Amélioration de performance de la navigation basée vision pour la robotique autonome : une approche par couplage vision/commande

Helène Roggeman

NNT : 2017SACLS497

Thèse de doctorat de l'Université Paris-Saclay préparée à l'Université Paris-Sud

Ecole doctorale n°580
Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Automatique

par

Hélène Roggeman

Amélioration de performance de la navigation basée vision pour la robotique autonome : une approche par couplage vision/commande

Thèse présentée et soutenue à Palaiseau, le 13 décembre 2017.

Composition du Jury :

M.	Philippe Bonnifait	Professeur UTC Compiègne	Président
M.	Roland Lenain	Directeur de recherche Irstea	Rapporteur
M.	Simon Lacroix	Directeur de recherche LAAS-CNRS	Rapporteur
Mme.	Yasmina Bestaoui	Professeure Université d'Evry	Examinateur
M.	Guy Le Besnerais	Maître de recherche ONERA, DTIS	Directeur de thèse
M.	Julien Marzat	Ingénieur de recherche ONERA, DTIS	Encadrant

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Amélioration de performance de la navigation basée vision pour la robotique autonome : une approche par couplage vision/commande

Mots clefs : Robotique mobile, Navigation autonome, Localisation basée vision, Commande prédictive, Couplage commande/vision, Qualité visuelle

Résumé : L'objectif de cette thèse est de réaliser des missions diverses de navigation autonome en environnement intérieur et encombré avec des robots terrestres. La perception de l'environnement est assurée par un banc stéréo embarqué sur le robot et permet entre autres de calculer la localisation de l'engin grâce à un algorithme d'odométrie visuelle. Mais quand la qualité de la scène perçue par les caméras est faible, la localisation visuelle ne peut pas être calculée de façon précise. Deux solutions sont proposées pour remédier à ce problème. La première solution est l'utilisation d'une méthode de fusion de données multi-capteurs pour obtenir un calcul robuste de la localisation. La deuxième solution est la prédiction de la qualité de scène future afin d'adapter la trajectoire du robot pour s'assurer que la localisation reste précise. Dans les deux cas, la boucle de commande est basée sur l'utilisation de la commande prédictive afin de prendre en compte les différents objectifs de la mission : ralliement de points, exploration, évitement d'obstacles. Une deuxième problématique étudiée est la navigation par points de passage avec évitement d'obstacles mobiles à partir des informations visuelles uniquement. Les obstacles mobiles sont détectés dans les images puis leur position et vitesse sont estimées afin de prédire leur trajectoire future et ainsi de pouvoir anticiper leur déplacement dans la stratégie de commande. De nombreuses expériences ont été réalisées en situation réelle et ont permis de montrer l'efficacité des solutions proposées.

Title : Performance improvement of vision-based navigation for autonomous robotics : a vision and control coupling approach

Keywords : Mobile robotics, Autonomous navigation, Vision-based localization, Model predictive control, Control and perception coupling, Visual quality

Abstract : The aim of this thesis is to perform various autonomous navigation missions in indoor and cluttered environments with mobile robots. The environment perception is ensured by an embedded stereo-rig and a visual odometry algorithm which computes the localization of the robot. However, when the quality of the scene perceived by the cameras is poor, the visual localization cannot be computed with a high precision. Two solutions are proposed to tackle this problem. The first one is the data fusion from multiple sensors to perform a robust computation of the localization. The second solution is the prediction of the future scene quality in order to adapt the robot's trajectory to ensure that the localization remains accurate. In the two cases, the control loop is based on model predictive control, which offers the possibility to consider simultaneously the different objectives of the mission : waypoint navigation, exploration, obstacle avoidance. A second issue studied is waypoint navigation with avoidance of mobile obstacles using only the visual information. The mobile obstacles are detected in the images and their position and velocity are estimated in order to predict their future trajectory and consider it in the control strategy. Numerous experiments were carried out and demonstrated the effectiveness of the proposed solutions.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Remerciements

Je voudrais tout d'abord remercier Julien et Guy pour m'avoir acceptée en thèse et m'avoir encadrée pendant ces trois années. Merci pour votre accompagnement et vos précieux conseils.

J'aimerais aussi remercier l'équipe COPERNIC, Anthelme, Martial, Alexandre, Julien, Aurélien pour leur disponibilité et leur aide lors des expérimentations.

Merci également à toute l'équipe du DTIS sans qui ces trois années auraient été beaucoup plus ennuyeuses. Merci pour tous ces bons moments en salle de pause, à jouer aux échecs, résoudre des énigmes ou discuter de sujets divers et variés.

Merci à mes camarades de promo'17, Calum, Guillaume, Joris et David et aux autres doctorants, apprentis, stagiaires et permanents de l'équipe : Maxime, Maxime, Maxime, Isabelle, Nicolas, David, Marcela, Sémy, Elyse, Oriane, Robin, Thibaut, Fred, Fabrice, Elise, Bertrand, Alexandre, Pauline, Philippe. J'espère que je n'oublie personne.

Je voudrais également remercier l'équipe de foot de l'ONERA pour ces pauses sportives bien sympathiques ainsi que Alice pour m'avoir régulièrement accompagnée sur le difficile trajet Palaiseau-Villebon - ONERA.

Et enfin, je voudrais remercier mes amis, ma famille et Benjamin, pour leur présence et leur soutien.

Table des matières

Notations

Acronymes

IMU : Inertial Measurement Unit, centrale inertielle
EKF : Extended Kalman Filter, filtre de Kalman étendu
SLAM : Simultaneous Localisation and Mapping
eVO : efficient Visual Odometer
RRT : Rapidly exploring Random Tree
PRM : Probabilistic Roadmap Method
RANSAC : RANdom SAmple Consensus
MPC : Model Predictive Control, commande prédictive

Caméras

C_l et C_r centre optique caméra gauche et droite
b ligne de base (ou baseline)
α distance focale
(u_0, v_0) est la projection du centre optique C_l dans le plan image gauche.
d disparité
K matrice caméra
$p = (u, v)$ point image 2D
P point 3D de coordonnées $Y = (x, y, z)^T$
$\mathcal{Y} = (Y_i)_{i \in [\![1,M]\!]}$ liste de points 3D avec M le nombre de points
$\Pi(Y)$ fonction de projection d'un point 3D Y
$\Pi^{-1}(\eta)$ fonction de triangulation à partir de $\eta = (u, v, d)$, la position du point dans l'image gauche et la disparité

Capteurs de profondeur

δ portée
β demi-angle d'ouverture

Incertitudes

Si X est une variable aléatoire,
σ_X^2 variance de X
Σ_X matrice de covariance de X
$|\Sigma_X|$ déterminant de la matrice de covariance Σ_X

$E[X]$ espérance de X
$E[X|Y]$ espérance de X conditionnée à Y
$p(X)$ distribution de probabilité sur X
$H(X)$ entropie associée à la variable X

Changements de repères

R matrice de rotation
t translation
P matrice de changement de repère

Cartes

$G(n)$ carte d'exploration à t_n
H hauteur
W longueur
r résolution

Système

t_n instant actuel
t_e période d'échantillonnage
$X = (x, y, \theta)^T$ vecteur d'état du robot
(x, y) position du robot sur le plan du sol
θ orientation du robot par rapport à l'axe x
$f(X, U)$ fonction du modèle du système
$\widehat{X}$ état estimé
$\tilde{X}$ état prédit

Commande

$U = (v, \omega)^T$ vecteur de commande
v vitesse linéaire
ω vitesse angulaire
U^* commande optimale
ω^* vitesse angulaire optimale
v_0 vitesse linéaire nominale
$v_{\min}$ vitesse linéaire minimale
$v_{\max}$ vitesse linéaire maximale
$\omega_{\min}$ vitesse angulaire maximale
$\omega_{\max}$ vitesse angulaire maximale
H_c horizon de commande

H_p horizon de prédiction
J fonction de coût
J_{expl} coût d'exploration
J_{wp} coût de ralliement de points
X_w point à rallier
J_{loc} coût de qualité de localisation
H_{loc} horizon de prédiction de scène
J_{u} coût de régulation de vitesse
J_{obs} coût d'évitement d'obstacles
$d_{\text{obs}}(X)$ distance entre la position X et l'obstacle le plus proche
$f_{\text{obs}}\left(d_{\text{obs}}(X)\right)$ fonction de pénalité sur la distance à l'obstacle le plus proche
d_{sec} distance de sécurité
d_{des} distance désirée
$Z = \left[\xi^T, r, h\right]^T$ description d'un objet mobile avec ξ^T position du cylindre, r rayon du cylindre et h hauteur du cylindre

Chapitre I

Introduction

L'exploration autonome de bâtiments par des plateformes robotiques mobiles présente un intérêt croissant car elle permet la réalisation de tâches nombreuses et diverses : missions de type Search-And-Rescue, recueil d'informations, surveillance de sites, etc... L'avantage principal d'exécuter ces missions sur des robots est d'éviter l'intervention humaine dans des zones à risques, par exemple dans le cas de risques d'incendie. Le fonctionnement en autonomie de ces plateformes permet de s'affranchir des difficultés pouvant survenir dans le cas de télé-opération de l'engin, comme les problèmes de perte de connexion ou de temps de latence.

L'utilisation de différents types de plateformes robotiques, terrestres ou aériennes, permet d'augmenter la capacité d'acquisition des informations dans l'environnement. Les robots terrestres ont l'avantage d'avoir une autonomie de fonctionnement plus importante que celle des robots aériens et ils sont plus faciles à manœuvrer. Ils possèdent, en outre, une charge utile importante. D'un autre coté, les drones présentent une plus grande mobilité par rapport aux robots terrestres, ils peuvent explorer des zones qui leur sont inaccessibles et peuvent ainsi obtenir des points de vue multiples. Cependant, ils ont une capacité de charge très limitée et leur manipulation est plus délicate.

Pour que ces plateformes puissent naviguer de façon autonome et sûre, différentes tâches embarquées doivent être exécutées en temps réel. Elles doivent tout d'abord avoir la capacité de percevoir l'environnement, afin d'obtenir des informations nécessaires pour pouvoir calculer la localisation de l'engin, cartographier l'environnement ou détecter des obstacles. A partir de ces différentes informations, le robot peut ensuite calculer la commande à envoyer aux moteurs afin de réaliser la mission définie. Il est indispensable d'obtenir des mesures fiables pour que ces différentes tâches soient accomplies de façon correcte et ainsi permettre au robot d'évoluer de façon sûre dans son environnement et de répondre aux objectifs de la mission.

Dans cette étude, on s'intéresse plus particulièrement à la précision de la localisation qui est indispensable pour pouvoir effectuer des déplacements de façon sécurisée, pour calculer des trajectoires qui évitent les obstacles, ainsi que pour calculer une reconstruction précise de l'environnement. De nombreuses méthodes existent pour obtenir la localisation d'un robot. Les missions considérées se déroulant dans des environnements intérieurs, l'utilisation d'un système de localisation global tel que le GPS n'est pas envisageable car il est peu fiable dans ce type d'environnement. La localisation doit alors être calculée en utilisant d'autres capteurs embarqués sur le robot. Parmi toutes les possibilités, les capteurs de vision sont un choix intéressant car ils sont légers, ce qui permet l'embarquement sur les drones et ils offrent une perception riche de l'environnement. Grâce à des algorithmes de traitement d'images, il est possible d'exécuter toutes

les tâches décrites (localisation, cartographie...) à partir des informations extraites dans les images.

Cependant, des limitations apparaissent dans l'utilisation des images pour les tâches considérées. En effet, quand la quantité d'information est insuffisante dans les images, les algorithmes ne peuvent pas fonctionner correctement et renvoient des résultats imprécis ou même parfois complètement aberrants. Or, un robot qui n'est pas correctement localisé ne peut plus se déplacer de façon sûre dans son environnement car il risque de heurter des obstacles. De plus, une mauvaise localisation de l'engin provoque des erreurs dans la reconstruction de l'environnement, ce qui ne permet pas une exploitation future de la carte obtenue.

Une première solution possible est de fusionner les données de localisation visuelle avec celles de capteurs complémentaires comme l'odométrie des roues ou les centrales inertielles afin d'améliorer la robustesse de la localisation. La limite de charge utile d'un drone conduit cependant à embarquer des centrales de précision faible, d'où une dérive rapidement importante, qui rend cette solution inapplicable dans ce cas.

Une deuxième solution est alors d'exploiter le couplage existant entre la commande et la vision. Généralement, on considère que les tâches d'estimation des paramètres et de commande peuvent être effectuées de manière séparée. Or, la commande est calculée en utilisant les paramètres estimés à partir des informations visuelles extraites dans les images, qui sont, dans le cas de scène peu informatives, de qualité insuffisante pour obtenir une estimation fiable. Par conséquent, la commande calculée risque d'être peu performante voire dangereuse pour le robot. Réciproquement, la commande envoyée au robot définit la trajectoire à suivre et donc la scène qui sera perçue par les caméras. Par conséquent, la commande a une influence sur la qualité des images reçues et donc sur l'estimation des paramètres. La qualité de l'estimation a un impact sur l'efficacité de la commande et la commande réalisée influe sur la qualité future de l'estimation. Les deux tâches de localisation et de commande sont donc dépendantes l'une de l'autre.

La stratégie fondée sur ce couplage est alors d'anticiper ce qui sera vu dans les images futures afin d'adapter la trajectoire réalisée par le robot dans le but de garantir que la localisation visuelle restera précise tout au long du déplacement.

L'objectif de cette thèse est de mettre en place une boucle de perception-commande dans le but d'effectuer des missions diverses sur des plateformes robotiques équipées de capteurs visuels. Un banc stéréo composé de deux caméras permet d'obtenir la localisation de l'engin et un capteur RGB-D permet d'obtenir une modélisation de l'environnement proche. Les différentes missions qui peuvent être considérées sont :

— la navigation par points de passage

— l'exploration de zones inconnues

— l'évitement d'obstacles fixes ou mobiles

Certains de ces objectifs doivent être réalisés simultanément, par exemple, pendant l'exploration de zones inconnues, on veut que le robot évite également les obstacles. L'environnement dans lequel le robot évolue peut présenter des zones peu texturées, où le calcul de la localisation visuelle peut être difficile. Il est proposé d'estimer la qualité de la

scène future et de la prendre en compte dans la commande du robot afin de s'assurer que la localisation reste précise tout au long de la mission. Les algorithmes développés ont été embarqués sur des robots terrestres et de nombreuses expériences ont été réalisées afin de valider ces algorithmes dans des situations réelles.

Dans le chapitre II, nous détaillons le matériel et les méthodes utilisés pour réaliser les expérimentations.

Dans le chapitre III, nous présentons l'état de l'art dans les domaines de la perception pour la robotique, de la commande robotique et du couplage entre la perception et la vision.

Dans le chapitre IV, nous décrivons une première boucle de perception-commande développée pour effectuer des missions d'exploration autonome. Pour s'assurer que la localisation reste correcte, une fusion de données multi-capteurs est réalisée par un filtre de Kalman à partir des données de l'odométrie visuelle, des odomètres des roues et de la centrale inertielle. La chaîne de perception-commande est complétée par un module de reconstruction de l'environnement qui construit la carte d'occupation de la pièce et un module de commande, basé sur la commande prédictive, qui permet de calculer la commande optimale à donner au robot en fonction des objectifs de la mission. Nous présentons les résultats d'expériences menées sur un robot terrestre afin de valider la boucle de commande développée. Ces travaux ont débouché sur la publication de l'article [Roggeman et al., 2014].

Le chapitre V porte sur la solution de couplage vision/commande. Nous décrivons deux méthodes développées pour estimer la qualité d'une scène future pour la localisation visuelle et leur validation expérimentale. Nous expliquons l'intégration d'une de ces méthodes dans une boucle de commande pour réaliser des missions de ralliement de points dans des environnements où la localisation est susceptible d'être peu précise à cause de zones peu texturées. Nous exposons les résultats des expériences réalisées en situation réelle. L'article [Roggeman et al., 2016] présente ces différents développements et résultats.

Dans le chapitre VI, nous présentons les résultats d'expérimentations avancées sur des robots équipés uniquement de caméras. Deux types de missions sont étudiées : l'exploration autonome d'un environnement présentant des zones peu texturées. Nous détaillons la boucle de commande développée qui intègre le critère sur la qualité future d'une scène pour la localisation et les résultats des expérimentations réalisées. Ces travaux ont fait l'objet de la publication d'un article de conférence [Roggeman et al., 2017a] et de la soumission d'un article de journal [Roggeman et al., 2017b], qui intègre ces résultats ainsi que ceux du chapitre précédent. Le deuxième type de mission étudié dans ce chapitre est le ralliement de points de passage par un robot terrestre avec évitement d'obstacles mobiles. Nous présentons la chaîne de perception-commande mise en place, qui permet de détecter les objets mobiles et de prédire leur trajectoire future afin d'être pris en compte dans le calcul de la commande. Nous exposons les résultats des expérimentations effectuées. L'article [Roggeman et al., 2017c] décrit ces résultats ainsi que ceux obtenus sur des drones, réalisés par d'autres membres de l'équipe de recherche.

Chapitre II

Matériel et méthodes

Sommaire

Ce chapitre présente les méthodes et le matériel utilisés dans cette thèse.

La section II.1 décrit les conventions et le formalisme de représentation choisis dans ces travaux de thèse. La section II.2 présente le matériel employé, robots et capteurs. La section II.3 décrit les outils informatiques utilisés.

II.1 Représentation dans l'espace

Cette partie décrit le formalisme utilisé dans cette thèse pour représenter l'état du robot, les rotations et les transformations dans l'espace.

II.1.1 Convention de représentation d'un robot

Par convention, le repère lié au robot, appelé repère robot, possède une origine au centre de gravité du robot, l'axe x est vers l'avant, l'axe y vers la gauche et l'axe z vers le haut, comme illustré dans la figure II.1. Les angles de rotation autour de ces axes sont l'angle de lacet, noté θ_z, l'angle de roulis, noté θ_x et l'angle de tangage θ_y.

Par convention, le repère fixé pour une caméra est : l'origine est le centre optique de la caméra, l'axe z est vers l'avant, l'axe x vers la droite et l'axe y vers le bas.

Comme les caméras sont placées sur les robots de telle façon que l'axe optique est colinéaire à l'axe du robot, un changement de repère est nécessaire pour exprimer un point soit dans le repère caméra, soit dans le repère robot.

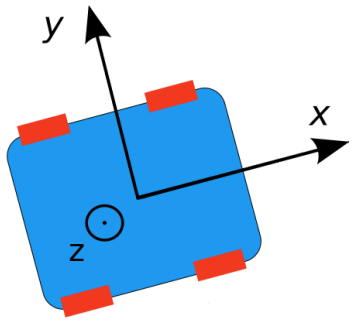


FIGURE II.1 – Repère robot

II.1.2 Représentation des rotations

On note R une matrice de rotation. Elle s'exprime à partir des angles d'Euler (θ_z,θ_y et θ_x) par la formule :

$$R = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) \tag{II.1}$$

avec,

$$R_z(\theta_z) = \begin{pmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{II.2}$$

$$R_y(\theta_y) = \begin{pmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{pmatrix} \tag{II.3}$$

$$R_x(\theta_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{pmatrix} \tag{II.4}$$

On a choisi d'utiliser les angles d'Euler et les matrices de rotation comme représentation des rotations mais d'autres représentations existent. On peut par exemple utiliser les quaternions qui présentent l'avantage d'éviter les singularités apparaissant avec les angles d'Euler, ou plus généralement le groupe SO(3)des rotations dans l'espace.

II.1.3 Transformations dans l'espace

Une transformation dans l'espace est donnée par un couple (rotation, translation). Soit un point de coordonnées X_0 dans le repère $\mathcal{R}_0$, ses coordonnées dans le repère $\mathcal{R}_1$ sont obtenues par la formule :

$$X_1 = R_{10}X_0 + t_{10} \tag{II.5}$$

t_{10} est la position du centre du repère $\mathcal{R}_0$ exprimé dans le repère $\mathcal{R}_1$, les colonnes de la matrice R_{10} sont les vecteurs de base du repère $\mathcal{R}_0$ exprimés dans le repère $\mathcal{R}_1$.

