimer le facteur d'échelle. L'expérience est réalisée sur NAO, et donc dans les conditions de flou cinétique les plus difficiles, ce qui montre que cette approche est valide.

En pratique, on observe régulièrement des fausses corrélations pendant le mouvement. Cependant, ces erreurs sont obtenues avec un facteur d'échelle incohérent par rapport au mouvement effectif du robot : par exemple, alors que le robot se rapproche de son image de référence et que le facteur d'échelle est donc supérieur à 1, on observe des estimations avec un facteur d'échelle nettement inférieur à 1. Ces artefacts pourraient probablement être éliminés en ne considérant dans la pyramide que les échelles proches de la dernière échelle calculée, en supposant qu'il n'y a pas de changements brutaux. Cette hypothèse est valide tant que le robot est en mouvement, surtout que les mouvements brusques (robot soulevé ou bousculé) peuvent être détectés. On observe aussi parfois un mauvais positionnement dans la requête, probablement dû au fait que le modèle utilisé correspond à un champ de vue restreint et donc ambigü. Cependant, ce problème peut probablement être compensé avec un filtre simple sur la position (auquel on pourrait aussi rajouter un filtre passe-bas pour compenser les effets des mouvements de la tête pendant la marche).

4.4 Applications et intégration dans la suite logicielle

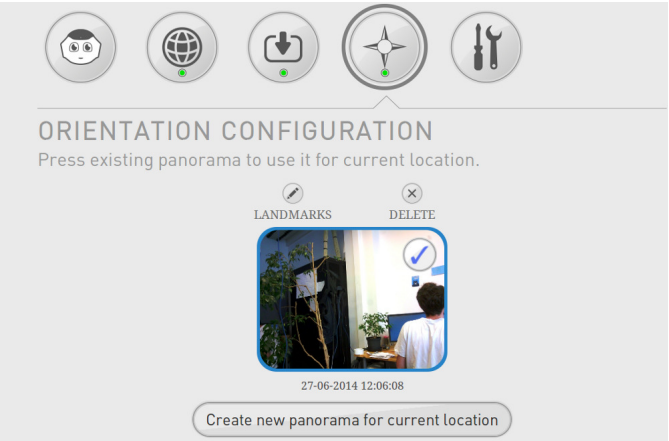
Le travail décrit dans cette thèse a été utilisé en intégré en aval au sein d'Aldebaran. On s'intéresse ici à deux applications qui ont été développées basées sur le travail de localisation. Ces applications ont été développées pour le robot Pepper.

4.4.1 Orientation

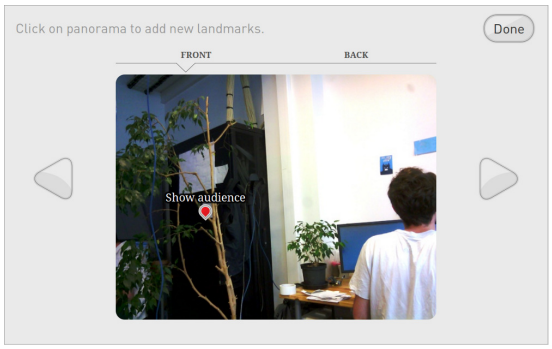
L'application "orientation" a pour but de permettre à des clients de se repérer dans une boutique en demandant leur chemin à un robot d'accueil. Le but est que le robot, après un court dialogue, soit capable de d'indiquer ensuite au client une direction à suivre pour aller à un endroit donné de la boutique (par exemple, l'accueil, les toilettes etc). Cela implique à la fois de connaître sa position actuelle, et de stocker les directions appropriées.

Pour stocker les directions de référence et se repérer, le robot utilise un nœud fixe, acquis au préalable. Ce nœud est ensuite annoté manuellement par un opérateur, par l'intermédiaire de la tablette tactile de Pepper : les images couleur acquises pour le nœud défile sur la tablette, et l'opérateur place des points annotés sur les images. Leurs positions en pixel sont ensuite converties en angulaire. Chaque position est associée à un mot-clé, qui pourra ensuite être reconnu et utilisé dans un dialogue dédié.

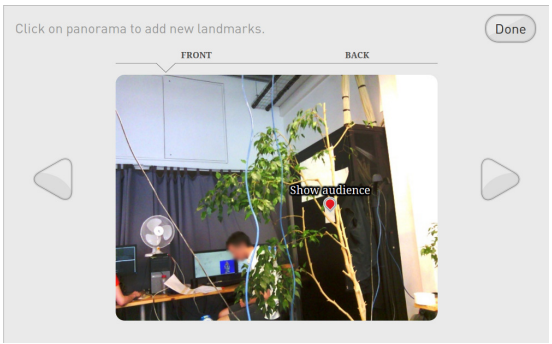
En cours d'utilisation, une fois le mot clé reconnu, le robot doit calculer sa position courante. On n'utilise qu'une seule image et on calcule l'angle à l'aide de la boussole visuelle : on gagne ainsi en rapidité d'acquisition et d'exécution. En contrepartie, on y perd en robustesse, car une



(a) Page d'accueil de l'interface d'acquisition



(b) Ajout d'une annotation ("show audience")



(c) Persistance de l'annotation sur une autre image

FIGURE 4.26 – Exemple de nœud annoté

unique image peut être partiellement recouverte ou bruitée. Pour compenser, en cas d'échec de la boussole, on acquiert d'autres images, en tournant la tête et le corps du robot dans trois directions successives : en face, à gauche puis à droite. Ainsi, le robot n'est pas gêné par la présence d'une personne en face de lui, ou bien par un manque de textures ponctuelles. L'utilisation de la boussole uniquement se justifie par le fait que le robot dans ce cas d'application ne s'éloigne pas de son nœud initial, et effectue principalement des rotations.

Cette application a été développée par l'équipe Studio à Aldebaran. L'application a été développée en Python pour l'utilisation de l'API de localisation et en Javascript pour la partie tablette. Elle doit être livrée et utilisée en magasin. Elle contient notamment des animations, pour la recherche de l'orientation et ensuite pour indiquer une direction, et un dialogue automatique. La condition de déclenchement de cette application se fait donc au sein d'un dialogue, dans le cas où l'utilisateur pose une question suivant le modèle "Où se trouve * ?" ou autre question similaire.

La Figure 4.26 montre l'interface d'annotation développée pour cette application. On voit notamment qu'il est possible de rajouter des annotations sur une image donnée, ici l'annotation

"show audience" qui peut servir à orienter le robot vers son public pour une danse. L'application utilise ensuite la correspondance entre position en pixel et position angulaire pour faire persister les annotations d'une image à l'autre.

4.4.2 Retour à la maison

L'application "retour à la maison" a pour but de garantir que le robot ne s'éloigne pas trop de son emplacement prévu au cours de la journée, et est capable de revenir à son point de départ le cas échéant. En effet, le robot effectue régulièrement des petits déplacements, pour se diriger vers les personnes proches et entrer en interaction, ou tout simplement pour rechercher des personnes avec qui interagir. Il est également amené à danser à intervalles réguliers. Ces déplacements cumulés ramènent le robot approximativement à sa position initiale, mais les erreurs d'orientation en particulier s'accumulent. Il est donc nécessaire de replacer le robot régulièrement.

Pour cela, on vérifie périodiquement la position du robot. Pour économiser du temps d'acquisition et de calcul, on commence par vérifier si l'information de l'odométrie est suffisamment précise. Pour cela, on estime l'incertitude de l'odométrie à l'aide du déplacement cumulé du robot depuis la dernière localisation complète, et en comptant d'éventuelles perturbations. Si l'incertitude est suffisamment réduite et si le robot n'a pas été poussé depuis la dernière localisation, alors l'odométrie est suffisante et on utilise cette position estimée pour recaler le robot. Si l'odométrie est trop incertaine, alors on déclenche une localisation complète. Pour cela, on rajoute la condition de déclenchement appropriée à la vie autonome, qui déclenche la localisation dès que possible. Une fois la localisation obtenue, on ramène si nécessaire le robot à sa position initiale.

L'intégration de la localisation complète dans une application est particulièrement délicate, car il s'agit d'un processus assez long, et les mouvements de la tête sont imposés (pour couvrir un champ de vision suffisant). Pour cela, le temps de calcul est maquillé par des animations et des sons. Il est aussi important de vérifier au préalable que le champ de vision n'est pas obstrué par un trop grand nombre de personnes. Cependant, cette étape est cruciale, car son absence implique un robot qui dérive, sort de son périmètre ou tourne le dos aux spectateurs.

L'application est développée en Python, via le logiciel Choregraphe, et conçue comme une extension de la vie autonome du robot. L'application est en cours d'intégration et de livraison aux clients d'Aldebaran

4.5 Conclusion

On a montré dans ce chapitre la validité des approches décrites dans le chapitre 3. Pour cela, on a utilisé un simulateur pour avoir à la fois un environnement simple et contrôlé et une vérité terrain précise. Une base de données de test importante a également été collectée, dans des environnements différents (de bureaux, peu texturé avec des miroirs, dans une boutique). Elle a été utilisée pour une validation qualitative en vérifiant le taux de succès de la localisation dans des environnements proches des nœuds, et quantitative en utilisant une vérité terrain approximative.

Ces expériences ont donc montré que les différentes sources de localisation donnaient des résultats performants individuellement. La boussole visuelle est robuste aux faux appariements,

aux occultations et au flou. La corrélation permet d'obtenir des informations fiables même dans des environnements peu texturés, même en présence d'obstacles ou de flou cinétique. On voit tout de même qu'elle est sensible à des fausses détections en l'absence d'hypothèse préalable. Le recalage de profils par l'ICP reste efficace malgré la forte déformation du signal, et donne des résultats dont la précision est satisfaisante par rapport à l'amplitude du bruit. On obtient des résultats satisfaisants en présence d'obstacles, tant qu'il reste un champ de vision suffisant. La structure hiérarchique permet de renforcer le résultat en utilisant les résultats des sources comme hypothèse pour les suivantes, et permet d'obtenir des taux de réussite importants dans tous les environnements considérés.

En ce qui concerne les transitions, la boussole visuelle permet de contrôler l'orientation du robot de façon fiable, à la fois pour des rotations pures et des translations. Le contrôle par un PID permet d'obtenir une précision bien meilleure que l'odométrie, tout en étant utilisable à une fréquence suffisante en embarqué. La corrélation donne également des résultats prometteurs en mouvement, puisqu'elle estime de façon robuste la position et l'échelle du robot, même en utilisant des images prises par le robot en mouvement.

Cette méthode a été intégrée dans le code client Aldebaran, et est actuellement en cours d'intégration au sein d'applications livrées. Même si la méthode a encore des limites, notamment le temps nécessaire pour l'acquisition d'un nœud (de référence ou de requête), il s'agit d'un aspect important qui sera intégré et amélioré dans la suite. Par exemple, une application de "robot guide" est en train d'être mise en place, qui permet notamment de rajouter un aspect sémantique à la localisation, à travers la mise en place de repères.

CHAPITRE 5

Conclusion et perspectives

Le sujet de la localisation et de la cartographie est particulièrement important en robotique. Il peut s'agir de SLAM, de localisation par rapport à une carte préétablie, de détection de fermeture de boucle, ou encore de suivi de trajectoire. Ce sujet est couvert depuis maintenant plus de vingt ans, mais les cas d'application et les moyens disponibles sont de plus en plus contraignants. Historiquement, le problème a commencé par être traité pour des robots disposant de capteurs métriques très complets, avec notamment un large champ de vision et une portée importante : télémètres laser, Velodyne, et plus récemment caméras 3D. Des algorithmes de plus en plus efficaces ont été développés pour obtenir des informations de position métrique du robot, en deux voire trois dimensions. Certains ont même été appliqués à l'aide du robot NAO, modifié pour l'occasion en y rajoutant un laser ou une caméra 3D (mais en effectuant tous les calculs en déporté). Cependant, les capteurs métriques posent un problème de coût et de volume de données. Pour cela, la communauté s'est intéressée à des capteurs non métriques, à savoir les caméras. Ces capteurs ont l'avantage d'être peu volumineux, peu chers et de contenir une information riche. On trouve donc des systèmes de localisation métriques basés sur des caméras (monoculaire ou paire stéréo).

Le formalisme métrique, qui a été le premier envisagé historiquement, n'est cependant pas nécessairement le plus adapté. En particulier, il supporte mal le passage à l'échelle, ou le stockage d'informations qualitatives comme celles que peut fournir une caméra. Le formalisme dit topologique a donc été développé progressivement : il consiste à repérer le robot par rapport à un graphe et non plus une carte métrique. Ce graphe ne contient donc pas nécessairement des informations métriques, mais plutôt qualitatives ou sémantiques. Ce formalisme est particulièrement employé dans le cas de systèmes équipés uniquement de caméras. Il existe de nombreuses approches, à la fois pour fournir une localisation topologique et pour effectuer des suivis de trajectoires. Elles sont en général basées sur des méthodes parcimonieuse à base de points d'intérêt,

éventuellement renforcées par des descripteurs globaux. C'est donc ce type d'approche qui a été privilégié ici.

Cette thèse a cherché à résoudre le problème de la localisation et de la navigation de robots humanoïdes en environnement non contraint. Il s'agit donc d'une version particulièrement difficile du problème général. En effet, les robots sont équipés de capteurs restreints, avec des contraintes de coût et d'intégration forte. De plus, les applications visées ont lieu dans un environnement non contraint, qui ne peut pas être instrumenté, et qui est amené à changer : des boutiques avec des clients non experts, ou un domicile d'un particulier. Le but de la thèse est donc de tirer un maximum d'informations de localisation, aussi précises que possible, avec une information partielle. La structure choisie pour gérer ces informations est un graphe topologique, où les emplacements clés correspondent à des nœuds, et les déplacements possibles entre les nœuds sont repérés par les transitions. La thèse s'articule donc autour de deux aspects principaux : la localisation du robot par rapport à nœud, et la transition entre les nœuds. Les contributions principales de la thèse sont donc des méthodes qui permettent le calcul de l'orientation du robot et des indications sur sa position uniquement à partir de caméras monoculaires : la boussole visuelle et la corrélation multi-échelle. On peut ajouter à cela l'utilisation d'une structure hiérarchique souple et l'utilisation d'un formalisme topologique. On obtient donc alors un système de localisation le plus complet possible à partir d'informations limitées, sur un robot humanoïde à bas coût.

Le graphe topologique qui décrit l'environnement est construit a priori, lors d'une visite préalable de l'environnement par le robot. Les nœuds du graphe sont construits de façon à avoir des données panoramiques : le robot prend une série d'images (caméra monoculaire et éventuellement caméra 3D) avec différentes positions de la tête et en tournant sa base. On compense ainsi le faible champ de vision des capteurs du robot, au prix d'une acquisition plus longue et plus contrainte. Les données ainsi acquises fournissent des images de référence pour la suite.

La localisation par rapport à un nœud est faite de façon hiérarchique et progressive. L'odométrie du robot est la source d'information la plus simple mais aussi la plus imprécise : elle dérive fortement (surtout sur NAO), et ne supporte pas les irrégularités du sol, les chutes ou les perturbations fortes de l'équilibre. Elle est particulièrement imprécise pour l'orientation du robot. Il s'agit toutefois d'une indication rapide à obtenir, et dont on peut majorer l'imprécision. Pour cette raison, on s'en tient uniquement à l'odométrie aussi longtemps que possible, en estimant son incertitude à l'aide des mouvements cumulés du robot.

Pour compenser la dérive de l'odométrie, on utilise tout d'abord la boussole visuelle, qui calcule l'orientation du robot de façon plus précise et fiable. La boussole visuelle est basée sur l'appariement de points d'intérêt (de type BRIEF) entre les images du nœud courant et des images courantes : elle repose sur l'hypothèse que les mouvements des points sont dûs aux rotations du robot, et une recherche optimisée des meilleures images à apparier dans le noeud. Cette méthode est rapide et plutôt précise tant que l'hypothèse est raisonnablement valide, mais atteint ses limites lorsque le robot s'éloigne du nœud initial. En effet, l'hypothèse ne tient que pour des rotations pures ou des points à l'infini, ce qui n'est pas le cas quand le robot bouge.