Pour un point de coordonnées X, on note $(X, 1)^T$ ses coordonnées augmentées. Les coordonnées homogènes de ce vecteur sont définies comme l'ensemble des multiples du vecteur augmenté $\tilde{X} = \lambda.(X, 1)^T$ pour tout λ réel. En coordonnées homogènes, on peut exprimer une transformation de rotation R et de translation t par une matrice P de dimension 4 :

$$P_{10} = \begin{pmatrix} R_{10} & t_{10} \\ 0 & 1 \end{pmatrix} \tag{II.6}$$

Ainsi, le point de coordonnées X_0 dans le repère R_0 est exprimé dans le repère R_1 par :

$$\tilde{X}_1 = P_{10}\tilde{X}_0 \tag{II.7}$$

Cette formule est équivalente à la formule (II.5). Elle présente l'avantage d'être exprimée comme une équation linéaire, ce qui facilite les calculs de composition et d'inverse des transformations.

II.2 Matériel

Cette partie décrit le matériel utilisé pour les expériences. La section II.2.1 décrit les différents capteurs montés sur les robots : caméras RGB, bancs stéréo, capteurs de profondeurs, centrales inertielles. Les robots utilisés pour effectuer les expériences sont présentés dans la section II.2.2.

II.2.1 Capteurs embarqués

On distingue généralement deux types de capteurs :

— les capteurs proprioceptifs qui donnent une estimation de l'état du robot par rapport à des mesures des paramètres internes du système (odomètres des roues, centrales inertielles, ...)

— les capteurs extéroceptifs qui prennent des mesures dans l'environnement extérieur (caméras, LIDARs, ...)

Les sections qui suivent décrivent les différents capteurs embarqués sur les robots.

II.2.1.1 Caméras RGB et bancs stéréo

Les bancs stéréo sont composés de deux caméras identiques et synchronisées, séparées par une distance d'une vingtaine de centimètres, voir l'image II.2. Chaque robot est équipé d'un banc stéréo possédant ses caractéristiques propres, les détails sont exposés dans la section II.2.2.

Les caméras utilisées sont des caméras RGB uEye avec des tailles image de 640 pixels x 512 pixels. Les distances focales sont de quelques millimètres. La fréquence d'acquisition des images est de l'ordre de 10 à 20 Hz.

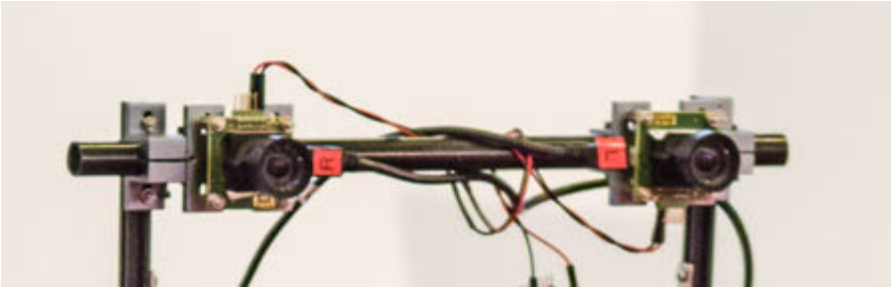


FIGURE II.2 – Banc stéréo

Les bancs sont calibrés afin de connaître les paramètres des caméras et du banc stéréo : paramètres intrinsèques des caméras (focale, point à l'origine), coefficients de distorsion, transformation entre les deux caméras sur le banc, voir [Hartley and Zisserman, 2003].

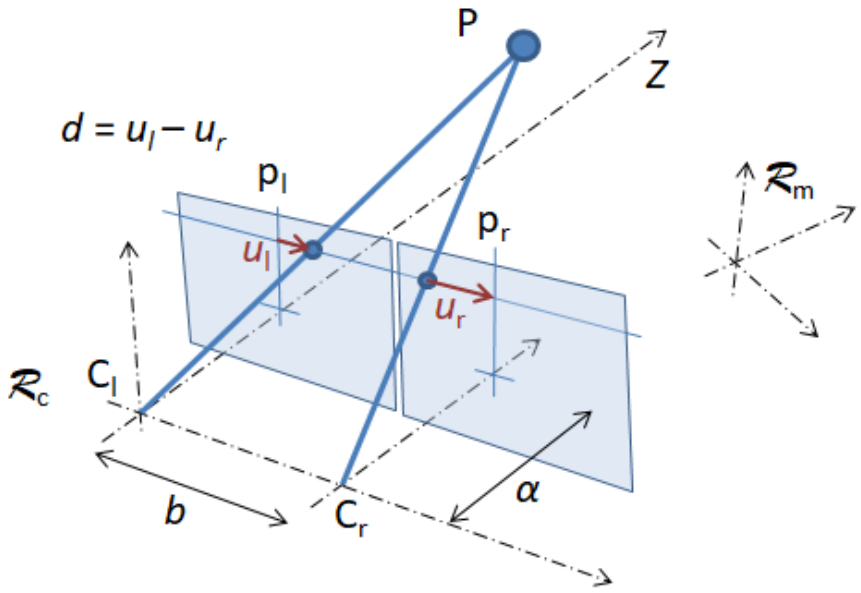


FIGURE II.3 – Schéma d'un système stéréoscopique

Grâce à la calibration du banc stéréo, on peut utiliser la configuration rectifiée présentée dans la figure II.3, comme représentation du système stéréo. Le repère de référence est le repère de la caméra gauche, centré sur le centre optique C_l. L'axe optique est l'axe Z. La caméra droite est positionnée à une distance b de la caméra gauche, le long de l'axe X. Cette distance est appelée ligne de base. La projection d'un point 3D P de coordonnées $Y = (x, y, z)^T$ sur le plan de l'image gauche s'écrit :

$$p_l = \Pi(Y) = \begin{pmatrix} \frac{\alpha}{z} \cdot x + u_0 \\ \frac{\alpha}{z} \cdot y + v_0 \end{pmatrix} \tag{II.8}$$

En utilisant les coordonnées homogènes, on obtient :

$$\tilde{p}_l = (u_l, v_l, 1)^T = z^{-1} K \cdot Y \tag{II.9}$$

avec K, la matrice caméra

$$K = \begin{pmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{II.10}$$

α est la distance focale et le point (u_0, v_0) est la projection du centre optique C_l dans le plan image gauche. Dans une configuration rectifiée, la projection du point P sur l'image droite se trouve sur la même ligne, appelée ligne épipolaire. La différence de position entre les deux points sur l'axe horizontal est appelée disparité et est notée d. La profondeur du point z et la disparité d sont inversement proportionnelles :

$$z = \frac{-b.\alpha}{d} \tag{II.11}$$

II.2.1.2 Capteurs de profondeur

Les capteurs de profondeur utilisés sont l'Asus Xtion ou la Microsoft Kinect, voir photo II.4. Ce sont des capteurs actifs RGB-D, composés d'une caméra classique RGB, d'un émetteur et d'un récepteur infrarouges. Ils sont branchés au processeur en USB. Pour la Xtion, la prise USB permet d'alimenter l'appareil tandis qu'il faut une alimentation externe pour la Kinect.

Seule la fonctionnalité infrarouge de ce capteur nous intéresse. Elle est modélisée ici comme une caméra centrale de demi-angle d'ouverture $\beta = 0.5$ rad et une portée maximale de $\delta = 4$ à 5 mètres. Leur fréquence d'acquisition est de 30 Hz. Les images RGB reçues ne sont pas utilisées.

Ces capteurs permettent d'obtenir une image 3D de la scène dans leur champ de vue sous la forme de nuage de points. Grâce à l'émission et à la réception d'un motif formé d'un ensemble de points lumineux dans le domaine infra-rouge, l'information 3D est disponible même lorsque la scène est peu texturée, comme cela arrive notamment en environnement intérieur ou en éclairage faible. Cependant, comme les longueurs d'ondes infrarouge utilisées sont trop proches de celles émises par le soleil, le récepteur infrarouge est aveuglé en environnement extérieur.

FIGURE II.4 – ASUS Xtion

II.2.1.3 Centrales inertielles

Une centrale inertielle est composée de trois accéléromètres et de trois gyromètres. Les accéléromètres permettent de mesurer l'accélération linéaire dans les trois directions de l'espace x, y et z. Les gyromètres permettent d'obtenir les vitesses angulaires autour des trois axes de rotation (roulis, tangage et lacet).

La centrale inertielle calcule la position (x, y, z) et l'orientation (θ_x, θ_y, θ_z) par rapport à la position précédente à partir des mesures de ces six capteurs. L'estimation est généralement réalisée par un filtre, comme le filtre de Kalman qui fournit une estimation de la position, de l'orientation et de la vitesse. La méthode de localisation est intégrative, le calcul de la position et de l'orientation par rapport au point d'origine est effectué en combinant au cours du mouvement les déplacements calculés pas à pas.

Les centrales inertielles embarquées sur les petits robots que nous utilisons sont des centrales de dimensions réduites, fondées sur des composants MEMS, qui présentent une dérive qui augmente rapidement pendant le déplacement avec l'intégration des mesures réalisées pour calculer la position. Elles ne peuvent pas être utilisées seules pour effectuer un calcul de localisation précis.

II.2.1.4 Odométrie des roues

Les roues des robots terrestres possèdent des encodeurs qui permettent de compter le nombre de tours de roues effectué et ainsi d'évaluer la distance parcourue. Un calcul permet ensuite d'estimer la position et l'orientation du robot par rapport à son point de départ.

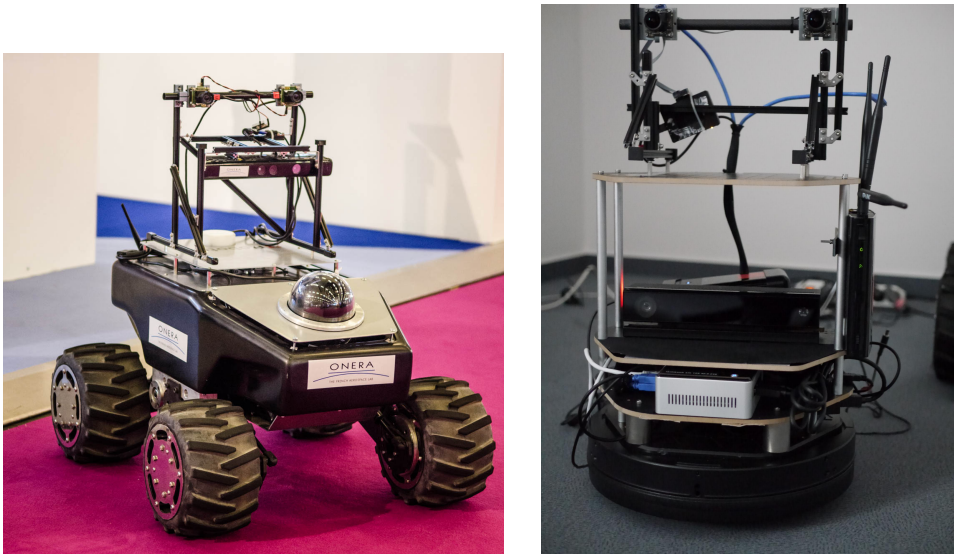
L'odométrie des roues présente également une dérive qui augmente au cours du déplacement. Cette dérive provient notamment des pertes d'adhérence de la roue, elle est donc est assez variable en fonction de la nature du sol (sol lisse ou rugueux) et de la trajectoire suivie par le robot (trajectoire plutôt rectiligne ou comportant de nombreux virages).

II.2.2 Plateformes robotiques

Cette section décrit plus précisément les deux plateformes robotiques qui ont été utilisées pour les expériences réalisées dans le cadre de cette thèse.

Le robot Summit XL (figure II.5a) est un robot à quatre roues non directionnelles, conçu par la société Robotnik. Son empreinte au sol est de 0.6 m x 0.65 m. Sa hauteur avec les capteurs embarqués, comme présenté sur la photo est de 0.7 m. Il possède un processeur embarqué Intel quad-core i7 et un GPU Nvidia (GT640).

Le robot Turtlebot (figure II.5b) est un robot à trois roues : deux roues motrices et une roue folle. Les roues sont non directionnelles. Le robot a une base ronde de 0.35 m de diamètre et il mesure 0.5 m de hauteur avec ses capteurs. L'ordinateur embarqué est un Intel NUC Kit NUC5i7RYH qui possède un processeur dual-core Intel i7-5557U. Le processeur est moins puissant que celui du Summit XL. Certains algorithmes ont ainsi dû être exécutés sur une station déportée qui possède un processeur quad-core Intel i7-4710MQ.



(a) Robotnik Summit XL (b) Turtlebot

FIGURE II.5 – Plateformes robotiques utilisées pour les expériences

Les deux robots embarquent les mêmes types de capteurs : des odomètres sur les roues, une centrale inertielle, un capteur de profondeur Microsoft Kinect et un banc stéréo composé de deux caméras. Sur le Summit XL, la distance focale des caméras est de 2.8 mm et elles sont séparées par une ligne de base de 32 cm. Sur le Turtlebot, la distance focale est de 4.0 mm et elles sont séparées par une ligne de base de 21 cm.

Les robots sont commandés par leur vitesse linéaire v et leur vitesse angulaire ω. Pour ces deux robots, les roues sont non directrices. Les rotations sont réalisées par l'application d'un différentiel de vitesse sur les roues.

Les vitesses linéaire et angulaire sont liées aux vitesses de rotation des roues gauche

et droite, notées respectivement ω^l et ω^r, par les équations :

$$\begin{cases} v = \frac{r}{2}(\omega^l + \omega^r) \\ \omega = \frac{r}{2L}(\omega^r - \omega^l) \end{cases} \tag{II.12}$$

avec L, la demi-longueur de l'axe et r, le rayon des roues. On fait ici l'hypothèse que le centre de gravité est confondu avec le centre du robot. voir figure II.6.

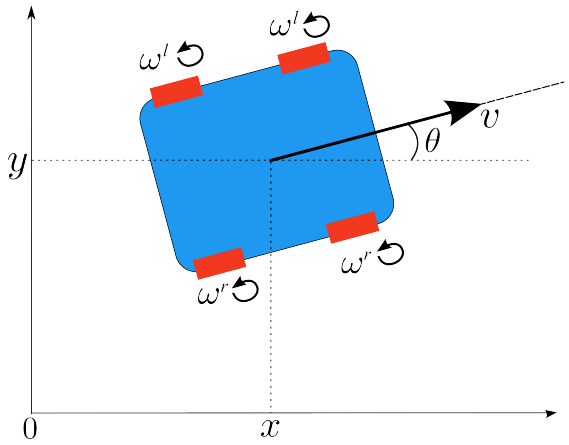


FIGURE II.6 – Schéma du robot

Le robot Summit XL présente l'avantage d'avoir un processeur plus puissant ce qui permet d'embarquer et d'exécuter tous les algorithmes alors qu'une station déportée est nécessaire pour le Turtlebot. De plus, ses roues lui permettent d'aller dans des terrains accidentés. Le robot Turtlebot est plus petit et plus maniable, il est plus apte à se déplacer dans les environnements intérieurs et encombrés que le Summit XL.

II.3 Logiciel

Cette section décrit les logiciels et outils utilisés pour le développement des algorithmes embarqués sur le robot.

Les robots sont installés avec le middleware ROS qui facilite la programmation et l'embarquement des algorithmes sur le robot, décrit en section II.3.1.

La partie II.3.2 décrit les librairies open-source dont certaines fonctions sont utilisées dans les algorithmes développés.

II.3.1 ROS

Robot Operating System (ROS) [http://www.ros.org/] est un ensemble d'outils informatiques open source pour le développement d'applications robotiques. Son fonctionnement est distribué, composé de différents modules codés en C++ ou en Python.

FIGURE II.7 – ROS

Le fonctionnement de ROS repose sur un système de nœuds communiquant entre eux via des topics ou des services, s'échangeant des messages standardisés. Un nœud est un processus qui accompli une tâche spécifique. Il peut communiquer avec les autres nœuds en publiant des messages sur un topic. Tous les nœuds peuvent souscrire aux topics et accéder aux messages qui ont été publiés. Un nœud peut envoyer une requête directement à un autre nœud en utilisant un service. Le schéma II.8 illustre le fonctionnement de ROS.

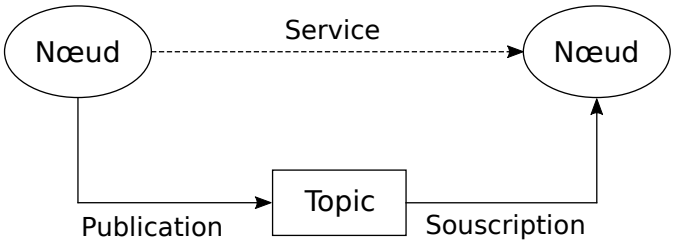


FIGURE II.8 – Le concept de fonctionnement de ROS

Ce mode de fonctionnement permet une grande modularité dans le développement des systèmes robotiques et permet d'embarquer facilement les nœuds développés sur des plateformes hétérogènes.

II.3.2 Librairies

OpenCV (Open Computer Vision) [http://opencv.org/] est une bibliothèque graphique libre qui fournit des fonctions pour le traitement d'images en temps réel. Elle propose des fonctions pour le traitement bas niveau des images (histogramme, lissage, filtrage, segmentation...), pour le traitement vidéo (détection de formes, de mouvement, de points d'intérêts, flot optique...). La plupart des fonctions proposées sont issues de l'état de l'art en vision par ordinateur.

PCL (Point Cloud Library) [http://www.pointclouds.org/] est une librairie open source qui fournit des fonctions pour le traitement des nuages de points et de la géométrie 3D. Elle propose des algorithmes portants sur la détection d'amers, la reconstruction de surface, la segmentation, le calcul de modèle... Ces fonctions sont très utiles pour la perception dans le domaine de la robotique.

OctoMap [https://octomap.github.io/] [Hornung et al., 2013] est une librairie qui fournit des structures de données et des algorithmes de cartographie en C++ particulièrement adaptés à la robotique. L'environnement est modélisé sous la forme d'une carte d'occupation 3D, en utilisant une structure d'octrees. Pour chaque élément de la

carte, trois états sont possibles : occupé, libre ou inconnu. La figure II.9 montre un exemple de carte obtenue. L'implémentation offre de nombreuses possibilités : la carte peut être mise à jour à tout instant si de nouvelles mesures capteurs sont acquises, la taille et la résolution de la carte sont modifiables en cours d'exécution. La modélisation et la mise à jour sont probabilisées, ce qui permet de prendre en compte le bruit sur la mesure. Enfin, plusieurs capteurs et/ou robots peuvent contribuer simultanément à la construction de la même carte.

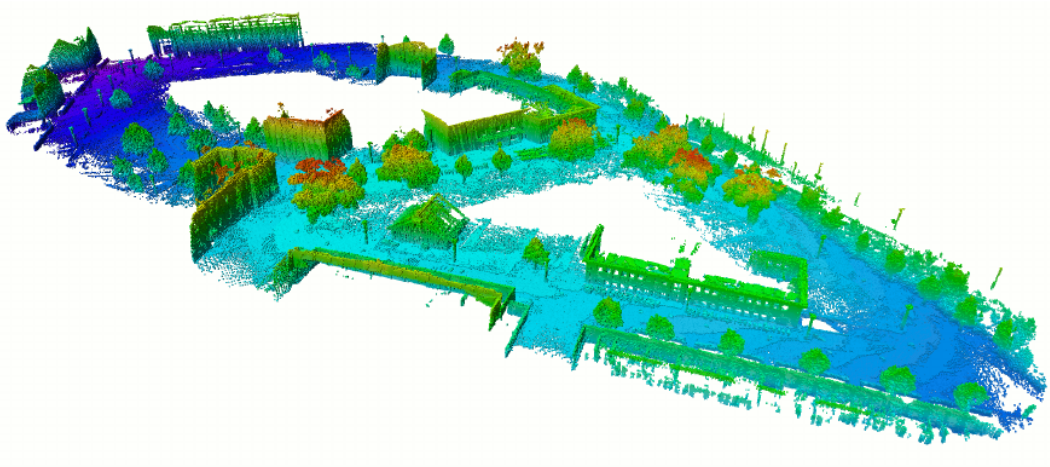


FIGURE II.9 – Exemple de la reconstruction obtenue d'un environnement extérieur avec une résolution de 0.2 m. Seuls les espaces occupés sont représentés, l'échelle de couleur permet de visualiser la hauteur des voxels. Illustration issue de [Hornung et al., 2013].

NumPy [http://www.numpy.org/] est une bibliothèque de fonctions pour le calcul vectoriel dans le langage python.