La boussole visuelle doit donc complétée. Pour cela, on utilise une méthode dense, plus robuste mais aussi plus lourde en temps de calcul, basée sur la corrélation de type ZNCC. On compare une image extraite du noeud avec la concaténation des images courantes, et on en déduit la

déviation angulaire correspondante. On rend cette approche robuste aux changements d'échelle en calculant la corrélation sur une double pyramide d'image, et en sélectionnant la meilleure échelle possible. Cette méthode permet à la fois de s'affranchir des inconvénients d'une méthode parcimonieuse et de supporter les changements d'échelle. Elle permet donc d'obtenir une mesure du cap plus précise à plus grande distance, tout en fournissant une mesure qualitative de l'éloignement du robot, au prix d'un temps de calcul plus important, mais toujours envisageable sur le robot.

Enfin, l'information de position du robot peut être complétée si jamais le robot possède un capteur métrique (caméra 3D de Pepper ou de ROMEO, NAO modifié). On utilise un algorithme d'ICP pour recaler des profils de noeuds et ainsi obtenir une position métrique par rapport au nœud de référence. Cette méthode nécessite une initialisation proche du résultat final pour converger correctement, au minimum en orientation. Si aucune information n'est disponible a priori, on construit une hypothèse d'orientation approximative à l'aide des descripteurs Shape Context. On cherche également à compenser les imprécisions du recalage dûs à la fois à l'environnement dynamique et à l'intégration du capteur (effet des lentilles). On obtient ainsi une localisation en position, avec une précision meilleure que l'odométrie.

Toutes ces sources d'informations sont combinées de façon hiérarchique, par ordre de complexité et de précision croissantes. Chaque source de localisation fournit son résultat partiel aux suivantes, qui l'utilisent comme hypothèse préliminaire, ce qui permet de simplifier les calculs en garantissant une bonne initialisation du problème. Les résultats des différentes sources sont combinés et renforcés en utilisant un formalisme probabiliste simplifié.

Pour résoudre le problème du robot capturé, qui n'a aucune information a priori sur sa position, on utilise une identification par sac de mots, en construisant un vocabulaire à partir des points d'intérêt extraites des nœuds du graphe. Pour cela, on construit un vocabulaire hiérarchique sur les descripteurs binaires BRIEF. On apparie ensuite un nœud candidat avec le graphe en utilisant la mesure du TF-IDF utilisée classiquement dans ce genre de cas.

En ce qui concerne les transitions entre noeuds, on met au point une méthode basée sur la boussole visuelle pour contrôler la trajectoire du robot. On utilise l'information de déviation relative à une image de la boussole pour contrôler le cap du robot par rapport à une direction donnée. On peut alors effectuer des rotations sur place, ou des translations en ligne droite dans une direction donnée. Cela permet donc à la fois de contrôler les mouvements du robot pendant une acquisition de nœud, et d'effectuer des transitions dans la direction d'un nœud voisin. On peut alors contrôler l'éloignement par rapport au nœud suivant en utilisant la méthode de calcul du facteur d'échelle de la corrélation, qui peut également être effectuée pendant le mouvement du robot (y compris NAO).

On a donc développé un système qui permet d'obtenir une localisation et une navigation dans un graphe topologique, tout en restant réalisable en embarqué sur des robots humanoïdes contraints. Ces méthodes ont été testées et validées sur des bases d'exemples en conditions réelles et sont en cours d'intégration dans la suite logicielle d'Aldebaran. Une des applications permet à Pepper de devenir un robot guide, en annotant un nœud avec des informations sémantiques et en intégrant un dialogue qui permet à l'utilisateur de demander son chemin au robot.

La plupart du travail de la thèse a porté sur la localisation dans un nœud donné. Les transitions et l'identification du nœud courant sont encore des prototypes, et méritent des améliora-

tions. En particulier, il serait intéressant d'établir un contrôle de trajectoire directement à partir de la corrélation, et non de l'utiliser uniquement comme un critère d'arrêt. On pourrait notamment ainsi limiter la recherche de l'échelle en ne s'intéressant qu'aux échelles cohérentes et vraisemblables. On pourrait également ajouter de nouvelles transitions au graphe en les extrapolant à partir de corrélations visibles d'un noeud à l'autre : s'il est possible d'établir une corrélation entre deux noeuds, avec un angle et un facteur d'échelle correspondant, alors il est probable qu'il soit possible d'effectuer une transition.

L'identification du nœud courant est encore basique, et pourrait s'inspirer de descripteurs plus pertinents qui prennent en compte une information globale et non des descripteurs locaux. En particulier, il serait intéressant de s'inspirer des descripteurs qui tiennent compte de la répartition spatiale des points d'intérêt, et éventuellement d'intégrer le signal 3D s'il est disponible, pour rendre l'identification plus robuste. Cette robustesse est notamment importante dans le contexte de l'utilisation des robots Pepper dans les boutiques SoftBank, car les différentes parties d'une boutique sont assez similaires (présence de grandes vitrines uniformes).

De plus, il serait également très intéressant de se pencher davantage sur l'évolution d'un nœud avec le temps. En effet, il serait utile de détecter lorsque les changements de l'environnement sont trop importants et nuisent à la qualité de la localisation : il peut s'agir de changements dans les conditions d'éclairage avec l'heure de la journée, ou bien de changement de l'environnement comme des objets déplacés. Le robot pourrait alors acquérir un nouveau nœud avant que les différences ne soient trop importantes, et compléter l'information du graphe.

La question de l'évolution du graphe en général et de sa construction autonome reste aussi ouverte. Par exemple, il est nécessaire de gérer les transitions qui sont invalidées par des obstacles, ou d'en rajouter. Il serait aussi utile de pouvoir rajouter de nouveaux nœuds à la volée pendant la localisation, par exemple pour compléter une zone découverte par hasard ou pour compenser les changements de l'environnement. La thèse suppose qu'un tour du propriétaire a été réalisé au préalable, mais on pourrait imaginer appliquer des méthodes de fermeture de boucle et de segmentation pour réaliser le découpage en nœuds de façon autonome.

Cette thèse a permis de montrer qu'il est possible d'obtenir un système de localisation et de navigation sous des contraintes très fortes, à la fois de plateforme, de capteurs, de puissance de calcul et de contexte d'utilisation. Il existe des possibilités d'extension et d'amélioration du travail réalisé, avec des perspectives intéressantes pour passer à l'échelle, donner plus d'autonomie au système et mieux tenir compte du caractère dynamique de l'environnement.

ANNEXE A

Points d'intérêt et descripteurs

Les points d'intérêt sont une méthode très populaire en traitement d'images pour toute application de suivi ou d'identification. Ils permettent une approche parcimonieuse et donc économe en temps et en mémoire. Le principe est de détecter dans une image donnée des points remarquables, et de leur associer un descripteur représentatif. Le but est alors de retrouver ces points dans d'autres images pour pouvoir les apparier. On peut alors utiliser ces appariements dans des applications de reconnaissance d'objets, d'assemblage d'image ou de suivi d'objets et d'amers visuels.

L'extraction des points d'intérêt est effectuée par un détecteur. Pour chaque point d'intérêt, celui-ci donne une position, une taille, une intensité et éventuellement une échelle et une orientation. On définit donc une zone centrée sur un point. Le but d'un détecteur est d'extraire des points significatifs d'une référence, et de les retrouver par la suite dans des images de référence.

Pour décrire et comparer les points d'intérêt, on utilise un descripteur. Celui-ci calcule à partir d'un point d'intérêt donné un descripteur du point. Un descripteur est généralement représenté sous la forme d'un vecteur de taille donnée. Pour chaque type de descripteur, on définit également une notion de distance entre deux descripteurs.

A.1 Détecteurs

Un détecteur doit répondre à plusieurs critères importants. Parmi ces critères, on trouve la précision, la robustesse au flou, aux changements de luminosité et d'orientation. Chaque descripteur répond de façon particulière à ces critères, qui impactent la performance des algorithmes qui les utilisent.