NLopt [https://nlopt.readthedocs.io/en/latest/] est une bibliothèque de python qui fournit des algorithmes d'optimisation non linéaire.

Chapitre III

Etat de l'art

Sommaire

Ce travail s'inscrit dans le contexte de la vision pour la robotique. A partir des images reçues par le système, des tâches de localisation visuelle et de reconstruction de l'environnement peuvent être réalisées. Quand les deux tâches sont réalisées en simultané, on parle de SLAM (Silmultaneous Localization And Mapping). Les travaux portant sur les techniques de localisation à partir des informations visuelles sont décrits dans la section III.1.

A partir des informations de localisation et de cartographie, le robot va pouvoir évoluer dans son environnement et accomplir diverses missions. La commande envoyée

aux actionneurs du robot doit pouvoir prendre en compte la multitude d'informations disponibles afin de réaliser correctement la tâche demandée. Les stratégies de commande utilisées pour exécuter ces missions sont décrites dans la section III.2.

Une précision élevée de l'estimation de la localisation et de la reconstruction de l'environnement est nécessaire pour le bon déroulement de ces missions. Le robot doit pouvoir connaître son environnement et sa position de façon fiable pour pouvoir se déplacer en toute sécurité. Il faut donc que les observations effectuées par le robot soient suffisamment riches en information pour permettre l'estimation précise de la localisation. La qualité des informations reçues par les caméras dépend de la nature des scènes perçues et donc de la trajectoire suivie par le robot. Pour réaliser une commande sûre, il est nécessaire d'avoir une estimation précise des paramètres de localisation et de cartographie, ce qui dépend de la qualité des images reçues. Les problématiques de couplage entre la perception et la commande sont décrites dans la section III.3.

III.1 Perception embarquée

Au milieu des années 80, les chercheurs en robotique ont constaté que pour un robot équipé de capteurs se déplaçant dans un environnement inconnu, les problématiques de construction d'un modèle de l'environnement et de l'estimation de la localisation étaient indissociables. De nombreux travaux ont porté sur la résolution conjointe de ces deux problèmes, approche appelée aujourd'hui SLAM pour Simultaneous Localization And Mapping [Durrant-Whyte and Bailey, 2006, Bailey and Durrant-Whyte, 2006]. La figure III.1 donne une représentation schématique du problème du SLAM.

La problématique était notamment de démontrer la convergence du SLAM, c'est-à-dire la possibilité de réduire l'incertitude de localisation jusqu'à zéro au cours du déplacement du robot dans un environnement donné, inconnu au départ de la mission. La plupart des expérimentations de SLAM utilisaient alors des capteurs actifs fournissant une information télémétrique, comme des capteurs SONAR (et plus tard des LIDARs ou des caméra RGB-D). Le SLAM visuel, fondé uniquement sur l'utilisation de caméras passives, apparaît vers le milieu des années 2000. Une revue très récente sur les algorithmes de SLAM visuel est disponible dans [Cadena et al., 2016].

La partie III.1.1 décrit le SLAM basé sur les capteurs actifs et le SLAM visuel, monoculaire et stéréo. Dans cette thèse, on s'intéresse tout particulièrement à la localisation visuelle à partir d'un système stéréo. La partie III.1.2 présente l'odométrie visuelle. La partie III.1.3 décrit les différentes étapes de fonctionnement d'un algorithme d'odométrie visuelle stéréo. La partie III.1.4 présente eVO, l'algorithme d'odométrie visuelle utilisé dans les expérimentations.

III.1.1 SLAM

Le SLAM a d'abord été développé en utilisant des capteurs actifs. Ces capteurs permettent d'obtenir une information de profondeur de l'obstacle le plus proche dans une direction donnée par la projection et la réception d'une onde. La distance est calculée

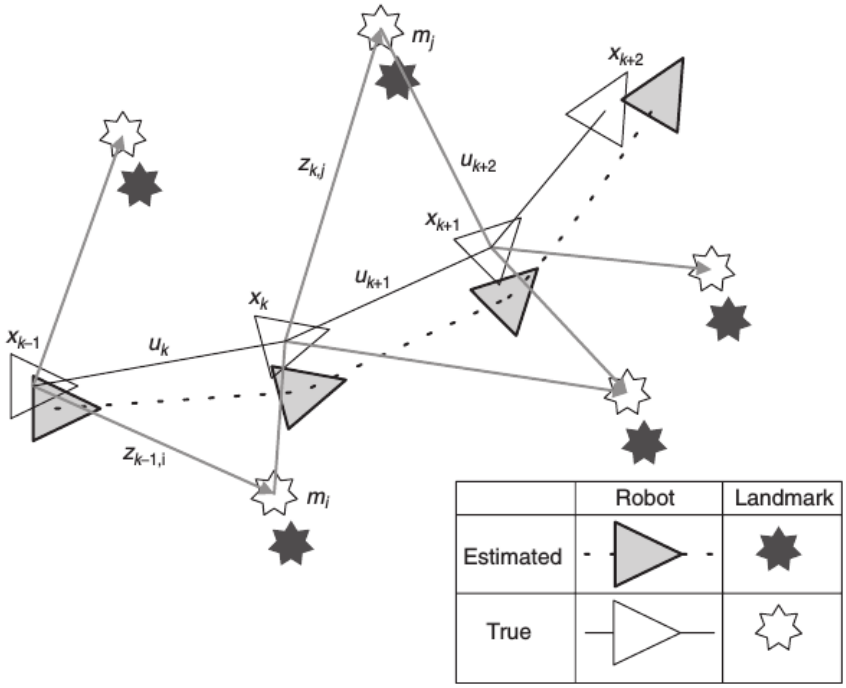


FIGURE III.1 – Schéma représentant le problème du SLAM : estimation simultanée des positions successives du robot et de la position des amers dans l'environnement. Illustration issue de [Durrant-Whyte and Bailey, 2006].

en mesurant le temps écoulé entre l'émission et la réception du signal réfléchi par l'objet.

Le SLAM a été étudié avec des SONARs [Crowley, 1989] mais ce capteur présente un bruit de mesure important et sa portée est limitée à quelques mètres. Il a ensuite été développé avec des capteurs LIDARs [Leonard and Durrant-Whyte, 1991, Smith et al., 1990]. Des missions de navigation avec des drones ont été réalisées à partir d'un algorithme de SLAM LIDAR par [He et al., 2008]. Dans ces études, les LIDARs utilisés sont équipés d'un système de balayage permettant d'avoir une information de profondeur sur un ensemble de points avec une cadence élevée. Ce capteur présente l'avantage d'avoir une portée importante, de plusieurs dizaines de mètres et une précision de mesure de l'ordre du centimètre. Cependant, la plupart des capteurs LIDARs ne peuvent effectuer le balayage que sur une nappe, ce qui donne un modèle de l'environnement uniquement sur un plan. Sur d'autres capteurs LIDARs, il est possible de réaliser plusieurs nappes mais la représentation de l'environnement obtenue reste discrète.

Le SLAM a également été étudié avec des capteurs RGB-D, présentés en section II.2.1.2. Des missions de navigation sur drone ont été réalisées avec ce type de capteur [Bachrach et al., 2012]. La portée de ce capteur est également limitée à quelques mètres et l'utilisation est essentiellement restreinte aux environnements intérieurs.

Le SLAM a ensuite été étudié avec des capteurs passifs tels que les caméras. Les caméras ont l'avantage de consommer moins d'énergie que les capteurs actifs car elles captent la lumière émise par des sources externes. Cependant, il faut s'assurer que les conditions d'éclairement de l'environnement permettent de percevoir des images suffisamment contrastées pour être exploitables alors que les capteurs actifs perçoivent nécessairement le signal émis, si des obstacles sont présents dans la zone à portée du capteur. Dans des conditions d'éclairement suffisantes, les images perçues sont très riches en terme d'information : les images sont en effet des représentations structurées, offrant une haute résolution sur un secteur angulaire important en un temps d'acquisition réduit. Enfin, comme ces capteurs sont légers et consomment peu, ils sont plus adaptés à être embarqués sur des plateformes robotiques, en particulier les drones qui ont une capacité de charge très limitée.

Le SLAM visuel peut être réalisé avec une caméra seule, on parle de SLAM monoculaire, ou avec deux caméras, appelé SLAM stéréo. Ce dernier est présenté dans la partie suivante. On suppose que les caméras sont fixes sur le robot et que l'environnement est statique. Le principe est de suivre des amers extraits dans les images durant le déplacement. Le mouvement de la caméra dans l'espace est calculé à partir du mouvement apparent des amers dans les images. Ce mouvement est calculé par rapport à la position et à l'orientation initiale du robot, selon un processus d'intégration ou de filtrage. Ce processus conduit à une dérive des paramètres estimés, dérive qui peut être corrigée lorsque le robot repasse deux fois au même endroit, événement appelé fermeture de boucle.

MonoSLAM [Davison, 2003] est un algorithme de SLAM monoculaire dans lequel la position des amers dans l'environnement et la localisation du robot sont probabilisés. L'estimation de la position de la caméra et des amers est réalisée par un filtre de Kalman étendu. PTAM (Parallel Tracking And Mapping) [Klein and Murray, 2007] est un algorithme de SLAM monoculaire dans lequel un processus d'images-clé est utilisé pour la tâche de cartographie : au lieu de générer des amers à chaque image reçue, le processus est réalisé seulement à certains instants, choisis de façon astucieuse. Cette stratégie permet la réalisation d'un processus d'ajustement de faisceaux, opération qui optimise les positions de la caméra et des amers sur toutes les images-clé. LSD-SLAM [Engel et al., 2014] est un algorithme de SLAM monoculaire direct. Au lieu d'extraire des amers dans les images puis de calculer la localisation de la caméra et des amers, le calcul de la position de la caméra est réalisé directement par l'alignement des images selon leur intensité. Cette technique permet d'exploiter plus d'information dans les images qu'une technique basée primitives où l'information est limitée par le type de primitives utilisé (les coins par exemple). Dans le LSD-SLAM, comme dans de nombreuses références récentes, la problématique d'estimation peut être représentée comme un graphe de poses, dans laquelle les sommets du graphe sont les poses des images-clés, sur lesquelles on réalise l'optimisation. Les arêtes de ce graphe codent les contraintes (ou facteurs) qui portent sur les poses, qui peuvent provenir de mesures odométriques ou de mesures image. Ce procédé, qui s'approche en fait de l'odométrie visuelle, permet de poursuivre l'estimation dans des environnements très étendus en gardant un fonctionnement temps-réel.

De très nombreuses solutions algorithmiques ont été proposées pour le SLAM depuis les années 1990. Pour juger de l'intérêt de ces approches, il est utile de disposer de données de référence permettant de faire des évaluations comparatives. Le dataset KITTI [Geiger et al., 2013] est un jeu de données de navigation urbaine collecté par une voiture équipée de nombreux capteurs et mis en ligne par l'institut technologique de Karslruhe et Toyota. Il permet d'évaluer la performance des algorithmes de SLAM, qu'ils soient à base de LIDAR ou de vision passive. Les meilleurs algorithmes présentent des dérives en translation de l'ordre de 0.64 % et en rotation de 0.0014 deg.m^{-1} pour des temps d'exécution d'environ 0.1 s. Parmi les algorithmes classés dans les 20 premiers, 5 algorithmes sont basés sur du SLAM LIDAR et 15 sur du SLAM visuel ou de l'odométrie visuelle en juillet 2017 [http://www.cvlibs.net/datasets/kitti/eval_odometry.php].

Un système monoculaire obtient une information de profondeur en utilisant deux images reçues à deux instants différents selon deux points de vus par une opération de triangulation. L'information de profondeur restituée dépend alors de la connaissance de la distance entre les deux points de vues. La plus grande difficulté du SLAM monoculaire est d'estimer le facteur d'échelle, qui ne peut pas être directement observé. Pour cela, on peut utiliser un capteur complémentaire, comme une centrale inertielle pour estimer dynamiquement ce facteur ou initialiser l'algorithme de SLAM par une distance connue entre les deux premières positions.

Le système stéréo est composé de deux caméras fixes l'une par rapport à l'autre, qui permettent, à tout instant, de trianguler la position de tout point situé dans le champ de vue commun. La distance entre les deux caméras (ligne de base) est connue précisément grâce à la calibration du système stéréo (voir II.2.1.1), ce qui résout le problème du facteur d'échelle. Avec un système stéréo, la portée est limitée par la ligne de base. Avec les caméras utilisées dans nos expériences, elle est de l'ordre d'une dizaine de mètres pour une ligne de base mesurant 20 à 30 cm. Aujourd'hui, la plupart des systèmes opérationnels basés sur la localisation visuelle utilisent des systèmes stéréo. Ils ont en particulier été utilisés avec succès pour effectuer des missions de navigation sur des drones [Orsag et al., 2015],[Krombach et al., 2016], [Engel et al., 2015], [Marzat et al., 2017].

[Nistér et al., 2006] ont développé un algorithme compatible avec des systèmes monoculaire ou stéréo. Dans les deux cas, le système travaille sur des paires d'images, provenant des deux caméras dans le cas stéréo et provenant de la même caméra à deux instants différents dans le cas monoculaire. L'estimation de pose est réalisée par une procédure RANSAC [Fischler and Bolles, 1981], qui permet de rejeter les valeurs aberrantes. Cet algorithme est décrit en section III.1.3.3. Le rejet de données aberrantes est indispensable pour augmenter le nombre de points suivis utilisés dans l'algorithme et ainsi avoir une estimation de pose plus précise. L'algorithme LSD-SLAM monoculaire a été adapté à la vision stéréo [Engel et al., 2015]. L'estimation de la profondeur par le système stéréo permet d'éliminer le procédé d'estimation de l'échelle nécessaire avec le SLAM monoculaire. L'algorithme a été testé sur le dataset KITTI, la dérive moyenne en translation est de 1.21 % et la dérive en rotation est de 0.35 deg.m^{-1}. L'algorithme est plus rapide que les autres méthodes présentant une précision similaire.

III.1.2 Odométrie visuelle

Le SLAM nécessite la création d'une carte globale mise à jour au fur et à mesure de la réception de nouvelles informations. La taille de la carte augmente donc au cours du déplacement. Il est alors nécessaire d'utiliser des stratégies pour réduire la quantité d'information mémorisée et de limiter le temps de calcul. On peut par exemple utiliser le procédé d'images-clé, comme dans [Klein and Murray, 2007]. Si la localisation est l'unique objectif, construire la carte globale de l'environnement n'est pas indispensable. On peut alors utiliser l'odométrie visuelle qui peut être vue comme la réalisation d'un SLAM sans construction de carte globale. Pendant son déplacement, le robot construit une carte locale qui lui permet de se localiser. Quand les amers connus deviennent inutiles pour la localisation, notamment parce qu'ils ne sont plus visibles dans l'image courante, ils sont effacés de la carte. Cette technique permet d'éviter les problèmes de temps de calcul et de taille mémoire dans un SLAM avec carte globale. L'odométrie visuelle présente une dérive qui augmente au cours du déplacement comme tous les systèmes basés sur l'intégration dans le temps d'une mesure bruitée. Le procédé de fermeture de boucle, utilisé dans le SLAM, qui consiste à reconnaître un lieu déjà visité et à réestimer la trajectoire en conséquence n'est plus applicable avec l'odométrie visuelle. La localisation calculée peut donc être moins précise. Il existe cependant des systèmes d'odométrie visuelle très performants, comme le montre la synthèse effectuée dans [Scaramuzza and Fraundorfer, 2011] et [Fraundorfer and Scaramuzza, 2012].

Ici nous nous focalisons sur le cas particulier de l'odométrie visuelle stéréo, dont une application particulièrement impressionnante est la navigation des rovers martiens [Maimone et al., 2007]. [Mallet et al., 2000] ont développé un algorithme d'odométrie stéréo qui associe des nuages de points 3D obtenus à différents instants, en suivant les pixels correspondants dans les images. Dans [Howard, 2008], les auteurs présentent un algorithme d'odométrie stéréo qui opère sur des images de disparité denses. [Cvišić and Petrović, 2015] effectuent une sélection des amers dans leur algorithme afin d'obtenir des amers stables et ainsi réduire la dérive sur la localisation. L'estimation de la rotation est séparée de celle de la translation. Pour la rotation, un procédé monoculaire est utilisé tandis que la translation est estimée avec une technique d'odométrie stéréo. Les données de localisation visuelle sont fusionnées avec celles d'une centrale inertielle grâce à un filtre de Kalman étendu. L'algorithme LSD-SLAM stéréo [Engel et al., 2015] a été testé sur le dataset KITTI avec un paramétrage qui ne garde que quelques frames en mémoire afin de le transformer en un algorithme d'odométrie visuelle stéréo. La dérive en translation est plus importante 1.40 % contre 1.21 % pour la version SLAM. Par contre, le temps de calcul est beaucoup plus court, 28 ms contre 69 ms.

III.1.3 Fonctionnement de l'odométrie visuelle stéréo

Dans cette thèse, on s'intéresse tout particulièrement à l'odométrie visuelle stéréo, qui est utilisée sur les robots de l'ONERA pour calculer leur localisation. Les sections suivantes décrivent les étapes principales d'un algorithme d'odométrie visuelle stéréo basé amers.

III.1.3.1 Détection des points d'intérêt et appariement

La première étape consiste à détecter des points d'intérêt dans l'image gauche et à chercher les points correspondants dans l'image droite. Les points associés gauche et droit, qu'on appelle aussi points homologues, sont la projection d'un même point 3D de la scène dans chaque image.

Différents types de détecteurs existent : Harris [Harris and Stephens, 1988], Shi-Tomasi [Shi and Tomasi, 1994], SIFT (Scale-Invariant Feature Transform) [Lowe, 2004], FAST (Features from Accelerated Segment Test) [Rosten and Drummond, 2006], ORB (Oriented FAST and Rotated BRIEF) [Rublee et al., 2011]. Le principe de tous ces détecteurs est de chercher des points saillants dans l'image, qui se démarquent du voisinage. Ces points doivent pouvoir être reconnus dans les images suivantes après déplacement de la caméra. Les méthodes Harris et Shi-Tomasi recherchent ces points à partir d'un calcul local, autour de chaque pixel, de la matrice de covariance empirique des gradients spatiaux de l'image. Ces méthodes sont rapides mais ne sont pas robustes face aux changements de point de vue. La méthode SIFT associe une méthode de détection multi-échelles et un descripteur qui caractérise le contenu visuel de la région pour qu'il soit invariant aux rotations et aux changements d'échelle. Mais cette méthode est plus coûteuse en temps de calcul et donc moins adaptée pour une utilisation temps-réel.

La recherche du point correspondant dans l'image droite est effectuée le long de la ligne épipolaire, en utilisant des techniques de recherche locale telle que la corrélation.

III.1.3.2 Triangulation

La seconde étape consiste à calculer la position des points 3D dans l'espace par triangulation à partir des points projetés dans les deux images. En reprenant les notations de la section II.2.1.1, notamment (u_l, v_l) pour la position du point dans l'image gauche, b la ligne de base et d la disparité, l'estimation de la position du point 3D triangulé est données par :

$$\hat{Y} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \Pi^{-1}(u_l, v_l, d) = \frac{-b}{d} \cdot \begin{pmatrix} u_l - u_0 \\ v_l - v_0 \\ \alpha \end{pmatrix} \tag{III.1}$$

III.1.3.3 Calcul de pose

La dernière étape est le calcul du déplacement de la caméra entre l'instant précédent t_{n-1} et l'instant actuel t_n. Une association temporelle entre les points extraits à ces deux instants est effectuée. Pour cela, on peut par exemple suivre les points extraits dans l'image gauche à t_{n-1} pendant le déplacement en utilisant le tracker KLT [Shi and Tomasi, 1994].

Deux approches principales existent pour estimer la pose :

— estimation 3D-3D : on utilise le suivi temporel pour former des associations entre les ensembles de points 3D triangulés aux instants t_{n-1} et t_n. Le déplacement est obtenu par la minimisation d'un critère des moindres carrés :

$$\arg\min_{P_n} \sum_i \|\tilde{Y}_n^i - P_n \tilde{Y}_{n-1}^i\|^2 \tag{III.2}$$

i désigne le i ème point 3D et P_n est la matrice de transformation entre les instants t_{n-1} et t_n. L'algorithme P3P (Perspective-Three-Point) [Umeyama, 1991] permet d'estimer la position et l'orientation d'un objet par rapport à la caméra à partir de quatre associations 3D-3D. Il est souvent utilisé pour initialiser la minimisation de (III.2) dans le cadre d'une procédure RANSAC (voir plus bas).