Il existe de très nombreux types de détecteur et descripteurs. Parmi les détecteurs les plus courants, on peut citer :

— le détecteur de coin de **Harris** (Harris and Stephens, 1988). Il s'agit d'un détecteur peu coûteux à extraire, mais également assez peu robuste ;
— Feature Accelerated Segment Test **FAST**, (Rosten and Drummond, 2006). Il s'agit d'un détecteur de coins, qui a l'avantage d'être très rapide à extraire et relativement robuste au flou. Ce détecteur repose sur la comparaison d'intensité de pixels entre un pixel central et les pixels de la périphérie.
— Maximally Stable Extremal Regions **MSER** ; (Matas et al., 2004). Ces régions sont détectées à l'aide d'un système de seuillage progressif des images, qui permet de détecter des régions "maximales". Celles-ci sont stables aux transformations affines et au changement de luminosité
— Scale Invariant Feature Transform **SIFT** (Lowe, 1999), construit par une méthode de différence de Gaussiennes (DOG - Difference Of Gaussians) comme approximation de calcul de Laplacien : on considère les différences des transformations de l'image originale à laquelle on a appliqué des Gaussiennes de différentes tailles, et on sélectionne les points d'intensité maximales. Une orientation typique est calculé en calculant les orientations dominantes des gradients autour du point. Celui-ci est donc par construction robuste au changement d'échelle et d'orientation ;
— Speeded Up Robust Features **SURF** (Bay et al., 2006), qui est assez similaire au détecteur SIFT. Il est basé sur une recherche de valeurs maximales de la Hessienne de l'image sur différentes échelles, approchée par de simples filtres boîtes (basés sur des différences de zones de l'image) qui ramènent le calcul à des différences de zones d'images. Une orientation est calculée à partir des réponses à la transformée en ondelettes dans différentes directions.

Le choix du détecteur choisi repose sur un compromis entre robustesse et vitesse d'extraction. Il est nécessaire de choisir les critères en fonction de l'application. Par exemple, la robustesse aux changements d'orientation est cruciale pour des applications de reconnaissance d'objets : on doit pouvoir retrouver l'objet dans des positions différentes. En revanche, dans le cadre du suivi pendant la marche d'un robot, cela peut au contraire être un désavantage : on sait que les points ne seront pas soumis à des changements d'orientation importants, mais on laisse la possibilité de confondre des objets qui sont similaires à une rotation près, comme par exemple un pied de chaise et une table.

De même, lorsque les détecteurs ne sont pas robustes aux changements d'échelle, il est possible de compenser cet inconvénient en extrayant les points d'intérêt à différentes échelles simulées. Ces échelles sont obtenues en construisant une pyramide d'images. Celle-ci est constitué d'images auxquelles on applique progressivement un facteur d'échelle de plus en plus faible. Ainsi, on simule la détection à différentes échelles dans l'image. Cette méthode, même si elle n'est pas aussi précise qu'un détecteur comme SIFT ou SURF, permet d'obtenir une approximation suffisante et est souvent plus rapide que d'extraire directement des détecteurs multi-échelle.

Dans le cas d'application de cette thèse, le détecteur FAST multi-échelle est le meilleur compromis. Il permet d'obtenir un grand nombre de points d'intérêt rapidement et à faible coût, ce qui donne des chances de compenser d'éventuelles obstructions ou changement de l'environnement. La robustesse aux changements d'échelle est une caractéristique importante, mais l'extension multi-échelle permet de l'obtenir approximativement : on perd en précision par rap-

port à SURF par exemple, mais le gain en temps de calcul est significatif. Enfin, la robustesse au flou et aux changements de luminosité est suffisante pour les applications considérées, à savoir la localisation par la boussole visuelle (section 3.2.1) et le contrôle du cap du robot (section 3.3.1). Cependant, comme le détecteur possède une interface générique, il pourrait être remplacé par un autre détecteur, par exemple si on a davantage de puissance de calcul.

A.2 Descripteurs

Un descripteur doit comme son nom l'indique caractériser un point d'intérêt donné. Le but d'un descripteur est de représenter de façon compacte la zone définie par le point d'intérêt. Ce descripteur doit être comparable à d'autres descripteurs du même type pour pouvoir ensuite apparier les points entre eux : il doit donc être également discriminant.

A.2.1 Construction du descripteur

Beaucoup de descripteurs sont représentés par un vecteur. La méthode d'extraction varie selon les descripteurs, mais le but est toujours de représenter la structure locale de l'image au niveau du point d'intérêt.

Parmi les descripteurs les plus courants, on trouve :

— **SIFT**. On ramène la région définie par le point d'intérêt à une grille divisée en 4 x 4 zones. Cette région est orientée à l'aide de l'orientation du point d'intérêt. Dans chacune de ces zones, on calcule un histogramme à 8 classes qui correspondent chacune à une plage d'orientations donnée du gradient. Le descripteur correspond donc au 128 valeurs concaténées des intensités de ces classes. Il s'agit d'un des descripteurs les plus robustes de l'état de l'art actuel ;

— **SURF**. Celui-ci est très proche du descripteur SIFT. On divise ici aussi la zone définie par le point d'intérêt en 4 x 4 zones. Pour chaque zones, on calcule 4 valeurs correspondant aux valeurs des Hessiennes dans les directions x et y et de leurs valeurs absolues. On obtient ainsi un vecteur de 64 nombres réels. Il existe également une variante SURF-128 qui contient davantage de descripteur, et une variante sensible à la rotation U-SURF (Upgraded-SURF), où on ne tient pas compte de la direction du point d'intérêt. Ce descripteur perd en robustesse par rapport au SIFT, mais est beaucoup plus rapide à extraire en compensation ;

— **BRIEF** Binary Robust Independent Elementary Feature (Calonder et al., 2010). Il s'agit d'un descripteur extrêmement simple à extraire et pourtant efficace. On définit 256 paires de points dans la zone du point d'intérêt. Ces 256 zones sont répartis en utilisant une répartition Gaussienne autour du centre de la zone. Pour chacune de ces paires de points, on compare les valeurs des pixels de chaque extrémité de la paire. Si la première valeur est plus faible que la seconde, alors on associe 1 à cette paire, et 0 sinon. On construit donc un vecteur binaire de taille 256 qui correspond aux résultats concaténés des comparaisons. Il existe une version robuste à la rotation, Oriented BRIEF ORB (Rublee et al., 2011). Pour cela, on rajoute à des détecteurs FAST une mesure de l'orientation du coin

détecté, et on utilise cette orientation pour transformer les positions des paires de points avec cette rotation.

Le choix du descripteur repose donc sur un compromis entre robustesse et facilité d'extraction. Le descripteur SIFT est reconnu comme étant le plus robuste, mais les pertes de robustesses des autres sont compensés par des gains significatifs en temps de calcul. De plus, les descripteurs SIFT et SURF sont brevetés et donc ne peuvent pas être utilisés pour des applications commerciales. Pour ces raisons, cette thèse utilise le descripteur BRIEF pour caractériser les points d'intérêt : sa robustesse est suffisante pour les applications considérées, et le gain de temps par rapport aux autres descripteurs est considérable.

A.2.2 Comparaison de descripteurs

Une fois les descripteurs obtenus, il faut les comparer pour pouvoir apparier les points d'intérêt. L'idée est de trouver le plus proche voisin d'un descripteur donné parmi un ensemble de points de référence.

Pour apparier les descripteurs, on peut utiliser la méthode naïve qui consiste à comparer le descripteur de requête à chaque descripteur de référence dans l'ordre où ils ont été extraits, et à sélectionner le plus proche. On a donc une complexité en $O(n)$ où n est le nombre de descripteurs de référence. Cette méthode est exacte et simple à implémenter, mais peu efficace. Elle ne peut donc être utilisée que si on a un nombre réduit de points de référence (ce qui n'est pas le cas en général).

Une approche plus efficace consiste à utiliser un K-d Tree. Un K-d Tree consiste à organiser les valeurs à comparer dans un arbre binaire, qui contient sur chaque noeud un vecteur de dimension k. Chaque noeud divise l'espace en deux sous-espace, et on associe aux fils de ce noeud les ensembles de points associés au noeud parent qui sont dans l'un ou l'autre sous-espace. Une construction classique consiste par exemple à considérer les hyperplans définis par la normale à une dimension du vecteur considéré. Après cette construction, qui n'a besoin d'être faite qu'une seule fois après avoir défini les descripteurs de référence, la recherche du plus proche voisin est beaucoup plus efficace. En effet, pour cherche le plus proche voisin, il suffit de chercher itérativement à quel sous-espace le noeud correspond. La recherche s'effectue alors en $O(log(n))$ en moyenne (même si dans le pire des cas elle peut s'effectuer en $O(n)$). Il existe des variations sur ce principe, comme par exemple les octrees, ou l'arbre considéré est d'arité 8.

Il existe également des méthodes approximatives qui sont beaucoup plus efficaces lorsque la dimension des données devient plus grande. En particulier, on peut citer la méthode Locality-Sensitive Hashing (LSH). Celle-ci consiste à approcher des descripteurs réels par des descripteurs binaires, en définissant une fonction de hachage telle que des descripteurs réels proches ont une forte probabilité d'être représentés de la même manière, alors que des descripteurs distants ont une probabilité faible d'avoir la même représentation. La recherche de plus proche voisin consiste donc à rechercher les représentations les plus proches du hachage de la requête. Dans cette thèse, on utilise le LSH sur les descripteurs binaires issus du calcul du BRIEF.

ANNEXE B

Exemples de données expérimentales

On présente ici quelques exemples représentatifs de données expérimentales. Pour la validation et le tests des algorithmes développés dans la thèse, on a acquis plusieurs séries d'images dans des contextes différents. Pour chaque exemple de nœud, on montre à la fois les images acquises et repositionnées dans une visualisation 3D et le profil acquis par la caméra 3D. Sur les profils 3D, la position initiale du robot est située au centre de la grille, et les graduations correspondent à des mètres. L'axe X est indiqué en rouge et correspond donc à la direction initiale du corps du robot au moment de l'acquisition, et l'axe Y est indiqué en bleu. On ajoute ces axes sur les images obtenues par la caméra classique afin de faciliter la correspondance entre les images.

La Figure B.1 montre un exemple de nœud acquis dans les locaux d'Aldebaran, à l'étage R&D. Il s'agit d'un open space de bureaux, où deux murs sont constitués de fenêtres, recouvertes de panneaux solaires. Ces fenêtres sont par exemple visibles sur la partie droite de l'image. L'environnement est très ouvert : on voit au fond de l'image que l'open space se prolonge sur une grande distance, et sur le profil 3D qu'une partie est hors de portée de la caméra 3D. Sur le profil 3D, on distingue principalement les deux murs les plus proches (à gauche et en haut) et un pilier. La forme en bas du profil correspond à un rideau, mais dont le profil est extrêmement déformée par les lentilles.

On observe donc des déformations significatives sur le signal 3D, même à distance réduite : environ 2 mètres suffisent. De plus, une partie de l'environnement est hors de portée de la caméra 3D, si bien qu'on ne peut pas obtenir d'information de ce côté pour tenter de se relocaliser. On observe aussi une zone relativement pauvre en texture dans les images 2D, qui correspond à un grand mur blanc dans l'environnement (dans la partie avant gauche de l'image panoramique).

La Figure B.2 représente un nœud acquis dans la salle de tests à Aldebaran. Cette salle est utilisée pour faire des tests de longue durée des robots Pepper dans des conditions variées. En particulier, elle est équipée de miroirs, situés sur un des murs et sur un pilier central. On peut

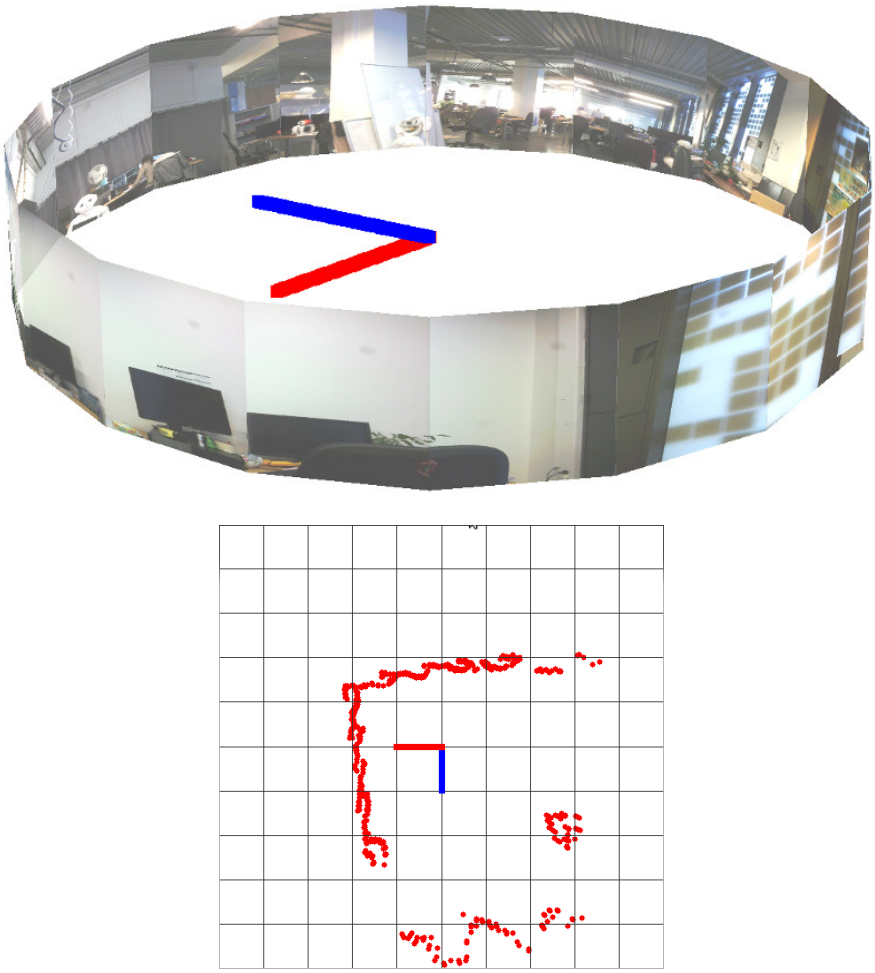


FIGURE B.1 – Exemple de nœud pris à l'étage R&D
Plateforme utilisée : Pepper

notamment voir le miroir du pilier sur la figure, en face de l'axe (on peut voir le robot lui-même en train d'acquérir le noeud), et un miroir sur le mur (à l'avant de l'image panoramique). Ces miroirs rendent la situation particulièrement difficile à traiter pour la localisation. Elles rendent les images ambiguës puisqu'on peut retrouver un motif reflété par un miroir, ce qui perturbe à la fois la corrélation et la boussole. Les miroirs sont également invisibles pour la caméra 3D, ou au mieux prennent l'apparence des obstacles reflétés par le miroir. On le voit ici sur le profil, puisque que le pilier situé en face de l'axe rouge n'est pas détecté, et le miroir derrière la position initiale apparaît également comme un trou dans le mur.