— estimation 3D-2D : on forme cette fois-ci des associations entre les points 2D de l'image gauche actuelle et la projection sur l'image gauche des points 3D obtenus à l'instant t_{n-1}. L'estimation du déplacement est calculée par la minimisation d'un critère des moindres carrés sur l'erreur de reprojection :

$$\arg\min_{P_n} \sum_i \|p_n^i - \tilde{p}_{n-1}^i\|^2 = \arg\min_{P_n} \sum_i \|p_n^i - \Pi\left(P_n \tilde{Y}_{n-1}^i\right)\|^2 \tag{III.3}$$

Il arrive que certaines associations entre deux points soient erronées. Pour les détecter et ne pas les considérer dans le calcul de la pose, on utilise généralement l'algorithme RANSAC.

RANSAC

L'algorithme RANSAC (RANdom SAmple Consensus) [Fischler and Bolles, 1981] est une méthode itérative utilisée pour l'estimation des paramètres d'un modèle avec détection des valeurs aberrantes (outliers). Une première estimation des paramètres est réalisée à partir d'un sous-ensemble sélectionné aléatoirement dans l'ensemble des données. La concordance avec le reste des données est calculée. Si le nombre de données qui concordent avec le modèle (inliers) est suffisant, les paramètres du modèle sont affinés en les réestimant avec tous les inliers. Sinon un nouveau choix aléatoire de points est effectué et la procédure est réitérée. Cette boucle est réitérée un certain nombre de fois, et le meilleur modèle est gardé.

Cette méthode est utilisée plusieurs fois dans les travaux présentés dans ce mémoire.

III.1.4 eVO

Dans les expérimentations réalisées dans le cadre de cette thèse, nous avons choisi d'utiliser eVO comme algorithme d'odométrie visuelle [Sanfourche et al., 2013], développé à l'ONERA. eVO est un algorithme d'odométrie visuelle stéréo, qui travaille avec une carte globale contenant les points 3D et mise à jour selon un processus d'images-clé.

Deux tâches travaillent en parallèle :

— Cartographie : cette tâche consiste à localiser par stéréovision de nouveaux points 3D dans l'espace et à les ajouter à la carte. Les points qui sont sortis du champ

de vue sont supprimés de la carte. Cette tâche est exécutée uniquement à l'initialisation de l'algorithme ou quand le rapport entre le nombre de points visibles et le nombre de points 3D dans la carte descend sous un certain seuil. Les images correspondant à ces instants sont appelées images-clé.

— Localisation : Cette tâche consiste à calculer la position et l'orientation de la caméra gauche dans l'espace. Les correspondances entre les points 3D stockés dans la carte et les amers extraits de l'image gauche sont suivis par le tracker KLT [Shi and Tomasi, 1994]. La pose est calculée en deux étapes : une première estimation est donnée par l'algorithme P3P utilisant une procédure RANSAC, expliquée dans la section précédente III.1.3.3. Ensuite, cette pose est raffinée en minimisant l'erreur de reprojection des points 3D déterminés comme inliers par la procédure RANSAC.

La figure III.2 illustre le fonctionnement de l'algorithme.

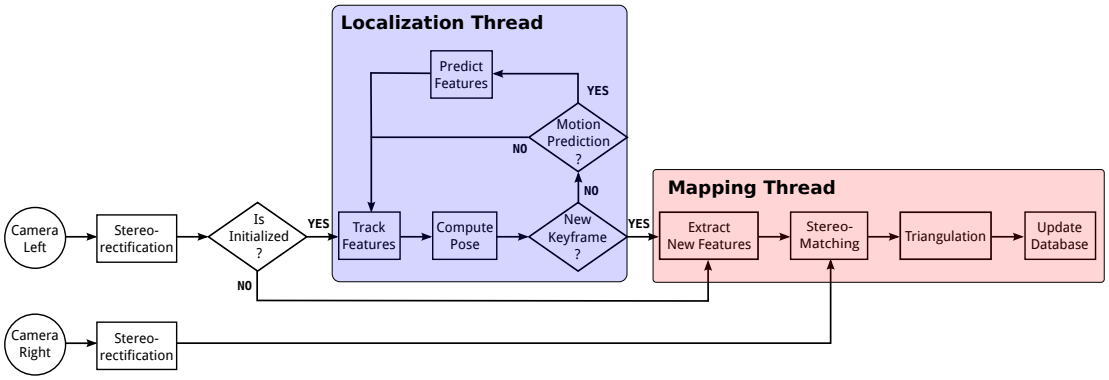


FIGURE III.2 – Schéma représentant la chaîne de fonctionnement de l'algorithme eVO.

Lors de sa publication [Sanfourche et al., 2013], les évaluations d'eVO sur le dataset KITTI donnaient une dérive moyenne en translation de 1.76 % et en rotation de 0.0036 deg.m^{-1} pour un temps de calcul de 0.05 s. Même si les performances des méthodes ont augmenté depuis, cela fait d'eVO un algorithme à l'état de l'art des méthodes d'odométrie stéréo basées amers actuelles.

III.1.5 Limites de la localisation visuelle

Cependant, quelle que soit la méthode utilisée, des limites existent sur l'utilisation de la localisation visuelle. On trouve d'abord une contrainte sur la qualité des images perçues. Il faut que celles-ci soient suffisamment illuminées et texturées pour pouvoir en extraire des informations utilisables pour le calcul de la localisation. Il faut également que les différentes images capturées se recouvrent en partie pour pouvoir observer des éléments en commun, ce qui nécessite d'avoir des fréquences d'acquisition suffisantes par rapport à la vitesse de déplacement de la caméra dans l'espace. Enfin, il faut que les scènes observées soient majoritairement statiques car le processus de calcul de la pose suppose que les points 3D sont fixes dans l'espace.

Quand la qualité des images reçues est mauvaise, l'extraction de points dans l'image est difficile et le calcul de pose est très imprécis ou peut même être impossible quand trop peu de points sont extraits. Plusieurs solutions existent pour obtenir une localisation robuste par rapport à une qualité d'image dégradée. On peut effectuer une fusion de données avec des capteurs complémentaires comme dans [Cvišić and Petrović, 2015]. Deux méthodes de fusion de données existent : la fusion lâche et la fusion serrée. La fusion lâche consiste à fusionner les données de localisation calculées à partir des mesures de chaque capteur embarqué [Weiss et al., 2012, Scaramuzza et al., 2014]. La fusion serrée exploite directement les mesures bas niveau de chaque capteur pour obtenir une estimation jointe de la pose [Mourikis and Roumeliotis, 2007, Leutenegger et al., 2015]. Au lieu d'utiliser la pose calculée par l'algorithme d'odométrie visuelle, ce type de fusion utilise directement les amers extraits dans les images et les mesures des autres capteurs. Cette deuxième solution est plus difficile à mettre en œuvre mais apporte une meilleure précision dans l'estimation. L'utilisation de la fusion de données pour augmenter la robustesse de la localisation est étudiée dans le chapitre IV.

Une deuxième solution consiste à modifier localement la trajectoire afin de s'assurer que les conditions sur la qualité des images soient respectées. La littérature portant sur cette solution est exposée en section III.3.2. Nous développons de nouvelles approches de ce type dans les chapitres V et VI.

III.2 Navigation robotique

A partir des informations de localisation et de cartographie, le robot va pouvoir se déplacer dans son environnement de façon sécurisée et accomplir les objectifs qu'il doit réaliser. Pour cela, différentes méthodes de guidage existent. On peut utiliser des méthodes de planification de trajectoire III.2.1 ou des méthodes de commande telles que la commande prédictive III.2.2.3.

III.2.1 Planification de trajectoire

L'objectif de la planification de trajectoire est de trouver un chemin réalisable à partir de la position initiale jusqu'à la position finale qui évite les obstacles. Un grand nombre de méthodes existent, quelques-unes de ces méthodes sont décrites ci-dessous : méthodes basées graphe, champs de potentiel. Une revue sur les méthodes de planification pour la robotique mobile est disponible dans [Goerzen et al., 2010].

III.2.1.1 Méthodes basées graphe

Le principe des méthodes basées graphe est de chercher un chemin entre deux nœuds dans un graphe utilisé pour donner une représentation topologique de l'environnement : les nœuds sont des états libres de l'environnement. Les arêtes représentent les chemins sans obstacles menant d'un état à un autre. La figure III.3 montre un exemple de carte topologique.

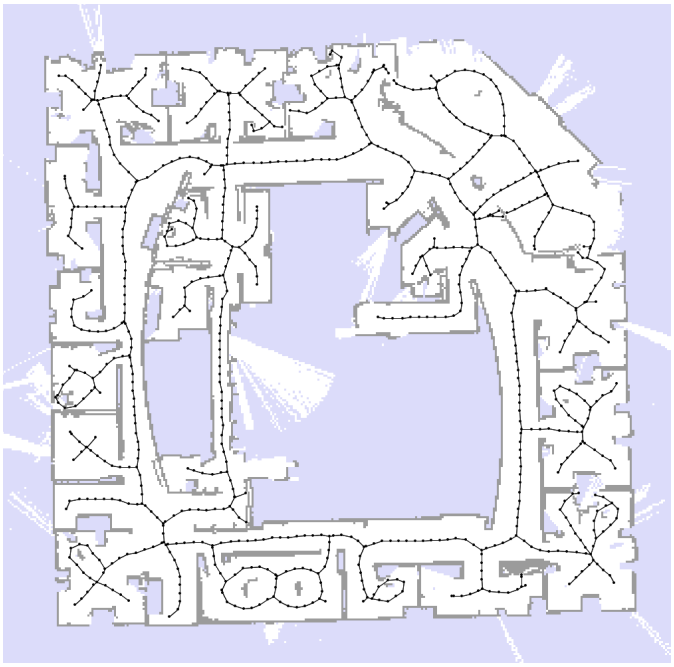


FIGURE III.3 – Carte de l'environnement et représentation sous forme de graphe (en noir), illustration obtenue sur [https://www.cs.washington.edu/node/4231/].

Dans cette partie, on note X un état du robot. L'ensemble des états est noté $\mathbb{X}$. L'ensemble des états libres est appelé $\mathbb{X}_{libre}$ et l'ensemble des états occupés $\mathbb{X}_{occ}$. On a $\mathbb{X} = \mathbb{X}_{libre} \cup \mathbb{X}_{occ}$. On appelle X_{init} l'état initial et X_{obj} l'état objectif du robot. Le but de la planification est de trouver un chemin entre X_{init} et X_{obj} inclus dans $\mathbb{X}_{libre}$. Un objectif supplémentaire peut être de trouver le chemin le plus court, appelé chemin optimal.

Plusieurs algorithmes sont régulièrement utilisés en robotique mobile pour effectuer une planification de trajectoire à partir d'un graphe :

— Les algorithmes A* et Dijkstra [Dijkstra, 1959, Hart et al., 1968]

Ces algorithmes supposent l'existence d'un graphe déjà construit.

L'algorithme de Dijkstra recherche le plus court chemin dans un graphe pondéré entre l'état initial X_{init} et l'objectif X_{obj} de façon itérative. Les pondérations représentent les distances entre les états représentés par les nœuds du graphe. La figure III.4 est un exemple d'application de l'algorithme.

Tout d'abord, un sous-graphe est initialisé avec le nœud X_{init}. Le sous-graphe est composé par les nœuds colorés en rouge sur la première illustration en haut à gauche de la figure III.4. La première étape consiste à ajouter au sous-graphe le nœud adjacent qui présente la distance la plus petite à X_{init}. Ce nœud est représenté en rouge sur la seconde illustration. La deuxième étape est la mise à jour des distances entre les nœuds adjacents au sous-graphe et X_{init}. Si plusieurs chemins

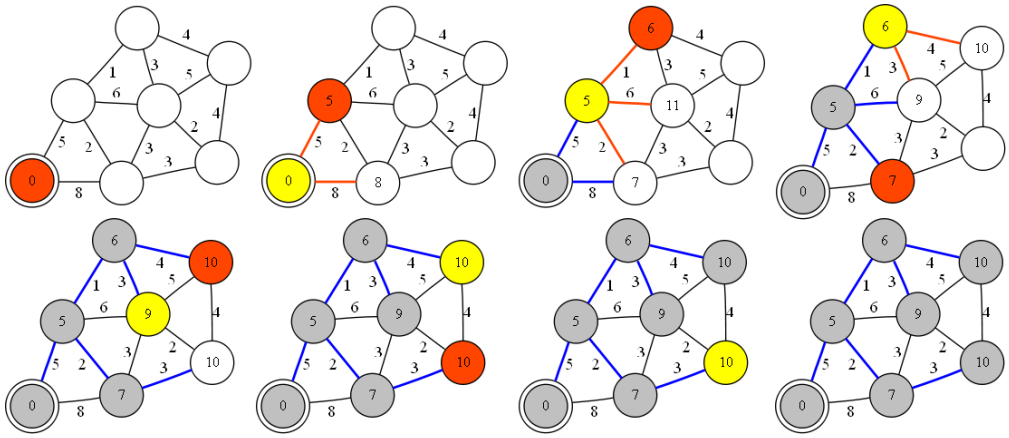


FIGURE III.4 – Illustration de l'algorithme de Dijkstra. Illustration issue de [http://www.math.cornell.edu/~mec/Winter2009/Thompson/search.html].

sont possibles entre les deux nœuds dans le sous-graphe, le chemin qui présente la plus petite distance est sélectionné. Les chemins minimaux sont surlignés en bleu sur les illustrations suivantes. Ces deux étapes sont réitérées jusqu'à atteindre le nœud X_{obj}.

A l'instant n, le nœud dessiné en jaune représente le nœud qui a été ajouté au sous-graphe à l'étape $n-1$. On peut remarquer que le nœud ajouté n'est donc pas forcément adjacent au nœud précédent, le sous-graphe progresse dans toutes les directions. L'algorithme A* est une variante de l'algorithme de Dijkstra qui permet de trouver une solution en un temps plus court. Une heuristique sur la direction à suivre est formulée afin de limiter le nombre de nœuds ajoutés au sous-graphe. Cependant, la solution obtenue n'est pas obligatoirement la solution optimale.

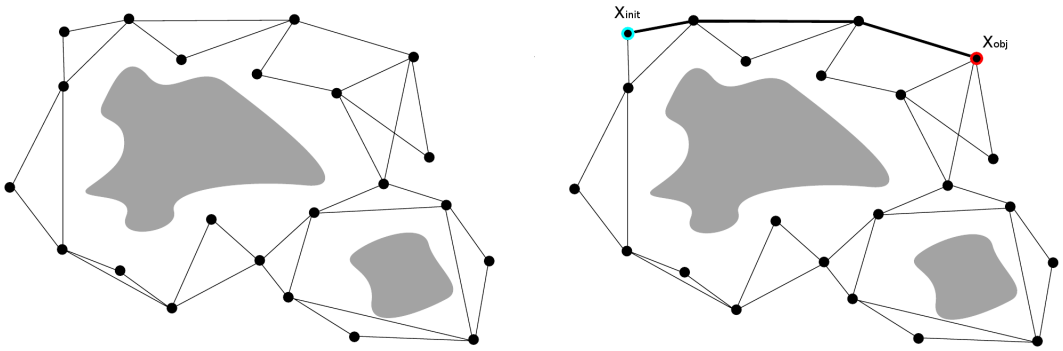
Pour ces deux algorithmes, le temps d'optimisation augmente avec le nombre de sommets ou d'arêtes du graphe.

— Probabilistic Roadmap Methods (PRM) [Kavraki et al., 1996]

Cet algorithme de planification de chemin dans un graphe procède en deux étapes. La première étape consiste en la construction du graphe. Des états sans obstacles sont définis par tirages aléatoires dans l'environnement, ils forment l'ensemble des nœuds du graphe. Les nœuds correspondants aux états X_{init} et X_{obj} sont également ajoutés au graphe. Les arêtes sont construites en recherchant des chemins sans obstacles entre les états proches. Tous les nœuds doivent être connectés au graphe.

Après avoir construit le graphe, la deuxième étape consiste à rechercher le plus court chemin dans le graphe. De nombreux algorithmes sont utilisables, on peut utiliser par exemple les algorithmes A* ou Dijkstra présentés dans la section précédente. Le chemin calculé par l'algorithme n'est pas obligatoirement le chemin optimal.

La figure III.5 montre les deux étapes de l'algorithme.



(a) Construction du graphe (b) Recherche d'un chemin entre X_{init} et X_{obj}

FIGURE III.5 – Illustration de l'algorithme PRM. Les images sont issues de [Pharpatara, 2015]

Cet algorithme a par exemple été utilisée par [He et al., 2008] et [Bachrach et al., 2012] pour la navigation de drones avec évitement d'obstacles.

Cependant, si l'environnement présente des passages étroits, il est possible de ne pas trouver de chemin connectant les nœuds de part et d'autre du passage, il peut alors ne pas exister de chemins reliant X_{init} et X_{obj}. De plus, le temps d'exécution dépend du nombre de nœuds dans le graphe. L'algorithme est plus rapide si le nombre de nœuds est plus faible, mais il risque alors de ne pas trouver de chemin libre joignant les deux états X_{init} et X_{obj}.

— Rapidly exploring Random Tree (RRT) [LaValle, 1998]

Cet algorithme cherche le chemin entre les nœuds X_{init} et X_{obj} en une seule étape. La construction du graphe et la recherche d'un chemin sont réalisées en simultané.

Le graphe est construit de manière incrémentale, en explorant progressivement l'espace. Un nouveau nœud X_{nv} est choisi aléatoirement dans l'espace proche et ajouté au graphe en créant simultanément l'arête avec son plus proche voisin, si le chemin entre les deux nœuds est sans obstacle. Sur la figure III.6, quatre images montrent la progression de la construction du graphe dans l'espace. L'expansion du graphe étant uniforme dans toutes les directions, le temps de convergence peut être long.

Il existe de nombreuses variantes de cet algorithme, on peut citer par exemple l'algorithme RRT* [Karaman and Frazzoli, 2011] qui assure la convergence vers le chemin optimal grâce à une procédure de modification locale du graphe afin que le chemin entre chaque nœud et l'état initial X_{init} soit le chemin le plus court. L'algorithme RRT* permet d'obtenir la solution optimale mais est plus lent que l'algorithme de base à cause de la procédure de replanification locale. [Sadat et al., 2014] utilise cet algorithme dans ses travaux pour faire naviguer un drone.

Les avantages des algorithmes de planification utilisant des graphes sont qu'ils sont

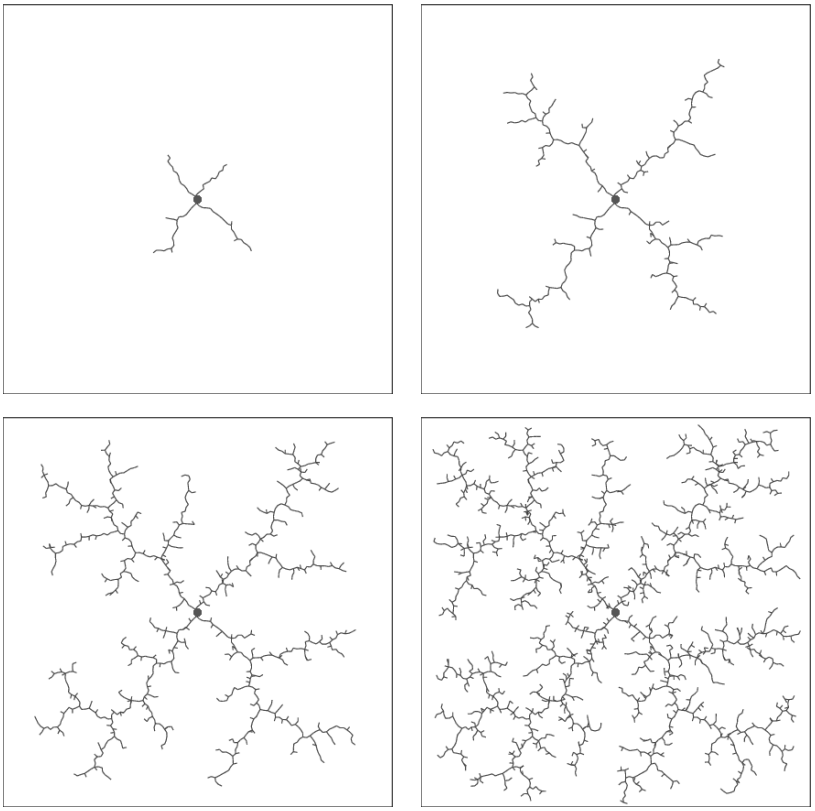


FIGURE III.6 – Progression de l'algorithme RRT. Les images sont issus du site [http://msl.cs.uiuc.edu/rrt/].

faciles à mettre en œuvre. Par contre, le temps de convergence est inconnu et peut devenir très long si la taille de l'espace d'état est importante. Ces algorithmes fournissent une trajectoire globale à l'instant du calcul. Il faut donc connaître l'environnement a priori. Dans le cas d'un environnement inconnu ou dynamique, il est nécessaire de faire de nouvelles planifications à mesure que l'environnement est exploré, ce qui est coûteux.

De plus, ces algorithmes ne définissent pas la manière dont la trajectoire doit être réalisée. Il faut mettre en place une loi de commande afin de réaliser le suivi de trajectoire. Des contraintes lors de la construction du graphe peuvent être ajoutées afin de prendre en compte les contraintes matérielles du robot (contraintes non-holonomes par exemple) afin de s'assurer que les trajectoires soient réalisables par le robot, mais le coût de calcul est alors plus important.

III.2.1.2 Champs de potentiel artificiels

La méthode de champs de potentiel artificiels a été introduite par [Khatib, 1986]. Elle consiste à considérer le robot comme une particule sous l'influence d'un champ de potentiel afin de le guider vers son objectif X_{obj} tout en évitant les obstacles. Le robot

se déplace dans un champ de forces composé d'une force attractive qui l'attire vers son objectif X_{obj} et d'une force répulsive, qui l'éloigne des obstacles. Un champ de potentiel $\varphi(X)$ est défini comme :

$$\varphi(X) = \varphi_{att}(X) + \sum_{i=1}^{m} \varphi^i_{rep}(X) \tag{III.4}$$

avec $\varphi_{att}(X)$, le champ attracteur, défini généralement comme une fonction de la distance au point X_{obj} :

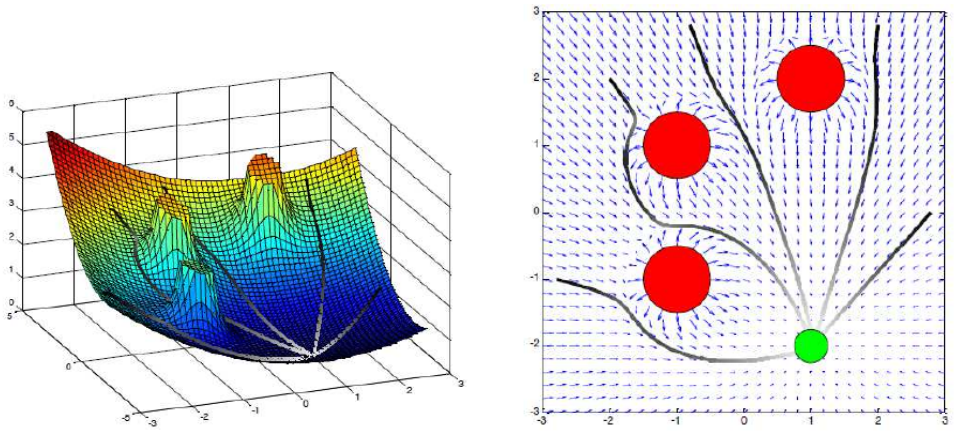
$$\varphi_{att}(X) = \|X - X_{obj}\|^2 \tag{III.5}$$

$\varphi^i_{rep}(X)$ est le champ répulseur défini pour chaque obstacle i avec m le nombre total d'obstacles. Il existe de nombreuses formulations pour définir le champ répulseur.

La force est calculée comme le gradient négatif de $\varphi(X)$:

$$F(X) = -\nabla\varphi(X) \tag{III.6}$$

On obtient un champ vectoriel qui donne pour chaque position X de l'espace la direction de la plus forte pente dans le champ de potentiel. La figure III.7 montre un exemple de champ de potentiel et du champ vectoriel correspondant.



(a) Champ de potentiel artificiel (b) Champ vectoriel

FIGURE III.7 – Illustration d'un champ de potentiel artificiel et du champ vectoriel correspondant. Les obstacles sont les cercles rouges et l'objectif est en vert. Quatre trajectoires obtenues au départ de quatre point différents sont dessinées. Les images sont issues de [Pharpatara, 2015]

A chaque itération, la trajectoire du robot est définie pour suivre la direction de F au point X. Dans [Vidal-Calleja et al., 2006], la fonction F est modifiée afin de devenir une fonction de l'entrée en commande U, ce qui permet d'obtenir directement la commande à envoyer au robot.

Cette méthode présente l'avantage d'être rapide et économe en calculs. Elle peut être facilement appliquée aux environnements dynamiques. Elle est efficace pour l'évitement d'obstacles mais elle est difficilement adaptable à des missions plus complexes dans lesquelles les objectifs de mission ne sont pas traduisibles en champ de potentiel.

Le problème majeur de la méthode est la présence de minima locaux dans le champ de potentiel dans lesquels le robot peut se retrouver piégé. Pour cela, de nombreuses améliorations ont été proposées : redéfinition du champ de potentiel, définition d'objectifs virtuels intermédiaires, méthode de suivi de mur, voir [Li et al., 2013].

La méthode CHOMP [Ratliff et al., 2009] s'inspire de la méthode des champs de potentiel artificiels pour raffiner des trajectoires obtenues à partir d'algorithmes de planification. Généralement, ces trajectoires sont non lisses et ne répondent pas aux contraintes dynamiques du robot. L'algorithme permet d'optimiser la trajectoire pour obtenir une trajectoire lissée et évitant les obstacles, en utilisant un champ de potentiel caractérisé par les obstacles et les contraintes sur le robot. Cette méthode peut également optimiser la trajectoire à partir d'une simple trajectoire initiale, même dans le cas où elle rencontre des obstacles.

III.2.2 Commande réactive

Les méthodes de planification de trajectoire décrites dans la partie précédente présentent des limites de fonctionnement qui les rendent difficilement applicables pour les missions considérées dans cette thèse. Les méthodes de planification basées sur les graphes ne sont pas compatibles avec des environnements inconnus et un fonctionnement temps-réel. Les méthodes sur les champs de potentiels artificiel ne sont pas applicables pour la réalisation de missions complexes. De plus, ces algorithmes donnent uniquement la trajectoire qu'il faut effectuer pour atteindre l'objectif, il est nécessaire de mettre en place une loi de guidage pour que le robot suive la trajectoire calculée.

La commande réactive cherche à optimiser la trajectoire locale du robot à partir des informations connues à l'instant actuel. Elle permet une adaptation aux changements dans l'environnement (découverte d'un nouvel obstacle, objets dynamiques...). Les contraintes dynamiques sur le robot sont considérées directement dans le calcul de la commande. Le temps de calcul est indépendant de la taille de l'environnement.

Cependant, la prédiction de la trajectoire est réalisée uniquement à court-terme et il est difficile d'anticiper par rapport à un objectif lointain.

III.2.2.1 Guidage simple

Une première approche simple pour la commande d'un robot mobile est de considérer une commande qui annule l'erreur entre la position courante et la position désirée. Pour cela, on peut faire appel à des correcteurs linéaires classiques comme le PID ou la commande par retour d'état.

Le régulateur PID (Proportionnel, Intégral, Dérivé) permet d'asservir un système en agissant sur l'erreur e entre la consigne et la mesure. La commande est exprimée sous la

forme :

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{III.7}$$

avec K_p le gain proportionnel, K_i le gain sur la partie intégrale et K_d le gain sur la partie dérivée. Ces trois gains sont réglés de telle sorte que la réponse du système suive des caractéristiques désirés en termes de robustesse, de précision et de rapidité. La figure III.8 montre un schéma d'une boucle de commande régulée par un correcteur PID.

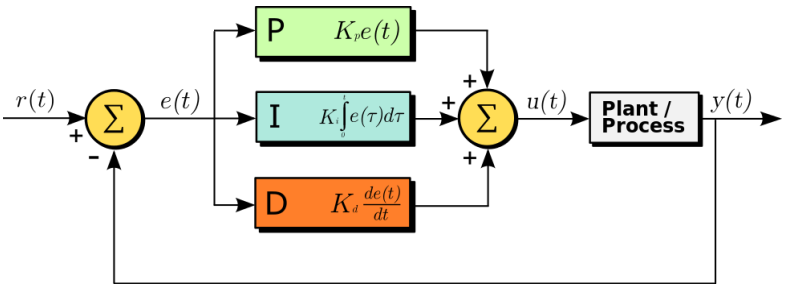


FIGURE III.8 – Schéma-bloc d'une boucle de commande par PID.
Par Arturo Urquizo - [http://commons.wikimedia.org/wiki/File:PID.svg], CC BY-SA 3.0, [https://commons.wikimedia.org/w/index.php?curid=17633925]

La commande par retour d'état (ou commande par placement de pôles) définit la commande U à appliquer à un système exprimé sous la forme d'une représentation d'état :

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX \end{cases} \tag{III.8}$$

La commande se présente sous la forme :

$$U = r - KX \tag{III.9}$$

avec r, le signal de référence et K la matrice de gain du retour d'état, dont les coefficients sont fixés afin d'obtenir les performances désirées. La figure III.9 montre un schéma d'une boucle de commande avec retour d'état. Sur cette figure, X est supposé être la sortie du système (C est alors la matrice identité). Dans le cas général, il faut mettre en place un observateur afin d'obtenir l'état X à partir de la sortie Y.

Une exemple d'application de ce type de commande pour un robot mobile consiste à calculer la différence δ entre l'orientation du robot θ et la direction entre la position actuelle du robot et le point à rallier ϕ, comme illustré sur la figure III.10.

La commande en vitesse angulaire peut alors être définie par $\omega = -k.\delta$ avec k gain réel positif fixe. La commande en vitesse linéaire v est gérée dans une boucle de régulation indépendante.

Cette commande a pour effet d'aligner l'orientation du robot sur la direction du point à rallier. Plus la différence δ est importante, plus la commande en vitesse angulaire est

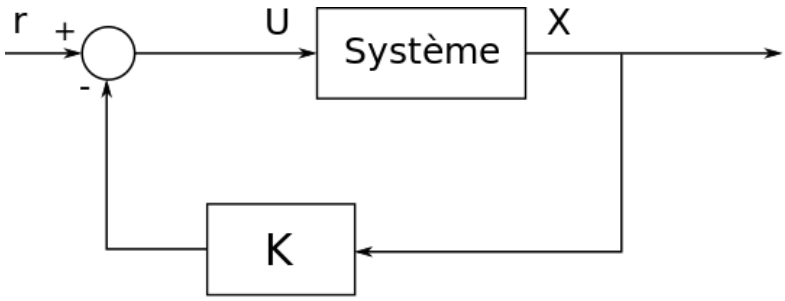


FIGURE III.9 – Schéma-bloc d'une boucle de commande par retour d'état

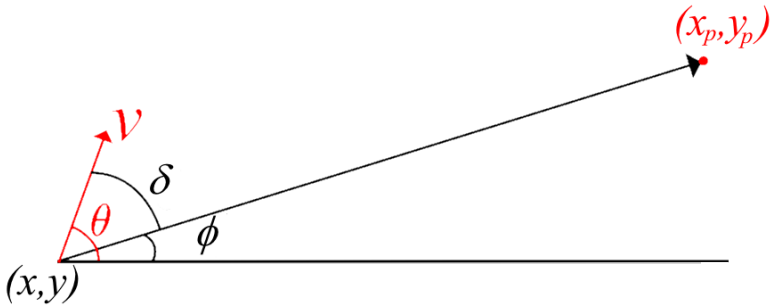


FIGURE III.10 – Schéma représentant le principe du guidage

importante. Quand le robot est aligné avec le point, la commande en vitesse angulaire devient nulle et le robot avance alors jusqu'à atteindre le point.

Cette commande permet de réaliser des missions de ralliement de points. Elle peut être utilisée par exemple pour le suivi d'une trajectoire définie par une liste de points de passage, à condition de ne pas avoir d'obstacles à proximité de la trajectoire. Par contre, ce type de commande ne peut prendre en compte que des objectifs définis sous la forme de points à rallier et ne s'adapte pas aux environnements complexes.

Les correcteurs linéaires sont capables de contrôler des systèmes robotiques pour effectuer des missions simples. Cependant, si le modèle présente de fortes non-linéarités ou que les missions demandées sont complexes, ces méthodes ne peuvent plus être utilisées. Il faut alors envisager d'autres types de commandes comme la commande optimale ou la commande prédictive. Ces méthodes sont présentées dans les sections suivantes.

III.2.2.2 Commande optimale

La commande optimale permet de déterminer la commande d'un système qui minimise un critère de performance.

L'état du système est noté X et la commande est notée U. Soit un système à temps continu défini par :

$$\dot{X} = f(X, U, t) \tag{III.10}$$

avec la condition initiale $X(t_0) = X_0$. t_0 est l'instant initial.

On définit usuellement le critère de performance J par :

$$J(X_0, t_0, U) = \Theta(X_f, t_f) + \int_{t_0}^{t_f} \phi\left(X(t), U(t), t\right) dt \tag{III.11}$$

avec t_f le temps final et X_f l'état du système à t_f. $\Theta(X_f, t_f)$ représente la fonction coût terminal. $\phi\left(X(t), U(t), t\right)$ est le coût à chaque instant sur la trajectoire $X(t)$.

La commande optimale est :

$$U^* = \arg\min_U J(X_0, U, t_0) \tag{III.12}$$

La valeur optimale du critère est notée :

$$J^*(X_0, t_0) = J(X_0, U^*, t_0) \tag{III.13}$$

On peut formuler des contraintes sur le temps final t_f, sur l'état final X_f et sur la commande U. Dans le cas d'un système non-linéaire, la formulation du critère de performance J rend difficile son optimisation dans un temps limité.

Soit t_1 tel que $t_0 < t_1 < t_f$. Le principe d'optimalité de Bellman énonce que la trajectoire optimale sur $[t_0, t_f]$ contient la trajectoire optimale sur $[t_1, t_f]$ avec comme condition initiale $x(t_1)$. On a donc :

$$J^*(X_0, t_0) = \min_{U_{[t_0,t_1]}} \left[\int_{t_0}^{t_1} \phi\left(X(t), U(t), t\right) dt + J^*(X_1, t_1) \right] \tag{III.14}$$

Ce principe permet d'obtenir la solution optimale en découpant l'intervalle et résolvant un problème récursif.

L'équation de Hamilton–Jacobi–Bellman énonce que :

$$\frac{\partial J^*(X(t), t)}{\partial t} + \min_U \left[\nabla J^*(X, t).f(X, U) + \phi\left(X(t), U(t), t\right)\right] = 0 \tag{III.15}$$

Cette équation caractérise la solution optimale du problème. Mais elle ne peut pas être résolue analytiquement dans la plupart des cas, des méthodes numériques doivent être utilisées.

On définit l'hamiltonien du système par :

$$H(X, U, p, t) = \phi\left(X(t), U(t), t\right) + p^T f(X, U, t) \tag{III.16}$$

p est appelé état adjoint. Le principe du minimum de Pontryagin énonce que la trajectoire optimale minimise H, soit :

$$\forall U, H(X, U^*, p, t) \leqslant H(X, U, p, t) \tag{III.17}$$

Le principe du minimum de Pontryagin donne les conditions nécessaires d'optimalité et l'équation de Hamilton-Jacobi-Bellman donne les conditions suffisantes d'optimalité.

Les démonstrations et des explications complémentaires sont disponibles par exemple dans [LaValle, 2006]. La solution analytique de ces équations n'est disponible que dans certains cas particuliers et est difficilement calculable dans le cas de dynamiques non-linéaires ou de contraintes non convexes.

La commande linéaire quadratique (LQ) étudie la commande optimale dans le cas d'un système linéaire, écrit sous la forme :

$$\dot{X} = A(t)X + B(t)U \tag{III.18}$$

La fonction de coût s'écrit alors :

$$J(X_0, t_0, U) = \frac{1}{2}X_f^T S X_f + \int_{t_0}^{t_f} \frac{1}{2}\left(X(t)^T Q(t) X(t) + U(t)^T R(t) U(t)\right) dt \tag{III.19}$$

Q et R sont des matrices de pondérations, définies positives.

En écrivant :

$$p(t) = P(t)X(t) \tag{III.20}$$

et en reprenant les équations (III.15) et (III.17), avec la condition $S = P(t_f)$ (système à horizon fini), on obtient le retour d'état :

$$U(t) = -R(t)^{-1}B(t)^T P(t) X(t) \tag{III.21}$$

L'utilisation d'un système linéaire permet d'obtenir une solution analytique pour la commande optimale du système. Ce correcteur est l'extension optimale des correcteurs par retour d'état mentionnés dans la section précédente.

Des lois de commande optimale sont utilisées pour le guidage de missile, pour augmenter l'observabilité du système [Hull et al., 1990] ou pour imposer l'angle d'interception [Shaferman and Shima, 2008, Ryoo et al., 2005].

Dans [Bouabdallah et al., 2004], un contrôleur LQ est développé pour contrôler un drone. Le système non-linéaire est linéarisé autour du point d'équilibre. Mais les résultats obtenus sont mitigés car la dynamique des actionneurs n'est pas prise en compte dans la commande. Un deuxième contrôleur de type PID a également été testé afin de stabiliser l'orientation du drone, ce correcteur est efficace à condition que les perturbations sur le système soient faibles. [Hemami et al., 1992] proposent une loi de commande pour le suivi de trajectoire pour un robot terrestre. Le critère cherche à minimiser l'erreur sur la position et l'orientation du robot pour déterminer l'angle de direction optimal. La commande optimale a également été utilisée pour trouver des trajectoires qui minimisent la consommation d'énergie pour un robot [Tokekar et al., 2014].

[Pharpatara, 2015] étudie la planification de trajectoire avec un algorithme RRT* en environnement complexe pour des aéronefs avec prise en compte de contraintes diverses. Pour s'assurer que la trajectoire est faisable, il propose une approche qui consiste à générer des trajectoires réalistes à partir d'une loi de commande optimale afin d'obtenir directement la séquence de commandes correspondant à la trajectoire optimale.

L'approche par commande optimale est moins utilisée en robotique mobile que les approches se basant sur la hiérarchie planification puis commande pour le suivi de la trajectoire. Une autre stratégie de commande régulièrement utilisée dans ce domaine est la commande prédictive, présentée dans la section suivante.

III.2.2.3 Commande prédictive

La commande prédictive est une stratégie de commande discrète qui optimise, à chaque pas de temps, la trajectoire future du système sur un horizon de prédiction fini, à partir de la connaissance du modèle cinématique du système. Contrairement à la commande optimale qui cherche une solution analytique globale ou une solution approchée, la commande prédictive calcule une séquence de commandes optimales qui est remise à jour à chaque pas de temps.

Le modèle dynamique discrétisé du système est décrit par :

$$X_n = f(X_{n-1}, U_{n-1}) \tag{III.22}$$

A partir de ce modèle et d'une séquence de commandes donnée $\mathcal{U}_n$, on obtient la séquence des états futurs $\mathcal{X}_n$ à l'instant actuel t_n sur l'horizon de prédiction.

Une fonction de coût $J(\mathcal{U}_n, \mathcal{X}_n)$ est définie pour représenter les objectifs de commande. Cette fonction est optimisée afin d'obtenir la séquence de commandes optimales $\mathcal{U}_n^*$ sur l'horizon de prédiction :

$$\mathcal{U}_n^* = \arg\min_{\mathcal{U}} J(\mathcal{U}_n, \mathcal{X}_n) \tag{III.23}$$

Seule la première composante de $\mathcal{U}_n^*$ est appliquée et le processus est répété au pas de temps suivant afin de prendre en compte les nouvelles informations acquises par le système.

Cette stratégie de commande peut considérer des systèmes non-linéaires et des contraintes diverses. La fonction de coût J peut prendre en compte un objectif multi-critères.

La principale difficulté de cette méthode se trouve dans l'optimisation de la fonction de coût J. Si J est linéaire ou convexe, l'optimisation peut être réalisée très rapidement par des algorithmes classiques d'optimisation tels que l'algorithme de descente de gradient. Par contre, quand la fonction de coût est non-convexe, il faut mettre en œuvre des algorithmes d'optimisation globale qui assurent le calcul d'une solution dans un temps limité.