La Figure B.3 représente un nœud acquis dans une boutique à Tokyo. Les expériences ont été réalisées de nuit, en dehors des horaires d'ouverture de la boutique et avec toutes les fenêtres obstruées : le projet était encore confidentiel au moment des expériences. Les acquisitions ont

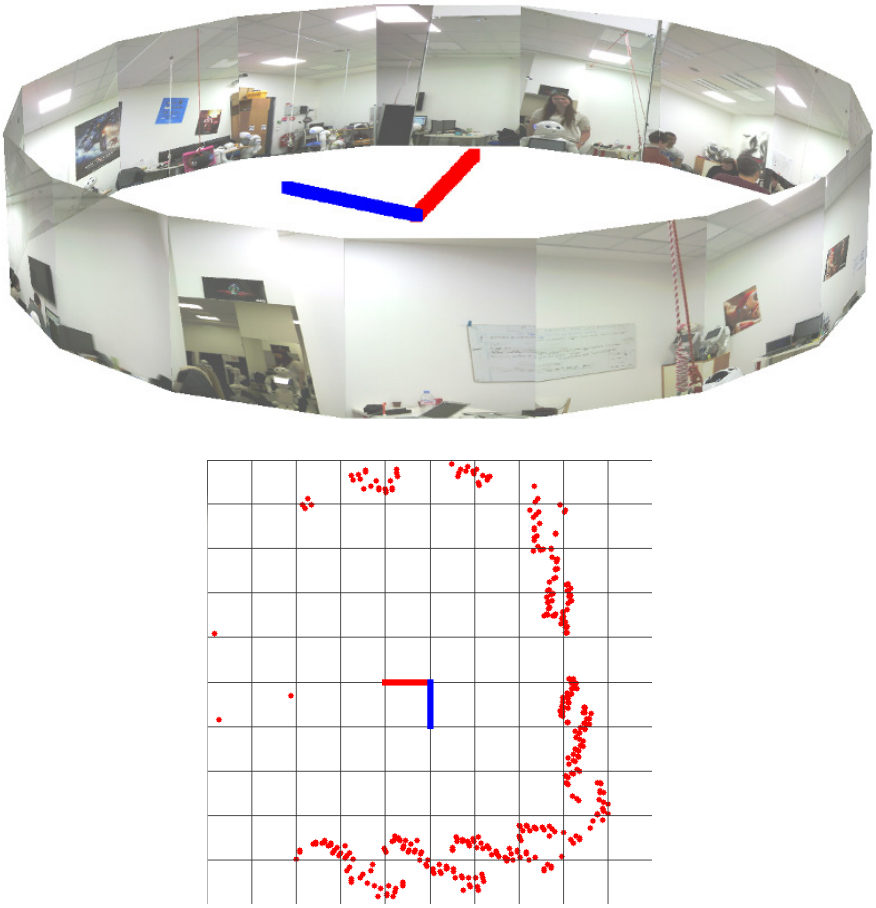


FIGURE B.2 – Exemple de nœud pris dans la salle de tests
Plateforme utilisée : Pepper

été réalisées à deux étages de la boutique : un étage d'exposition, qui contient notamment des modèles de téléphones ou d'accessoires exposés dans des vitrines et des étagères, et un étage réservé à la signature de contrats téléphoniques, qui contient donc une succession de bureaux et de sièges sans texture notables.

On voit sur la figure que les étagères portant les accessoires sont des éléments très texturés. On y détecte notamment de nombreux point d'intérêt. Leurs tailles et leur dispositions sont variables, si bien que l'environnement est propice à des calculs de corrélation. On voit cependant sur le profil que les murs latéraux de l'étage sont vus avec un bruit très fort, dû à la distance. Le recalage du profil par l'ICP est donc rendu plus difficile. En pratique, lorsque le robot circule dans cet espace, l'aspect des murs vus par la caméra 3D peut changer, puisque le bruit devient nettement plus important lorsque la distance dépasse quelques mètres.

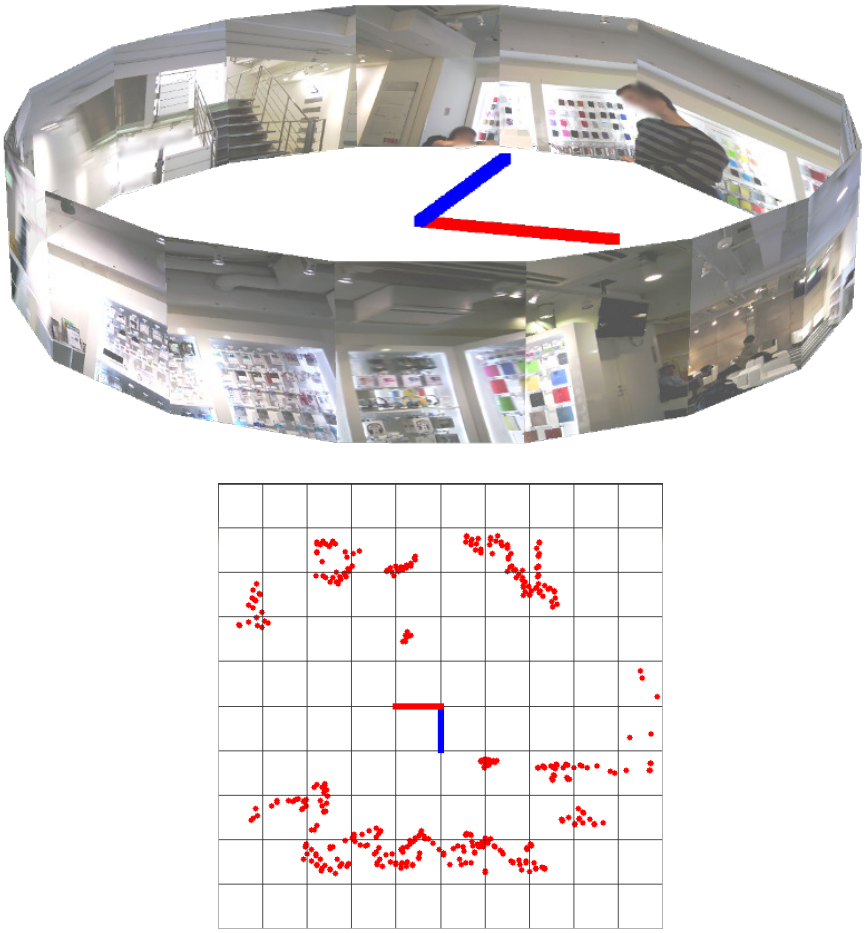


FIGURE B.3 – Exemple de nœud pris dans une boutique à Tokyo
Plateforme utilisée : Pepper

Bibliographie

"Aldebaran robotics." [Online]. Available : http://www.aldebaran.com/fr/

"Projet romeo." [Online]. Available : http://projetromeo.com/

"Strategic research agenda for robotics in europe." [Online]. Available : http://www.eu-robotics.net/cms/upload/PDF/SRA2020_0v42b_Printable_.pdf

"Syrobo." [Online]. Available : http://www.syrobo.org/

"Logiciel de simulation webots." [Online]. Available : http://www.cyberbotics.com/overview

M. Agrawal, K. Konolige, and M. Blas, "Censure : Center surround extremas for realtime feature detection and matching," *Computer Vision ECCV 2008*, pp. 102–115, 2008.

P. Alcantarilla, O. Stasse, S. Druon, L. M. Bergasa, and F. Dellaert, "How to localize humanoids with a single camera ?" *Autonomous Robot*, vol. 34, no. 1-2, pp. 47–71, 2013.

H. Andreasson, A. Treptow, and T. Duckett, "Localization for mobile robots using panoramic vision, local features and particle filter," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Apr. 2005, pp. 3348–3353.

A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, Oct. 2008.

A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Visual topological SLAM and global localization," in *IEEE International Conference on Robotics and Automation, ICRA*, May 2009, pp. 4300–4305.

H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 794–799.

——, “Real-time topometric localization,” in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2012, pp. 1635–1642.

H. Bay, T. Tuytelaars, and L. Van Gool, “Surf : Speeded up robust features,” *Computer Vision ECCV 2006*, pp. 404 – 417, 2006.

S. Bazeille and D. Filliat, “Incremental topo-metric SLAM using vision and robot odometry,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 4067–4073.

C. Beall, F. Dellaert, I. Mahon, and S. Williams, “Bundle adjustment in large-scale 3d reconstructions based on underwater robotic surveys,” in *OCEANS, 2011 IEEE - Spain*, June 2011, pp. 1–6.

H. Becerra, J. Courbon, Y. Mezouar, and C. Sagues, “Wheeled mobile robots navigation from a visual memory using wide field of view cameras,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 5693–5699.

S. Belongie and J. Malik, “Matching with shape contexts,” in *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, 2000, pp. 20–26.

S. Belongie, J. Malik, and J. Puzicha, “Shape context : A new descriptor for shape matching and object recognition,” in *NIPS*, vol. 2, 2000, p. 3.

J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

R. Biswas, B. Limketkai, S. Sanner, and S. Thrun, “Towards object mapping in non-stationary environments with mobile robots,” in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, vol. 1, 2002, pp. 1014–1019.

J. L. Blanco, J. Gonzalez, and J. A. Fernandez-Madrigal, “Subjective local maps for hybrid metric-topological SLAM,” *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 64–74, Jan. 2009.

F. Bonin-Font, A. Ortiz, and G. Oliver, “Visual navigation for mobile robots : A survey,” *J. Intell. Robotics Syst.*, vol. 53, no. 3, pp. 263–296, Nov. 2008.

G. Brooks, P. Krishnamurthy, and F. Khorrami, “Humanoid robot navigation and obstacle avoidance in unknown environments,” in *Control Conference (ASCC), 2013 9th Asian*, Jun. 2013, pp. 1–6.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief : Binary robust independent elementary features,” *Computer Vision ECCV*, pp. 778–792, 2010.

E. Cervera, A. A. Moughlbay, and P. Martinet, “Localization and navigation of an assistive humanoid robot in a smart environment,” Oct. 2012.

A. Chapoulie, P. Rives, and D. Filliat, "A spherical representation for efficient visual loop closing," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 335–342.

——, "Topological segmentation of indoors/outdoors sequences of spherical views," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4288–4295.

Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145 – 155, 1992, range Image Understanding.

Z. Chen and S. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 749–754, Jun. 2009.

A. Cherubini, F. Spindler, and F. Chaumette, "Autonomous visual navigation and laser-based moving obstacle avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. Early Access Online, 2014.

J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Robotics and Automation, ICRA Proceedings of the IEEE International Conference on*, April 2005, pp. 629–634.

C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, May 1968.

J. Courbon, Y. Mezouar, L. Eck, and P. Martinet, "Efficient hierarchical localization method in an omnidirectional images memory," in *IEEE International Conference on Robotics and Automation, ICRA*, May 2008, pp. 13–18.

J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet, "Visual navigation of a quadrotor aerial vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Oct. 2009, pp. 5315–5320.

J. Courbon, Y. Mezouar, and P. Martinet, "Autonomous navigation of vehicles from a visual memory using a generic camera model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 392–402, Sep. 2009.

J. Courbon, H. Korrapati, and Y. Mezouar, "Adaptive visual memory for mobile robot navigation in dynamic environment," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2012, pp. 873–878.

M. Cummins and P. Newman, "FAB-MAP : probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, Jan. 2008.

A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM : real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

J. Delfin, O. Mar, J.-B. Hayet, M. Castelan, and G. Arechavaleta, "An active strategy for the simultaneous localization and reconstruction of a 3D object from a humanoid platform," in *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nov. 2012, pp. 384–389.

F. Dellaert and M. Kaess, "Square root sam : Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

A. Diosi, A. Remazeilles, S. Segvic, and F. Chaumette, "Outdoor visual path following experiments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Oct. 2007, pp. 4265–4270.

A. Diosi, S. Segvic, A. Remazeilles, and F. Chaumette, "Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 870–883, Sep. 2011.

M. W. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 17, no. 3, p. 229, 2001.

C. Dune, A. Herdt, O. Stasse, P. B. Wieber, K. Yokoi, and E. Yoshida, "Cancelling the sway motion of dynamic walking in visual servoing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 3175–3180.

A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

W. Emery, D. Baldwin, and D. Matthews, "Maximum cross correlation automatic satellite image navigation and attitude corrections for open-ocean image navigation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 1, pp. 33–42, Jan. 2003.

D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3921–3926.

M. A. Fischler and R. C. Bolles, "Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

D. Galvez-Lopez and J. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

L. George and A. Mazel, “Humanoid robot indoor navigation based on 2D bar codes : Application to the NAO robot,” in *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)-Proceedings*, 2013.

L. Goncalves, E. Di Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlsson, and P. Pirjanian, “A visual front-end for simultaneous localization and mapping,” in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Apr. 2005, pp. 44–49.

D. Gouaillier, C. Collette, and C. Kilner, “Omni-directional closed-loop walk for NAO,” in *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Dec. 2010, pp. 448 –454.

D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, “Mechatronic design of NAO humanoid,” in *Proceedings of the IEEE international conference on Robotics and Automation*, ser. ICRA’09. Piscataway, NJ, USA : IEEE Press, 2009, pp. 2124–2129.

C. Harris and M. Stephens, “A combined corner and edge detector.” in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

E. Hashemi, M. Jadid, M. Lashgarian, M. Yaghobi, and R. Shafiei, “Particle filter based localization of the nao biped robots,” in *44th Southeastern Symposium on System Theory (SSST)*, Mar. 2012, pp. 168–173.

A. Hornung and M. Bennewitz, “Adaptive level-of-detail planning for efficient humanoid navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 997–1002.

A. Hornung, K. Wurm, and M. Bennewitz, “Humanoid robot localization in complex indoor environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 1690–1695.

V. Indelman, R. Roberts, C. Beall, and F. Dellaert, “Incremental light bundle adjustment,” in *BMVC*, 2012, pp. 1–11.

I. Jebari, S. Bazeille, E. Battesti, H. Tekaya, M. Klein, A. Tapus, D. Filliat, C. Meyer, S.-H. Ieng, R. Benosman, E. Cizeron, J.-C. Mamanna, and B. Pothier, “Multi-sensor semantic mapping and exploration of indoor environments,” in *2011 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, Apr. 2011, pp. 151–156.

C. Joly and P. Rives, “Self calibration of a vision system embedded in a visual SLAM framework,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2011, pp. 3320–3326.

M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “isam2 : Incremental smoothing and mapping using the bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

S. Kagami, K. Nishiwaki, J. Kuffner, K. Okada, M. Inaba, and H. Inoue, "Vision-based 2.5D terrain modeling for humanoid locomotion," in *IEEE International Conference on Robotics and Automation, Proceedings. ICRA*, vol. 2, Sep. 2003, pp. 2141–2146.

K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *IEEE International Conference on Robotics and Automation, ICRA*, vol. 2, Apr. 2004, pp. 1083–1090.

N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, "The vSLAM algorithm for robust localization and mapping," in *IEEE International Conference on Robotics and Automation, ICRA*, Apr. 2005, pp. 24–29.

K. Konolige and M. Agrawal, "FrameSLAM : from bundle adjustment to real-time visual mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

K. Konolige and J. Bowman, "Towards lifelong visual maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Oct. 2009, pp. 1156–1163.

K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 3041–3047.

N. Kwak, O. Stasse, T. Foissotte, and K. Yokoi, "3D grid and particle based SLAM for a humanoid robot," in *9th IEEE-RAS International Conference on Humanoid Robots, Humanoids*, Dec. 2009, pp. 62–67.

H. Lategahn and C. Stiller, "Vision-only localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. Early Access Online, 2014.

D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157.

C. Lutz, F. Atmanspacher, A. Hornung, and M. Bennewitz, "NAO walking down a ramp autonomously," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 5169–5170.

D. Maier, M. Bennewitz, and C. Stachniss, "Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1263–1269.

D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nov. 2012, pp. 692–697.

D. Maier, C. Lutz, and M. Bennewitz, "Integrated perception, mapping, and footstep planning for humanoid navigation among 3D obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 2658–2664.

J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and Vision Computing*, vol. 22, no. 10, pp. 761 – 767, 2004, british Machine Vision Computing 2002.

Y. Matsumoto, M. Inaba, and H. Inoue, “Visual navigation using view-sequenced route representation,” in *IEEE International Conference on Robotics and Automation, Proceedings*, vol. 1, Apr. 1996, pp. 83–88.

Y. Matsumoto, K. Ikeda, M. Inaba, and H. Inoue, “Exploration and navigation in corridor environment based on omni-view sequence,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS). Proceedings*, vol. 2, 2000, pp. 1505–1510.

M. Meilland, A. Comport, and P. Rives, “A spherical robot-centered representation for urban navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 5196–5201.

M. Milford, “Visual route recognition with a handful of bits,” in *Proceedings of Robotics Science and Systems Conference*. University of Sydney, 2012.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM : a factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the National conference on Artificial Intelligence*, 2002, pp. 593–598.