Une revue sur le MPC peut être trouvée dans [Findeisen et al., 2003]. Des éléments sur la convergence dans le cas non-linéaire sont donnés par [Jadbabaie et al., 2001].

Cette stratégie de commande est régulièrement utilisée pour des missions de navigation robotique en environnements complexes. Elle a beaucoup été appliquée pour le suivi de trajectoire par un robot. [Howard et al., 2010] ont développé une loi de commande pour effectuer le suivi d'une trajectoire présentant des discontinuités avec évitement d'obstacles dans des environnements accidentés. [Künhe et al., 2005] comparent des lois de commande linéaires et non-linéaires. La formulation non-linéaire donne un problème d'optimisation non-convexe. Le système linéaire est moins précis mais permet d'obtenir un coût d'optimisation plus faible pour des performances similaires. Dans [Kanjanawanishkul and Zell, 2009], le suivi de trajectoire est effectué par un robot omnidirectionnel. La commande prédictive permet de prendre en compte les contraintes sur le robot et la trajectoire tout en gardant une vitesse de déplacement élevée. Dans [Klančar and Škrjanc, 2007], la fonction de coût est formulée pour pénaliser les erreurs

de suivi et réduire l'effort de commande. Le système développé est comparé avec un contrôleur par retour d'état et obtient de meilleurs résultats.

D'autres types de missions ont également été considérés comme les missions d'exploration autonome avec SLAM [Leung et al., 2006] ou l'exploration de zone par une flotte de drones en coopération [Gorecki et al., 2013, Rochefort et al., 2014]. La commande prédictive a été également utilisée pour prendre en compte les contraintes liées à l'environnement, par exemple pour la planification temps-réel pour des véhicules tout terrain avec prise en compte des contraintes sur la traversabilité, la dynamique du véhicule et les obstacles mobiles [Bascetta et al., 2012] ou la génération de trajectoires en environnement urbain [Singh and Fuller, 2001].

III.2.3 Conclusion

Dans cette partie, nous avons vu que les algorithmes de planification de trajectoires présentent des inconvénients (temps de calcul, connaissance a priori de l'environnement, adaptation aux nouvelles informations) incompatibles avec le type de missions considérées dans cette thèse.

La commande réactive permet de prendre en compte ces contraintes. Dans les différentes méthodes présentées, nous avons choisi d'utiliser la commande prédictive. En effet, les méthodes de commande linéaire ne peuvent pas prendre en compte les contraintes liées aux environnements complexes dans lesquels les robots évoluent. La commande optimale donne des contraintes de différentiabilité sur le critère de performance qui rend difficile la formulation d'objectifs variés. De plus, la solution analytique est uniquement disponible dans le cas de systèmes linéaires, or les robots utilisés présentent des modèles dynamiques non-linéaires.

La commande prédictive peut prendre en considération des objectifs multiples et variés, sans contrainte sur leur formulation. Son point faible par rapport à l'utilisation désirée est le temps d'optimisation de la fonction de coût qui peut devenir long quand la formulation de la fonction est complexe. Ce problème nous oblige à adopter des stratégies pour s'assurer d'obtenir une solution dans un temps limité, voir en section IV.3.

III.3 Couplage perception/commande

Cette section se penche sur la problématique de couplage entre la perception et la commande. Habituellement, ces deux processus sont réalisés de manière indépendante, en supposant des systèmes où les estimations sont toujours faisables et exactes. Cependant, la qualité des observations futures dépend de la commande effectuée, et la qualité de commande dépend de la précision dans l'estimation des paramètres.

Ce problème a été étudié dans le cadre de l'automatique, sous le nom de commande duale (section III.3.1). Elle introduit un formalisme mathématique pour résoudre le problème, mais les solutions analytiques n'existent que dans des cas particuliers.

En robotique, ces problématiques ont été étudiées dans le cadre de missions de navigation où la trajectoire suivie par le robot est optimisée afin d'améliorer la qualité des

données perçues par les capteurs (section III.3.2), ce qui est le cadre de cette thèse.

III.3.1 Commande adaptative et duale

En automatique, dans le cas général, on applique habituellement les principes de séparation et d'équivalence à la certitude. Le principe de séparation fait l'hypothèse de découplage entre les tâches d'estimation des paramètres et de commande. Le principe d'équivalence à la certitude suppose que les paramètres estimés peuvent être utilisés pour le calcul de commande sans prendre en compte les incertitudes sur l'estimation. Mais ces hypothèses ne sont plus valides dans le cas où les incertitudes sur les paramètres sont importantes. La commande calculée est alors erronée et peut même s'avérer dangereuse pour le système.

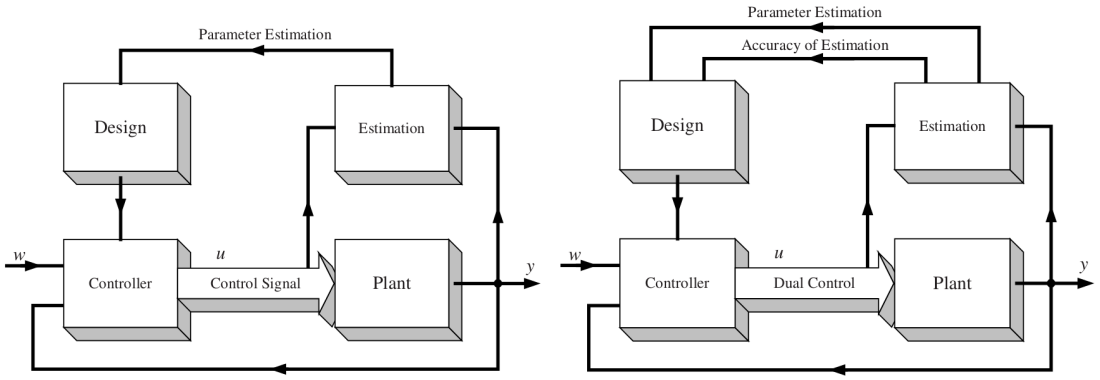
La commande adaptative et duale consiste en l'élaboration d'une commande prenant en compte son influence sur la qualité des observations futures. Elle a été introduite par [Feldbaum, 1961]. Elle concerne également les situations où les paramètres du système sont inconnus ou imprécis. Dans ce cas, la commande est optimisée en vue de mieux identifier ces paramètres. Des revues sur cette stratégie de commande sont disponibles dans [Wittenmark, 1995] et [Unbehauen, 2000] et dans le livre de [Filatov and Unbehauen, 2004].

La commande adaptative prend en considération les incertitudes sur l'estimation des paramètres dans le calcul de la commande avec pour objectif la réduction de celles-ci. En présence d'incertitudes importantes sur les estimations, on veut s'assurer que la commande calculée soit prudente, i.e. que le système reste dans les limites de fonctionnement. Pour cela, une solution est de calculer la commande en fonction de l'importance des incertitudes sur les paramètres : plus les incertitudes sont importantes et plus on limite l'amplitude du signal de commande.

[Ameho et al., 2013] ont développé un système de commande adaptatif pour drones, qui cherche à identifier en ligne les valeurs des paramètres physiques du véhicule, inconnus a priori. Un algorithme de moindres carrés récursifs est utilisé pour l'estimation de ces paramètres. Le contrôleur implémenté, basé sur une commande LPV (Linear Parameter Varying), prend en compte la covariance de l'estimation des paramètres pour atténuer le signal de commande en fonction de l'importance des incertitudes sur les paramètres.

Cependant dans le cas où les incertitudes sont très importantes, le contrôleur peut devenir trop prudent et le signal de commande devenir quasi-nul. L'objectif de commande n'est alors pas réalisé. Il est donc nécessaire, en plus de la réalisation d'une commande prudente, de commander le système dans le but d'obtenir des observations supplémentaires permettant de réduire les incertitudes sur les paramètres et ainsi d'atteindre l'objectif de commande. La commande adaptative et duale présente ce double objectif : d'une part, la réalisation de l'objectif de commande et d'autre part, l'excitation du système dans le but d'améliorer l'estimation des paramètres. La figure III.11 montre le fonctionnement d'une boucle de commande non-duale et d'une boucle de commande duale.

La solution formelle au problème de commande duale est formulée par la Programmation Dynamique Stochastique. Elle consiste en la minimisation d'un critère J_n traduisant l'objectif sur N pas, qui est formulé comme l'espérance d'un critère classique de



(a) Boucle de commande non duale (b) Boucle de commande duale

FIGURE III.11 – Fonctionnement des commandes duale et non-duale. Illustrations issues de [Filatov and Unbehauen, 2004].

commande optimale à partir des informations disponibles I_n à l'instant n :

$$U_n* = \arg\min E\left[J_n|I_n\right] \tag{III.24}$$

Le principe d'optimalité de Bellman énonce que toutes les décisions restantes doivent constituer une politique optimale par rapport à l'information courante, c'est-à-dire le critère à l'instant $n-1$ doit prendre en compte les critères futurs pour le calcul de la commande optimale. La commande optimale à $n-1$ s'écrit alors :

$$U_{n-1}* = \arg\min E\left[\ \min E\left[J_n|I_n\right]\ \ |I_{n-1}\right] \tag{III.25}$$

et ainsi de suite pour tout n entre 0 et $N-1$.

La solution analytique de cette équation s'obtient uniquement dans des cas spécifiques [Bar-Shalom and Tse, 1976, Bayard and Eslami, 1985]. Une solution possible est d'étudier des problèmes sous-optimaux comme :

— l'optimisation sur un pas uniquement

— le développement limité de la fonction de coût

— la simplification de la fonction de coût

— l'optimisation sur un espace réduit

[Murray-Smith et al., 2003] utilisent par exemple un a priori de répartition gaussienne des incertitudes sur les paramètres dans leurs travaux. Ils obtiennent une loi de commande analytique qui permet de suivre la trajectoire de référence de façon prudente. [Rathouskỳ and Havlena, 2013] ont créé un système imitant un comportement dual. Ce système est sous-optimal mais permet de donner une solution au problème. Une commande optimale est calculée par un algorithme MPC dans un premier temps. Afin d'exciter le système, une solution qui maximise la plus petite valeur propre de la matrice d'information est ensuite cherchée dans le voisinage.

Dans le cadre de suivi de cible, seules les mesures d'angle sont disponibles. Il est alors nécessaire de dégrader volontairement la trajectoire du système afin de réaliser plusieurs mesures à partir d'angles de vue différents et ainsi d'avoir la possibilité de calculer l'état de la cible. Ce problème donne un compromis entre l'excitation du système par la dégradation de la trajectoire et l'objectif de suivi de la cible. Il a par exemple été appliqué pour le guidage de missiles [Song and Um, 1996], [Hull et al., 1990].

III.3.2 Navigation robotique "active"

Le robot a besoin de bonnes observations pour estimer précisément sa localisation dans l'espace et reconstruire l'environnement. Si la localisation ou la carte de l'environnement sont imprécises, la navigation devient risquée pour le robot et la mission peut ne pas être accomplie. La qualité des informations perçues par les capteurs portés par le robot dépend de la trajectoire suivie. Le choix de la trajectoire à suivre va donc dépendre de la qualité prédite des observations futures.

Comme nous l'avons dit précédemment, la robotique d'exploration autonome utilise essentiellement deux types de capteurs extéroceptifs : les capteurs actifs comme le LIDAR ou le SONAR ou les capteurs de vision. Suivant le capteur utilisé, la problématique de couplage perception-commande se traduit différemment.

Dans les deux cas, on trouve une problématique commune de réduction des incertitudes sur les paramètres estimés (position du robot et/ou des amers dans la carte). La trajectoire future est optimisée en fonction de la possibilité de réduire ces incertitudes. Cette démarche est appelée navigation active ou SLAM actif ou même exploration active, suivant la mission considérée pour le robot. Les incertitudes sont systématiquement formulées sous la forme de matrice de covariance. Pour pouvoir les minimiser, il est alors nécessaire de formuler des métriques scalaires. De nombreuses métriques ont été proposées dans la littérature, elles sont décrites en section III.3.2.1.

Les capteurs actifs permettent une mesure directe de la position des amers. L'enjeu du couplage perception/commande se situe uniquement sur la réduction des incertitudes sur la position des amers et du robot pendant le déplacement. La section III.3.2.2 se penche sur la littérature portant sur l'utilisation de capteurs LIDAR ou SONAR.

Les capteurs de vision présentent une difficulté supplémentaire car l'estimation des paramètres n'est pas réalisée directement à partir des données reçues mais nécessite des traitements d'image, comme ceux décrits en section III.1.3 précédente pour l'approche par odométrie visuelle stéréo. Or, quand la qualité de l'image est insuffisante, aucune information ne peut en être extraite. Deux problématiques sont donc étudiées dans le domaine de la vision, la première est la réduction des incertitudes, comme avec les capteurs actifs. La deuxième est le choix d'une trajectoire qui assure la réception d'images de qualité suffisante pour permettre le calcul d'une localisation précise. La littérature portant sur ces problématiques est présentée dans la section III.3.2.3.

III.3.2.1 Métriques

Le but des métriques présentées dans cette section est de formaliser sous la forme d'une mesure l'importance des incertitudes sur les paramètres estimés. Ces métriques sont optimisées afin de réduire au maximum les incertitudes sur les paramètres. Elles sont formulées soit directement à partir de la matrice de covariance sur les paramètres, soit à partir de l'entropie de la distribution de probabilité des paramètres.

Pour minimiser l'incertitude sur les paramètres, on définit souvent une métrique scalaire sur la matrice de covariance. Les métriques suivantes peuvent être utilisées [Walter and Pronzato, 1997] :

— D-optimalité : minimisation du déterminant de la matrice de covariance, ce qui correspond au produit des valeurs propres. Cette valeur est proportionnelle au volume de l'ellipsoïde correspondant à la matrice de covariance.

— A-optimalité : minimisation de la trace de la matrice de covariance, ce qui correspond à la somme des valeurs propres. Cette valeur est proportionnelle à la somme des longueurs des axes de l'ellipsoïde.

— E-optimalité : minimisation de la plus grande valeur propre de la matrice de covariance.

D'après [Sim and Roy, 2005], il est préférable d'utiliser l'A-optimalité (calcul de la trace) plutôt que la D-optimalité (calcul du déterminant). En effet, le déterminant peut être faible si une seule des valeurs propres est petite même si les autres sont grandes. Dans cette situation, l'ellipsoïde représentant la covariance est presque "plat", le volume est donc faible mais les incertitudes peuvent tout de même être très importantes dans certaines directions. Utiliser la trace permet de s'assurer que la recherche de la valeur minimale est effectuée dans toutes les directions.

En théorie de l'information, plusieurs métriques ont été développées à partir du calcul de l'entropie. Ici, X représente une variable aléatoire, potentiellement multidimensionnelle. Elle peut être par exemple la position du robot dans l'espace.

— L'entropie mesure la quantité d'information contenue dans une distribution $p(X)$. Elle représente quantitativement l'incertitude sur l'information associée à la réception d'une réalisation de la variable X.

$$H(X) = -\sum p(X) \log p(X) \tag{III.26}$$

Dans le cas d'une distribution gaussienne, l'entropie s'écrit :

$$H(X) = \frac{1}{2} \log \left((2\pi e)^m |\Sigma_X| \right) \tag{III.27}$$

avec m la dimension de X. $|\Sigma_X|$ est le déterminant de Σ_X. Une manière de minimiser l'incertitude sur X peut alors être de minimiser l'entropie, ce qui revient dans le cas gaussien, d'après (III.27) à une notion de D-optimalité.

En pratique, on cherche souvent à juger de l'apport d'une action sur l'information disponible sur un paramètre, on a alors besoin d'outils pour comparer deux états d'information.

— La différence d'entropie fournit une mesure de gain d'information entre deux instants t_n et t_{n+1}

$$I_{n,n+1}(X) = H_n(X) - H_{n+1}(X) \tag{III.28}$$

Maximiser le gain d'information à l'instant t_n équivaut à chercher l'action qui apporte le plus d'information au système à t_{n+1}.

— L'entropie conditionnelle mesure l'information apportée sur X par la connaissance d'une autre quantité Z

$$H(X|Z) = -\sum p(X,Z) \log p(X|Z) \tag{III.29}$$

— En combinant les deux notions précédentes, on parvient à la notion d'information mutuelle qui indique le changement d'état d'information apporté par la mesure de Z sur la variable X.

$$I(XZ) = H(X) - H(X|Z) \tag{III.30}$$

Si la distribution est gaussienne,

$$I(XZ) = \frac{1}{2} \log \frac{|\Sigma_X|}{|\Sigma_{X|Z}|} \tag{III.31}$$

— L'information de Fisher est la différentiation locale de l'information de Shannon

$$\Gamma(X) = E_Z \left\{ \left(\frac{\partial \log p(X,Z)}{\partial X} \right)^T \left(\frac{\partial \log p(X,Z)}{\partial X} \right) \right\} \tag{III.32}$$

L'inégalité de Cramer-Rao donne une borne inférieure à la covariance de l'estimation $\hat{X}$:

$$\Sigma_{\hat{X}} \geq \Gamma(X)^{-1} \tag{III.33}$$

Une valeur propre de $\Gamma(X)$ élevée correspond donc à une meilleure localisation dans la direction du vecteur propre correspondant.

La matrice d'information de Fisher a par exemple été utilisée dans [Ucinski, 2000] pour planifier le déplacement de plusieurs capteurs afin de maximiser la précision de l'identification des paramètres d'un système distribué.

III.3.2.2 Capteurs LIDAR-SONAR

Les travaux présentés dans cette section se penchent sur l'utilisation de capteurs actifs sur des plateformes robotiques afin de réaliser des missions diverses en prenant en compte la possibilité de réduire les incertitudes sur la position des amers et/ou du robot en fonction de la trajectoire choisie.

Différents types de mission sont considérées, une partie des travaux porte sur la planification de trajectoire en environnement connu, le but est alors de réduire les incertitudes sur la localisation du robot. D'autres références se penchent sur des missions d'exploration en environnement inconnu. Dans ce cas, les études portent sur la réduction simultanée des incertitudes sur la position du robot et des amers 3D.

Dans le cadre de la navigation en environnement connu, le but est de faire naviguer le robot et de garder une localisation la plus précise possible. Une première solution étudiée dans [He et al., 2008] et [Valencia et al., 2011] est de modifier le graphe utilisé pour la planification de trajectoire en ajoutant sur les nœuds un paramètre sur la qualité de la localisation à l'endroit correspondant. Ensuite, un algorithme cherche le chemin optimal en prenant en compte ce paramètre. [He et al., 2008] travaillent sur la navigation avec des drones. La position est estimée en utilisant un filtre UKF (Unscented Kalman Filter) qui permet de pallier les faibles performances d'un filtre EKF dans le cas de fortes non-linéarités dans le modèle. Pour la planification de trajectoire, les auteurs ont utilisé un algorithme appelé Belief Roadmap (BRM), inspiré de l'algorithme de Probabilistic Roadmap, décrit dans la section III.2.1. Un paramètre sur la norme de la covariance est calculé pour chaque nœud du graphe, à partir d'un modèle de prédiction de la mesure. Ce critère est pris en compte dans la recherche de chemin optimal dans le graphe. Dans [Valencia et al., 2011], une planification de trajectoire sur une carte connue de type Pose SLAM est effectuée pour obtenir le chemin qui présente l'incertitude accumulée minimale sur la position du robot. Le critère à minimiser est basé sur l'entropie conditionnelle de la position future à partir de la connaissance de la position actuelle. [Roy et al., 1999] s'intéressent également à la navigation en environnement connu mais dans le cas où celui-ci est dynamique. Les objets mobiles peuvent rendre difficile le calcul de la localisation du robot. La planification de trajectoire présente un double objectif : minimisation de la distance à parcourir et certitude de pouvoir se localiser correctement. Ce deuxième objectif prend en compte la probabilité que les données capteurs soient corrompues par des obstacles mobiles et cherche à maximiser l'espérance de l'entropie sur les positions futures. [Huang et al., 2005] effectuent de la planification de trajectoire avec un EKF-SLAM dans un environnement inconnu. L'objectif est de minimiser l'erreur d'estimation sur la position du robot et des amers sur un horizon de temps fixé, en minimisant la trace de la matrice de covariance prédite dans l'EKF.

Dans le contexte de missions d'exploration, l'objectif de la réduction active des incertitudes est différent. Le but est de minimiser les incertitudes sur la position du robot et sur la position des amers utilisés dans le processus de SLAM. La précision de la zone explorée dépend de la précision de la localisation, ce qui donne un compromis entre explorer des zones inconnues et dégrader la qualité de la localisation du robot, ou revenir sur des zones déjà explorées pour diminuer l'incertitude sur la localisation. [Bourgault et al., 2002] cherchent la commande qui maximise un objectif multi-critères de diminution des incertitudes sur le SLAM et d'attraction du robot vers les zones inconnues dans le but d'effectuer des missions d'exploration sur des robots terrestres. Dans [Makarenko et al., 2002], les auteurs s'intéressent au couplage entre les tâches de localisation, de cartographie et de commande. L'algorithme développé définit des points potentiels à rallier, générés selon plusieurs critères : le gain d'information à l'arrivée, le coût pour atteindre le point et la capacité de se localiser à la position d'arrivée. Le point qui présente le coût le plus important est rallié et le processus est réitéré. [Feder et al., 1999] sélectionnent l'action qui permet d'obtenir l'information maximale à partir d'un capteur SONAR et un EKF-SLAM. Une métrique basée sur l'information de

Fisher, représentant la somme des aires des ellipses de covariance sur les estimations de la position du véhicule et des amers est minimisée pour obtenir l'action optimale à effectuer pour le robot. [Stachniss et al., 2005] utilisent un filtre particulaire pour représenter la distribution a posteriori de la carte et de la position du robot. Les actions potentielles sont évaluées à partir du gain d'information entropique sur le filtre particulaire et la prise en compte des observations possibles sur la trajectoire future.

Les problématiques de réduction active des incertitudes ont beaucoup été étudiées dans le contexte du SLAM LIDAR et permettent d'améliorer la précision de la localisation du robot et de la carte construite pendant l'exploration. Diverses missions ont été considérées mais le principe repose toujours sur la propagation des incertitudes sur la position du robot et des amers et la minimisation d'un critère lié à la covariance future. Ces techniques sont adaptables pour tout type de SLAM, à partir du moment où les incertitudes sur les mesures sont évaluées. Elles ont été appliquées par exemple au SLAM visuel. Mais l'utilisation des images pour le calcul de la localisation amène de nouvelles problématiques qui n'apparaissent pas avec les capteurs actifs.

III.3.2.3 Capteurs de vision

Depuis le début des années 2000, l'utilisation de capteurs de vision pour la navigation autonome de plateformes robotiques est de plus en plus étudiée. Quelques auteurs se sont intéressés à la problématique du couplage perception-commande avec ce type de capteurs. Une première partie des travaux dans ce domaine porte sur la réduction des incertitudes apparaissant dans le processus du SLAM visuel, sur des logiques très proches des travaux sur les capteurs actifs décrits précédemment. Différents objectifs de mission sont considérés également dans cette partie : navigation en environnement connu ou inconnu, exploration de zones inconnues et amélioration de la reconstruction de l'environnement.

Pour effectuer des missions de navigation en environnement connu, [Bachrach et al., 2012] travaillent avec un drone équipé d'un capteur RGB-D. L'environnement est décrit sous la forme d'un graphe en utilisant l'algorithme Belief Roadmap. Ils cherchent le chemin optimal dans le graphe qui minimise la distance au point à rallier et respectant une contrainte sur la trace de la matrice de covariance sur la position du drone.

Dans le contexte de navigation dans un environnement inconnu, [Vidal-Calleja et al., 2006] sélectionnent l'action qui permet de minimiser conjointement les incertitudes sur la position des amers et la distance au point à rallier dans un objectif global de navigation par points de passage avec évitement des obstacles fixes. Pour réduire les incertitudes, l'information mutuelle est maximisée, ce qui équivaut à choisir la commande qui réduit le plus l'incertitude sur la position de la caméra et des amers à partir des mesures. Pour l'évitement d'obstacle, une méthode de champs de potentiel artificiels est utilisée. Le champ attracteur est défini par le point à rallier et le champ répulseur par les obstacles.

Pour la réalisation de missions d'exploration, on retrouve le compromis entre l'exploration et la réduction des incertitudes sur la localisation du robot. Dans [Kim and Eustice, 2013], les auteurs s'intéressent à la problématique de couverture de zone pour des robots sous-marins utilisant le SLAM. Le but est d'explorer une zone prédéfinie en un temps minimal,

ce qui donne un compromis déjà évoqué entre revisiter les amers connus pour améliorer la précision de la localisation ou continuer l'exploration et finir la mission dans le temps défini. Le coût à minimiser considère la covariance prédite de la position du véhicule et le ratio de couverture de la zone. [Bryson and Sukkarieh, 2008] effectuent une planification de trajectoire qui cherche à maximiser la précision de la carte et de la position du véhicule pendant l'exploration de zones inconnues avec un drone. L'optimisation est exécutée sur un pas uniquement. Le critère est basé sur la maximisation du gain d'information de la distribution de probabilité jointe sur la position du véhicule et des amers. Une liste de points potentiels est créée. Pour chaque trajectoire les reliant, le coût est évalué et la trajectoire qui maximise le critère est réalisée.

Une autre problématique considérée est l'amélioration de la précision de la reconstruction de l'environnement. Ce problème est appelé Next-Best-View, le principe est de trouver le prochain point de vue qui permet d'apporter le plus d'information au système dans le but d'avoir la reconstruction de l'environnement la plus précise possible. [Haner and Heyden, 2011] cherchent l'ensemble de vues optimal pour maximiser la précision de la reconstruction 3D d'un objet. Le robot fonctionne avec un SLAM visuel. Un critère combinant la distance parcourue et l'erreur de reconstruction en utilisant la trace de la matrice de covariance sur les amers observés est optimisé. [Forster et al., 2014] cherchent la trajectoire qui fournit la meilleure précision dans la reconstruction 3D de l'environnement. Le critère maximise le gain d'information en prenant en compte l'incertitude sur la profondeur des points, l'incertitude sur la position du robot et la probabilité d'avoir des associations multi-vues correctes. Différentes stratégies de commande sont testées : marche aléatoire, trajectoire circulaire, MPC sur 1 ou N pas, avec N variable dans le temps. Cette dernière stratégie permet d'obtenir la trajectoire la plus informative, et donc d'obtenir la reconstruction la plus précise.

Les références précédentes ont transposé les méthodes utilisées sur le LIDAR et décrites dans la section précédente III.3.2.2 dans le SLAM visuel. Mais, mis à part [Forster et al., 2014] qui incorporent dans leur critère un terme sur la qualité d'association, elles ne se concentrent que sur la réduction des incertitudes sur la position calculée des amers et du robot. Or, le SLAM visuel nécessite des opérations de traitement sur les images (extraction et association de points, suivi temporel dans les images) qui peuvent s'avérer difficiles quand les images reçues sont peu texturées. Si cette étape de traitement n'est pas réalisée correctement, le calcul de la localisation n'est pas précis et dans le cas d'images très peu texturées, le calcul de localisation est même impossible. La qualité de la localisation visuelle dépend donc de la qualité des images reçues par les caméras. La réduction active des incertitudes, comme effectuée dans les références présentées dans ce début de section, est alors insuffisante pour garantir la localisation précise tout au long de la trajectoire.

Dans le paragraphe qui suit, les études citées cherchent à optimiser la trajectoire à suivre en tentant de prédire l'apparence de la scène future afin de s'assurer que les images reçues soient exploitables et donc que le calcul de la localisation visuelle reste précis.

Une première solution a été étudiée par [Davison and Murray, 1998]. Ils ont travaillé

avec un système stéréo orientable monté sur un robot dans le cadre de la navigation avec localisation basée vision. Dans leur travail, deux points sont très importants : le fait de disposer d'une tourelle stéréo orientable et le choix de travailler avec peu d'amers bien localisés et stables dans le temps, plutôt qu'avec beaucoup d'amers moins fiables. Dans ce cadre, leur problème devient le choix du point de fixation parmi les différentes directions correspondant aux amers qui sont maintenus, pour diminuer le plus possible leur incertitude de localisation. Au contraire, dans le présent travail, et d'une manière générale dans la plupart des systèmes robotiques actuels, les systèmes de stéréovision sont fixés sur les robots. D'autre part, l'évolution des algorithmes actuels d'odométrie visuelle favorise le suivi d'un grand nombre d'amers, répartis dans l'ensemble du champ, plutôt que de se focaliser sur quelques uns. Pour ces raisons, le travail de [Davison and Murray, 1998] n'est pas adapté à notre contexte et à celui des références qui suivent.

La prise en compte de l'apparence dans le processus d'optimisation de la trajectoire a été étudiée dans le contexte de la Next-Best-View [Dunn et al., 2009]. Dans ces travaux, on cherche la position future qui maximise la précision de la reconstruction 3D d'un objet. La fonction de coût cherche à minimiser deux critères : l'incertitude de triangulation et l'entropie de l'auto-corrélation de l'image. En ce qui concerne ce second critère, plus les images sont texturées et plus l'auto-corrélation est "piquée" donc son entropie est faible, la minimiser revient donc à chercher l'image la plus texturée.

Pour réaliser des missions de navigation en environnement inconnu en prenant en compte l'apparence de la scène, [Sadat et al., 2014] calculent un critère de texture à partir de la densité locale des triangles du maillage 3D généré par la reconstruction de l'environnement. La logique de ce critère est que plus la texture est contrastée, plus nombreux sont les points qui peuvent être extraits et donc plus le maillage est fin dans la zone 3D reconstruite. Le critère comporte une mesure de la qualité de la texture et une mesure de la visibilité de l'aire ciblée. La trajectoire optimale est ensuite calculée par un algorithme RRT* qui prend en compte ce critère de texture afin de planifier des trajectoires dans un objectif global de ralliement de points.

[Mostegel et al., 2014] effectuent des missions de navigation sur drones portant une caméra seule dans des environnements inconnus. La qualité d'une position future est estimée à partir du nuage de points 3D, obtenu par l'algorithme PTAM (décrit dans la partie III.1.1). Dans la fin de cette section, nous détaillons cette référence qui est la plus proche de nos travaux.

La mesure de la qualité de la localisation est définie pour chaque point par deux critères : la qualité géométrique du point et la probabilité de reconnaissance. La qualité géométrique d'un point préfère les points qui ont été vus plusieurs fois, ce qui prouve leur existence physique, et pénalise les points dont l'angle de triangulation est faible. En effet, plus cet angle est faible et plus l'incertitude sur la profondeur du point 3D triangulé est grande. La probabilité de reconnaissance cherche à limiter l'angle de vue et l'échelle afin que les points puissent être reconnus dans les images futures. La qualité finale d'un point est donnée par la multiplication de tous ces critères.

L'image est divisée en régions. Pour chaque région, un score est calculé, qui est la somme des critères de qualité de tous les points appartenant à la région. La formulation

du score total pour une position donnée évalue la répartition des scores dans l'image. Le but est d'obtenir une bonne répartition dans l'image, ce qui permet un calcul de localisation plus précis. La possibilité de trouver de nouveaux amers est estimée en générant des nouveaux points probables sur les régions de l'image possédant peu de points déjà triangulés mais des points 2D non triangulés, comme le montre la figure III.12. Leur profondeur est estimée en utilisant l'information de profondeur déjà connue grâce aux points triangulés.

Pour naviguer, le robot génère une liste de destinations potentielles dans son voisinage. Une position est choisie en fonction de sa qualité de localisation et de l'absence d'obstacles à cet endroit. La trajectoire joignant la position est évaluée sur la présence d'obstacles et la qualité de localisation. Si la qualité est insuffisante, une autre destination est testée. Ensuite, la trajectoire est suivie par le drone grâce à un contrôleur PID et la procédure est recommencée. Si toutes les destinations potentielles présentent une qualité de localisation insuffisante, l'algorithme génère des nouveaux points probables afin de trouver une nouvelle destination respectant le critère.

III.3.2.4 Conclusion

Dans les travaux réalisés dans le cadre de cette thèse, nous avons choisi de nous intéresser à la prédiction de la qualité de la scène future afin de guider le robot sur une trajectoire où on s'assure que les images reçues seront exploitables. Pour cela, nous nous sommes inspirés de la stratégie développée par [Mostegel et al., 2014] qui consiste à utiliser les points extraits par l'algorithme d'odométrie visuelle afin de prédire ceux qui seront visibles dans les images futures. Contrairement à leurs travaux, qui portent sur l'utilisation d'un système monoculaire embarqué sur le robot, nous nous sommes basés sur les systèmes d'odométrie stéréo disponibles sur les plate-formes ONERA. L'information de profondeur est obtenue directement et offre ainsi la possibilité de caractériser complètement les amers de façon immédiate. Les critères utilisés pour évaluer la qualité d'un point afin de prédire sa visibilité future, comme l'angle de triangulation, ne sont donc plus nécessaires dans le contexte de la stéréovision. La possibilité de connaître directement la profondeur des points rend également inutile la procédure de prédiction complexe et imparfaite proposée par [Mostegel et al., 2014].

De plus, dans les dernières références présentées, qui portent sur la prise en compte de la qualité de l'image, les incertitudes sur la position du robot et des amers ne sont plus considérées. Dans nos travaux, nous avons choisi de les utiliser, non pas pour les réduire de façon active comme dans les références sur le SLAM actif LIDAR, mais pour les prendre en compte dans la prédiction de la scène future afin d'affiner les résultats de prédiction. Ces développements sont décrits dans le chapitre V.

Au niveau de la stratégie de navigation, plutôt que de définir et qualifier des destinations potentielles et de tester ensuite les trajectoires les rejoignant, nous avons choisi d'utiliser une commande MPC qui cherche directement la trajectoire optimale par rapport aux objectifs de la mission. Cette stratégie de commande permet de considérer des objectifs divers (ralliement de points, exploration, évitement d'obstacles, qualité de localisation, etc...). Elle donne la possibilité de formuler un objectif global multi-

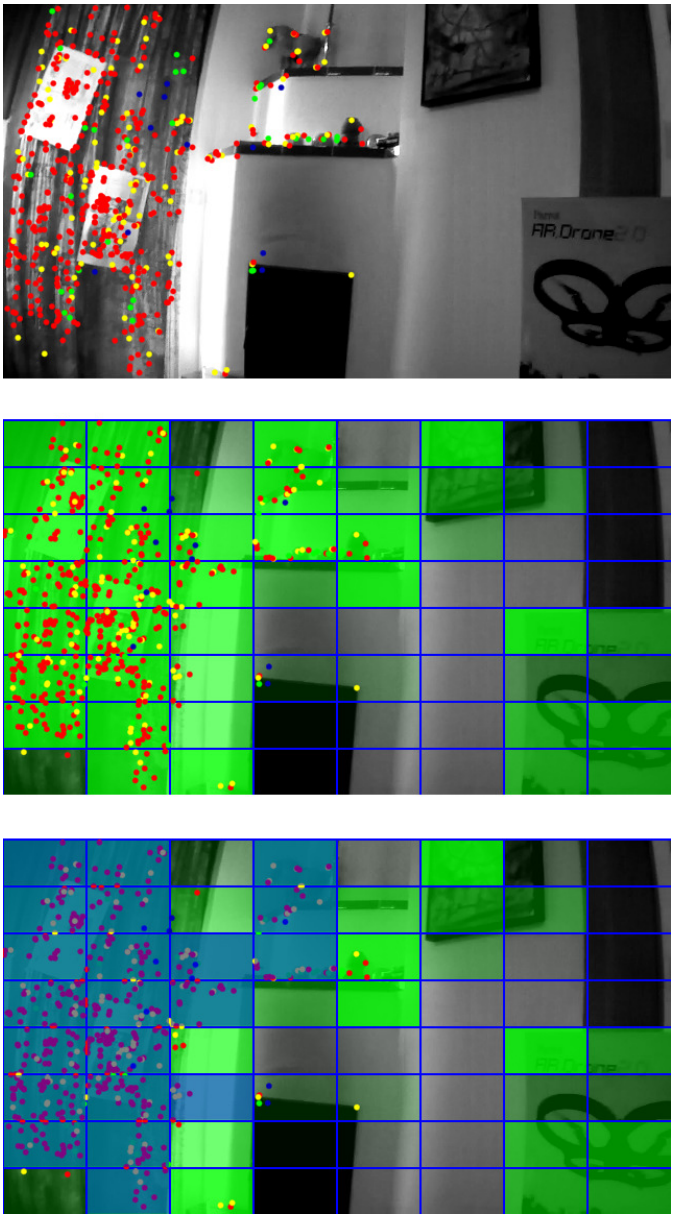


FIGURE III.12 – Répartition des points dans l'image par régions. Illustration issue de [Mostegel et al., 2014]

critères tel qu'un objectif d'exploration de zone avec évitement d'obstacles dans un environnement initialement inconnu. Au contraire, la stratégie de navigation utilisée par [Mostegel et al., 2014] permet d'effectuer uniquement des missions de ralliement de points.

Chapitre IV

Conception d'une boucle de perception-commande pour l'exploration autonome

Sommaire

Ce chapitre décrit l'architecture globale du système développé pour réaliser des missions d'exploration autonome sur un robot terrestre et les expériences qui ont été réalisées en situation réelle pour démontrer le bon fonctionnement du système proposé.

La mission considérée consiste en l'exploration autonome d'une zone inconnue de taille prédéfinie, avec présence éventuelle d'obstacles fixes que le robot doit éviter. Pendant son déplacement, le robot doit construire la carte de l'environnement représentant

les obstacles et les espaces libres. A la fin de la mission d'exploration, le robot doit revenir à son point de départ.

La localisation du robot est assurée uniquement à partir de ses capteurs embarqués : vision, centrale inertielle et odométrie des roues. Le système développé doit garantir un calcul précis de la localisation pendant toute la mission et entre autres gérer les problèmes pouvant affecter la localisation visuelle. Le système propose également des procédures pour gérer les situations où le robot est bloqué dans sa mission et ne peut l'achever.

La section IV.1 donne un aperçu global de l'architecture mise en place pour réaliser cette mission d'exploration.

IV.1 Aperçu de la boucle complète

Une boucle de perception-commande a été développée afin de réaliser la mission proposée. A partir des données reçues par les capteurs, le robot va pouvoir se localiser, construire la carte de l'environnement et se déplacer afin de remplir les objectifs désirés.

La figure IV.1 décrit l'architecture globale du système mis en place.

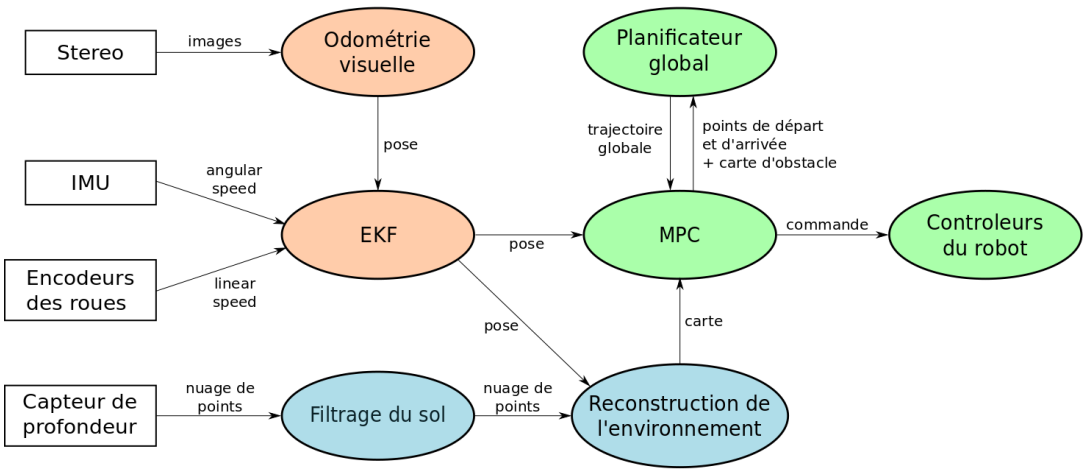


FIGURE IV.1 – Architecture du système. Les rectangles blancs représentent les capteurs embarqués sur le robot. Les cercles sont les modules utilisés dans la boucle de perception-commande. En orange, on trouve les modules de localisation, en bleu, ceux permettant la reconstruction de l'environnement et en vert, les modules de la partie commande.

L'architecture mise en place dépend des capteurs qui sont disponibles sur le robot. Pour effectuer cette mission, les capteurs embarqués sur le robot sont : un système de vision stéréo, l'odométrie des roues et une centrale inertielle utilisés pour la localisation, un capteur de profondeur utilisé pour la reconstruction de l'environnement.

Le module d'odométrie visuelle utilise l'algorithme eVO, décrit en section III.1.4 et permet d'obtenir la localisation (position et orientation du robot) dans l'espace à partir

des images reçues par les caméras du banc stéréo.

Pour remédier aux problèmes pouvant survenir avec la localisation visuelle, décrits dans la section III.1.5, un filtre de Kalman étendu permet de réaliser une fusion de données entre les données de localisation visuelle et les données de la centrale inertielle et de l'odométrie des roues. Il est décrit dans la section IV.4.

Pour pouvoir construire le plan de la pièce et éviter les obstacles pendant son déplacement, un module de reconstruction de l'environnement a été développé. Il est présenté dans la section IV.2. La partie reconstruction de l'environnement se décompose en deux tâches distinctes. La première tâche est le filtrage du plan du sol dans les données envoyées par le capteur de profondeur. La deuxième tâche consiste ensuite à construire la carte de l'environnement.

Une stratégie de commande basée sur la commande prédictive (MPC) a été développée pour calculer la commande que le robot doit appliquer. Elle est décrite en section IV.3. Ce module utilise les données de localisation envoyées par le module EKF et la carte de l'environnement envoyée par le module de reconstruction de l'environnement afin de calculer la commande optimale à effectuer.

Un planificateur global permet d'aider le robot à sortir des situations de blocage, qui peuvent apparaître à cause de l'horizon local ou de la complexité de l'environnement, il est présenté en section IV.5.2.

IV.2 Reconstruction de l'environnement

Pour pouvoir se déplacer en toute sécurité, le robot a besoin d'une carte de l'environnement représentant les espaces libres et les obstacles. Cette carte sera ensuite utilisée pour définir des trajectoires évitant les obstacles. Le robot a également besoin d'une carte de l'exploration pour connaître les zones déjà explorées ou non et ainsi définir une trajectoire future qui permet l'exploration des zones inconnues.

La perception de l'environnement est effectuée grâce au capteur de profondeur installé à l'avant du robot. Il modélise l'environnement proche sous la forme d'un nuage de points 3D, tel que représenté sur la figure IV.2a.

Dans ce nuage de points, le plan du sol est filtré afin de ne pas le considérer comme un obstacle. Cette opération est réalisée par une procédure RANSAC (voir section III.1.4). La procédure permet d'estimer les paramètres du plan du sol pour ensuite distinguer les points qui appartiennent à ce plan des autres points. L'équation cartésienne du plan est donnée par :

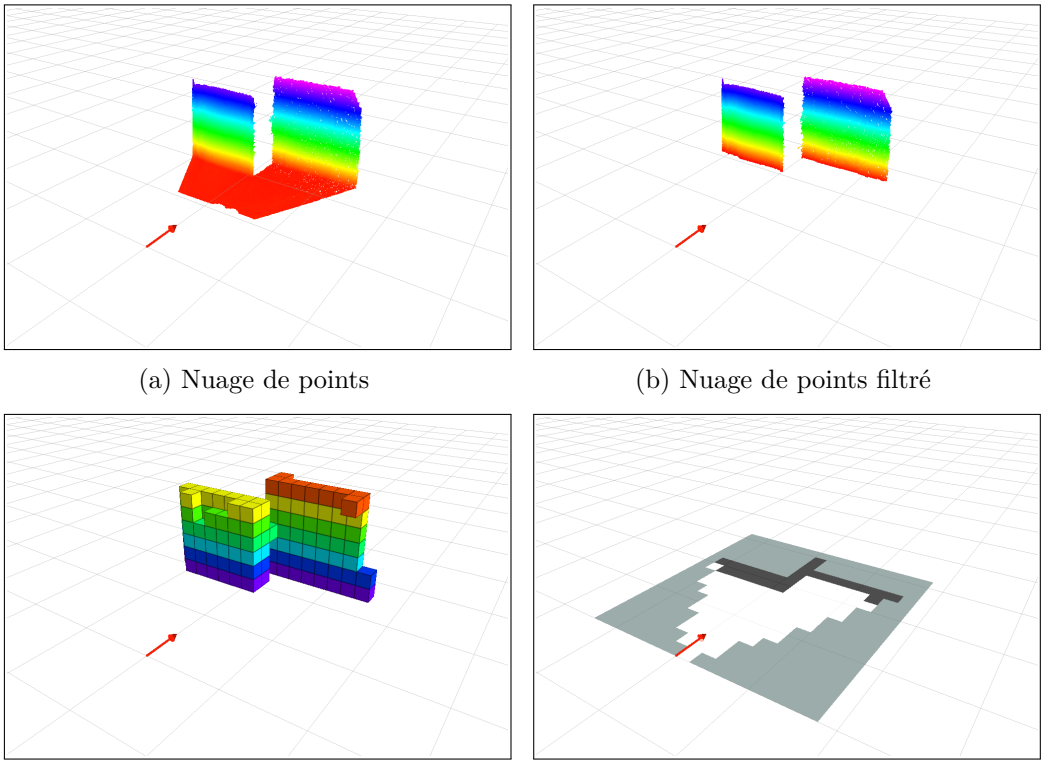
$$ax + by + cz + d = 0 \tag{IV.1}$$

avec $M(x, y, z)$ un point quelconque de l'espace. Le but de la procédure est de déterminer les valeurs des paramètres (a, b, c, d). Le capteur de profondeur est fixe sur le robot donc présente une distance et une orientation constantes par rapport au plan du sol. Ce qui permet de donner un a priori sur la valeur des paramètres du plan du sol dans la procédure d'estimation. On s'assure ainsi qu'on estime bien les paramètres du plan du

sol et non les paramètres d'un autre plan dans la scène, comme un mur. Les valeurs a priori des paramètres sont $(a = 0,\ b = 0,\ c = 1,\ d = 0)$.

Après avoir estimé les paramètres, les points considérés comme appartenant au sol sont retirés du nuage de points. Pour cela, on considère tous les points qui sont situés à une distance au plan inférieure à une certaine valeur. Un nouveau nuage de points 3D est obtenu, comme présenté sur la figure IV.2b. Ces algorithmes ont été développés en utilisant la bibliothèque PCL (Point Cloud Library) qui fournit des fonctions de traitement sur les nuages de points [Rusu and Cousins, 2011].

Le nuage de points filtré est ensuite transformé en un modèle Octomap. Dans ce modèle, l'environnement est représenté sous forme de voxels, avec trois états possibles, libre, occupé ou inconnu, voir en section II.3.2. Le résultat obtenu est visible sur la figure IV.2c.



(a) Nuage de points

(b) Nuage de points filtré

(c) Octomap

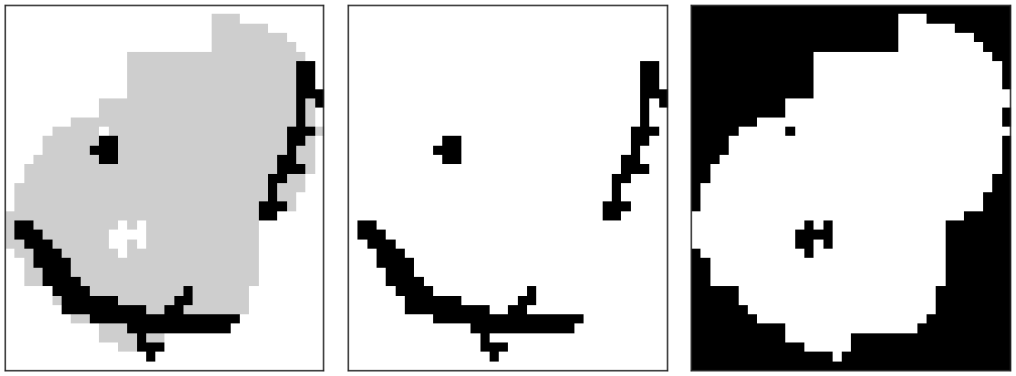
(d) Grille d'occupation 2D

FIGURE IV.2 – Images du processus de reconstruction de l'environnement. La flèche rouge inque la position et l'orientation du capteur de profondeur

Dans la mission considérée, le robot se déplace uniquement sur le plan du sol. Par conséquent, seule l'occupation de l'espace sur le plan (xy) est nécessaire. Les voxels occupés dans le modèle Octomap sont projetés sur ce plan. On obtient alors une grille d'occupation qui peut contenir trois valeurs différentes : 0 si la cellule est libre, 1 si la cellule est occupée et -1 si l'occupation de la cellule est inconnue, voir figure IV.2d. Cette

grille d'occupation est mise à jour au fur à mesure de l'exploration de l'environnement par le robot.

Deux cartes sont ensuite créées à partir de la grille d'occupation. La première est une carte d'obstacle. Un pixel vaut 0 si l'emplacement est libre et 1 s'il y a un obstacle. Cette carte sera utilisée pour l'évitement d'obstacles, voir IV.3.3.3. La deuxième carte est une carte d'exploration. Un élément vaut 1 s'il a été exploré, c'est-à-dire qu'il a été vu par le capteur embarqué au moins une fois, et 0 s'il n'a pas été exploré. Cette carte permet de définir la zone qui peut être explorée à partir d'une trajectoire, voir partie IV.3.3.4.



(a) (b) (c)

FIGURE IV.3 – Une grille d'occupation (a), la carte d'obstacle (b) et la carte d'exploration (c) correspondantes

IV.3 Commande prédictive pour l'exploration autonome

La commande prédictive est une stratégie de commande sur horizon fini qui cherche la commande optimale à appliquer au système à partir de la prédiction de son comportement futur. Cette stratégie de commande est régulièrement utilisée pour la navigation de véhicules autonomes dans des environnements complexes car elle permet de répondre à des objectifs multi-critères et de respecter des contraintes diverses. Elle permet également la commande des systèmes non-linéaires. Des détails sur cette commande sont disponibles dans la section III.2.2.3.

Pour prédire le comportement futur du système, il est nécessaire de connaître le modèle cinématique du robot. Ce modèle est présenté en section IV.3.1. Le principe de la commande prédictive est expliqué en section IV.3.2. Une fonction de coût est formulée pour traduire les objectifs de la mission, cette fonction est décrite en section IV.3.3. L'optimisation de la fonction de coût permet d'obtenir la commande optimale à appliquer au système, elle est expliquée en section IV.3.4.

IV.3.1 Modèle cinématique du robot

Cette partie décrit le modèle cinématique du robot qui permet de relier sa position et son orientation en fonction de la commande en vitesse linéaire et angulaire.

Le vecteur d'état est noté $X = (x, y, \theta)^T$ avec (x, y), la position du robot sur le plan du sol et θ, l'orientation du robot par rapport à l'axe x. Le vecteur de commande est noté $U = (v, \omega)^T$ avec v, la vitesse linéaire et ω, la vitesse angulaire, comme illustré dans la figure IV.4.

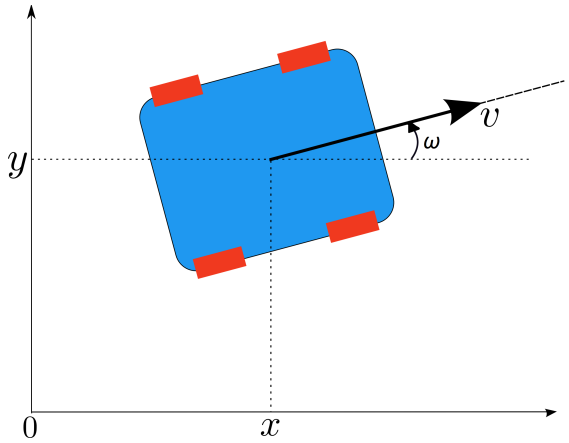


FIGURE IV.4 – Schéma représentant les paramètres d'état et de commande du robot

Le modèle cinématique en temps continu du robot s'écrit [LaValle, 2006] :

$$\begin{cases} \dot{x} = v \cos\theta \\ \dot{y} = v \sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{IV.2}$$

Le modèle cinématique discrétisé du système est noté :

$$X_n = f\left(X_{n-1}, U_{n-1}\right) \tag{IV.3}$$

qui peut s'écrire, avec une discrétisation d'Euler, comme :

$$\begin{cases} x_n = x_{n-1} + t_e v_{n-1} \cos\theta_{n-1} \\ y_n = y_{n-1} + t_e v_{n-1} \sin\theta_{n-1} \\ \theta_n = \theta_{n-1} + t_e \omega_{n-1} \end{cases} \tag{IV.4}$$

avec t_e, la période d'échantillonnage.

IV.3.2 Principe de la commande prédictive

Le principe de la commande prédictive (MPC : Model Predictive Control) est de prédire des trajectoires à partir du modèle cinématique du système sur un horizon de

temps fini. Pour chaque trajectoire, un coût représentant les objectifs de la mission est calculé. La commande correspondant à la trajectoire présentant le coût le plus faible est appliquée. Cette opération est réalisée à intervalles réguliers en prenant en compte les nouvelles informations acquises par le système.

En pratique, deux horizons H_p et H_c sont définis. H_c est l'horizon de commande sur lequel la séquence de commandes est définie. H_p est l'horizon de prédiction sur lequel les trajectoires sont prédites. H_c est toujours inférieur ou égal à H_p. A l'instant actuel t_n, la séquence de commande est notée $\mathcal{U}_n$ et la séquence des états, correspondants à la trajectoire prédite, est notée $\mathcal{X}_n$. Cette séquence s'obtient à partir de la séquence $\mathcal{U}_n$ et en utilisant l'équation du modèle du robot (IV.4).

$$\mathcal{U}_n = \{U_n, U_{n+1}, \ldots, U_{n+H_c-1}\} \tag{IV.5}$$

$$\mathcal{X}_n = \{X_{n+1}, X_{n+2}, \ldots, X_{n+H_p}\} \tag{IV.6}$$

Les vecteurs de commandes U_k dans la séquence $\mathcal{U}_n$ sont définis par :

$$U_k = \begin{cases} (v_k, \omega_k)^T & \text{si} \quad k \in [0, H_c - 1] \\ (v_{H_c-1}, 0)^T & \text{si} \quad k \in [H_c, H_p] \end{cases} \tag{IV.7}$$

Les valeurs des vitesses linéaire et angulaire sont bornées par $(v_{\min}, \omega_{\min})^T$ et $(v_{\max}, \omega_{\max})^T$.

Une fonction de coût $J(\mathcal{U}_n, \mathcal{X}_n)$ est définie, elle représente mathématiquement les objectifs de la mission. Cette fonction est minimisée afin d'obtenir la séquence de commande optimale $\mathcal{U}_n^*$:

$$\begin{gathered} \mathcal{U}_n^* = \arg \min_{\mathcal{U}_n \in \mathbb{U}_n} J(\mathcal{U}_n, \mathcal{X}_n) \\ \text{avec } X_k \in \mathcal{X}_n \text{ satisfaisant (IV.4)}, \\ \forall k \in [n+1, n+H_p] \end{gathered} \tag{IV.8}$$

$\mathbb{U}_n$ est l'espace de recherche des commandes.

La première composante U_n^* de la séquence optimale $\mathcal{U}_n^*$ est appliquée. Le processus est répété au pas de temps suivant, en prenant en compte les nouvelles informations acquises entre-temps.

IV.3.3 Fonction de coût

La fonction de coût est une représentation mathématique des différents objectifs de la mission. Elle est formulée comme une somme pondérée de coûts, correspondant chacun à un objectif distinct de la mission.

Chaque coût est formulé de telle sorte qu'il soit minimal pour la trajectoire optimale selon l'objectif du coût. Cependant, les différents objectifs peuvent être antagonistes. La trajectoire optimale selon la fonction de coût n'est pas forcément optimale pour tous les objectifs qui la composent. Il existe différentes solutions pour réaliser une optimisation multi-objectifs, nous avons choisi d'utiliser une somme pondérée des différentes objectifs. On pourrait utiliser des techniques de Pareto, des revues sur l'optimisation multi-objectifs sont disponibles dans [Andersson, 2000, Marler and Arora, 2004].

Dans cette thèse, différents objectifs sont considérés dans les missions du robot : exploration, évitement d'obstacles, ralliement de points de passage, etc... Ils ont été traduits en coûts :

— Le coût d'évitement d'obstacles J_{obs}, présenté en section IV.3.3.3.

— Le coût d'exploration J_{expl} cherche à attirer le robot vers les zones inconnues, il est décrit dans la section IV.3.3.4.

— Le coût de ralliement de points de passage J_{wp} permet au robot de rallier un point défini sur la carte, voir en section IV.3.3.2.

— Le coût de régulation de vitesse J_{u} permet de réguler les vitesses autour de valeurs nominales, il est expliqué en section IV.3.3.1.

Tous les coûts $J_\bullet$ sont normalisés entre 0 et 1.

La formulation de la fonction de coût pour la mission considérée dans ce chapitre, qui consiste à explorer l'environnement et éviter les obstacles est :

$$J = w_{\text{expl}} J_{\text{expl}} + w_{\text{obs}} J_{\text{obs}} + w_{\text{u}} J_{\text{u}} \tag{IV.9}$$

Les pondérations $w_\bullet$ permettent de régler l'importance de chaque coût par rapport aux autres. Le comportement du robot dépend de la valeur de l'ensemble des pondérations. Si la valeur d'une pondération est élevée par rapport aux autres pondérations, l'objectif associé à ce coût va être prépondérant par rapport aux autres objectifs.

Le réglage des pondérations n'est pas une tâche aisée. Il est effectué lors des expérimentations en testant de multiples jeux de valeurs et en gardant celui qui s'approche au mieux du comportement désiré. Mais il est difficile d'obtenir un réglage optimal et adapté à toutes les situations. En effet, le comportement du robot peut dépendre de la situation dans laquelle il se trouve. Un jeu de valeurs peut permettre le succès d'une mission d'exploration dans un premier environnement et se solder par un échec dans un autre environnement.

Les sections suivantes décrivent la formulation mathématique des quatre coûts présentés ci-dessus.

IV.3.3.1 Coût de régulation de vitesse

Le coût de régulation de vitesse J_{u} permet de réguler la vitesse linéaire autour d'une vitesse nominale notée v_0 et de pénaliser les trajectoires avec une vitesse de rotation importante.

$$J_{\text{u}} = \frac{1}{2H_c} \sum_{k=n}^{H_{\text{c}}-1} \left(\frac{\omega_k^2}{\omega_{\max}^2} + \frac{w_{\text{v}} \left(v_k - v_0\right)^2}{\left(\|v_0\| + \|v_{\max}\|\right)^2} \right) \tag{IV.10}$$

Le facteur w_{v} permet de pondérer l'importance de la régulation de vitesse linéaire par rapport à la régulation de la vitesse angulaire. Pour que le coût J_{u} reste compris entre -1 et 1, il faut que w_{v} soit inférieur à 1.

IV.3.3.2 Coût de ralliement de points de passage

Le but du coût de ralliement de points de passage est de favoriser les trajectoires qui s'approchent du point à rallier, noté X_w. Ce coût est formulé ainsi :

$$J_{\text{wp}} = \frac{1}{H_p.d_n} \sum_{k=n+1}^{n+H_p} \|X_w - \widehat{X_k}\|^2 \tag{IV.11}$$

avec $d_n = \|X_w - X_n\|^2$, la distance entre la position actuelle du robot et le point à rallier X_w.

Ce coût consiste à additionner la distance de chaque position de la trajectoire prédite avec le point à rallier. Le coût est minimal pour la trajectoire dont l'ensemble des positions s'approchent au plus du point à rallier.

Quand la trajectoire prédite est longue (ie. l'horizon H_p est grand), cette formulation du coût peut empêcher le robot d'atteindre le point X_w quand il se trouve très proche de celui-ci. En effet, minimiser la somme des distances au point dans ce cas peut défavoriser les trajectoires prédites qui traversent le point X_w, mais dont les dernières positions prédites s'en éloignent. Comme toutes les positions prédites ont la même influence, le coût peut alors devenir important même si le point X_w est atteint. Dans ce cas, la trajectoire optimale peut être une trajectoire qui n'atteint pas le point X_w mais dont toutes les positions prédites restent plutôt proches, par exemple une trajectoire qui effectue une rotation importante.

Pour pallier ce problème, on peut rajouter dans la formulation du coût un facteur qui diminue l'influence d'une position prédite dans le coût en fonction de son pas. Par exemple, on peut utiliser la fonction :