H. P. Moravec, “Sensor fusion in certainty grids for mobile robots,” *AI magazine*, vol. 9, no. 2, p. 61, 1988.

A. Moughlbay, E. Cervera, and P. Martinet, “Error regulation strategies for model based visual servoing tasks : Application to autonomous object grasping with nao robot,” in *12th International Conference on Control Automation Robotics Vision (ICARCV)*, Dec. 2012, pp. 1311–1316.

A. Murillo, J. Guerrero, and C. Sagues, “SURF features for efficient robot localization with omnidirectional images,” in *IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3901–3907.

K. Nishiwaki, J. Chestnutt, and S. Kagami, “Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor,” *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1251–1262, Sep. 2012.

A. Oliva and A. Torralba, “Building the gist of a scene : the role of global image features in recognition,” in *Progress in Brain Research*, 2006.

S. Osswald, A. Hornung, and M. Bennewitz, “Learning reliable and efficient navigation with a humanoid,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 2375–2380.

S. Osswald, A. Gorog, A. Hornung, and M. Bennewitz, "Autonomous climbing of spiral staircases with humanoids," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2011, pp. 4844–4849.

S. Osswald, A. Hornung, and M. Bennewitz, "Improved proposals for highly accurate localization using range and vision data," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 1809–1814.

F. K. P. Krishnamurthy, "GODZILA : a low-resource algorithm for path planning in unknown environments," *Journal of Intelligent and Robotic Systems*, vol. 48, no. 3, pp. 110– 115, 2005.

A. Pandey, K. Krishna, and H. Hexmoor, "Feature chain based occupancy grid SLAM for robots equipped with sonar sensors," in *International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS*, May 2007, pp. 283 –288.

E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV*, pp. 430–443, 2006.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB : an efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), IEEE International Conference on*, 2011, pp. 2564–2571.

Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo : system overview and integration," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, vol. 3, 2002, pp. 2478–2483.

N. Sawasaki, M. Nakao, Y. Yamamoto, and K. Okabayashi, "Embedded vision system for mobile robot navigation," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA*, May 2006, pp. 2693–2698.

G. Silveira, E. Malis, and P. Rives, "An efficient direct approach to visual SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 969–979, Oct. 2008.

J. Sivic and A. Zisserman, "Video google : a text retrieval approach to object matching in videos," in *Ninth IEEE International Conference on Computer Vision, Proceedings*, Oct. 2003, pp. 1470 –1477 vol.2.

C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard, "How to learn accurate grid maps with a humanoid," in *IEEE International Conference on Robotics and Automation, ICRA*, May 2008, pp. 3194–3199.

O. Tahri, H. Araujo, Y. Mezouar, and F. Chaumette, "Efficient iterative pose estimation using an invariant to rotations," *IEEE Transactions on Cybernetics*, vol. 44, no. 2, pp. 199–207, Feb. 2014.

R. Tellez, F. Ferro, D. Mora, D. Pinyol, and D. Faconti, "Autonomous humanoid navigation using laser and odometry data," in *8th IEEE-RAS International Conference on Humanoid Robots, Humanoids*, Dec. 2008, pp. 500–506.

S. Thrun, “A probabilistic on-line mapping algorithm for teams of mobile robots,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, Jan. 2001.

——, “Robotic mapping : A survey,” in *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

K. van de Sande, T. Gevers, and C. Snoek, “Color descriptors for object category recognition,” in *Proc. European Conference on Color in Graphics, Imaging and Vision, pg*, 2008, pp. 378–381.

E. Wirbel and A. de La Fortelle, “A method for localizing a robot in a localization plane,” Patent 069 203 EP GDE/ MAG, 2014.

E. Wirbel, F. Duriez, A. de La Fortelle, B. Steux, and S. Bonnabel, “Procédé d’estimation de la déviation angulaire d’un élément mobile relativement à une direction de référence,” French Patent 068 562 FR GDE/MAG, 2013.

E. Wirbel, B. Steux, S. Bonnabel, and A. De La Fortelle, “Humanoid robot navigation : From a visual slam to a visual compass,” in *Networking, Sensing and Control (ICNSC), 10th IEEE International Conference on*, April 2013, pp. 678–683.

E. Wirbel, S. Bonnabel, L. F. A. de, and F. Moutarde, “Humanoid robot navigation : Getting localization information from vision,” *Journal of Intelligent Systems*, vol. 23, no. 2, pp. 113–132, 2014.

X. Wu, Z. Shi, and Y. Zhong, “Detailed analysis and evaluation of keypoint extraction methods,” in *Computer Application and System Modeling (ICCASM), International Conference on*, vol. 2, Oct. 2010, pp. 562–566.

Y. Yamagi, J. Ido, K. Takemura, Y. Matsumoto, J. Takamatsu, and T. Ogasawara, “View-sequence based indoor/outdoor navigation robust to illumination changes,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Oct. 2009, pp. 1229–1234.

K. Yokoi, “Vision-based virtual information and semi-autonomous behaviours for a humanoid robot,” in *Intelligent Autonomous Systems*, 2003.

Localisation et navigation d'un robot humanoïde en environnement domestique

Résumé : Cette thèse traite du problème de la localisation et de la navigation de robots humanoïdes à bas coût dans un environnement dynamique non contraint. Elle a été réalisée en collaboration entre le laboratoire de robotique CAOR de Mines ParisTech et Aldebaran, dont les robots NAO et Pepper sont utilisés comme plateformes.

On verra ici comment il est possible de déduire des informations d'orientation et de position du robot malgré les fortes contraintes de puissance de calcul, de champ de vision et de généricité de l'environnement. L'environnement est représenté sous une forme topologique : les lieux sont stockés dans des nœuds, reliés par des transitions. On apprend l'environnement dans une phase préalable permettant de construire une référence.

Les contributions principales de la thèse reposent sur les méthodes de calcul de l'orientation et d'une mesure de position du robot à l'aide des caméras monoculaires à faible champ de vision, et leur intégration dans une structure topologique. Pour se localiser dans le graphe, on utilise principalement les données de vision fournies par les caméras monoculaires du robot, tout en laissant la possibilité de compléter à l'aide de caméras 3D. Les différentes méthodes de localisation sont combinées dans une structure hiérarchique qui permet à la fois d'améliorer la robustesse et de fusionner les données de localisation. Un contrôle de la trajectoire est également mis en place pour permettre d'effectuer de façon fiable les transitions d'un nœud à l'autre, et accessoirement fournir un système de retour pour la marche du robot. Les travaux de cette thèse ont été intégrés dans la suite logicielle d'Aldebaran, et testés intensivement dans différents environnements afin de valider les résultats obtenus et préparer une livraison aux clients.

Mots clés : SLAM, vision, robot humanoïde, robot Pepper, robot NAO

Localization and navigation of a humanoid robot in a domestic environment

Abstract : This thesis covers the topic of low cost humanoid robots localization and navigation in a dynamic unconstrained environment. It is the result of a collaboration between the Centre for Robotics of Mines ParisTech and Aldebaran, whose robots, NAO and Pepper, are used as experimental platforms.

We will describe how to derive information on the orientation and the position of the robot, under high constraints on computing power, sensor field of view and environment genericity. The environment is represented using a topological formalism : places are stored in vertices, and connected by transitions. The environment is learned in a preliminary phase, which allows the robot to construct a reference.

The main contribution of this PHD thesis lies in orientation and approximate position measurement methods, based on monocular cameras with a restricted field of view, and their integration into a topological structure. To localize the robot in the robot, we use mainly data provided by the monocular cameras of the robot, while also allowing extensions, for example with a 3D camera. The different localization methods are combined into a hierarchical structure, which makes the whole process more robust and merges the estimations. A trajectory control has also been developped in order to transition accurately from one vertex to another, and incidently to provide a feedback on the walk of the robot. The results of this thesis have been integrated into Aldebaran software suite, and thoroughly tested in various conditions, in order to validate the conclusions and prepare a client delivery.

Keywords : SLAM, vision, humanoid robot, Pepper robot, NAO robot

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech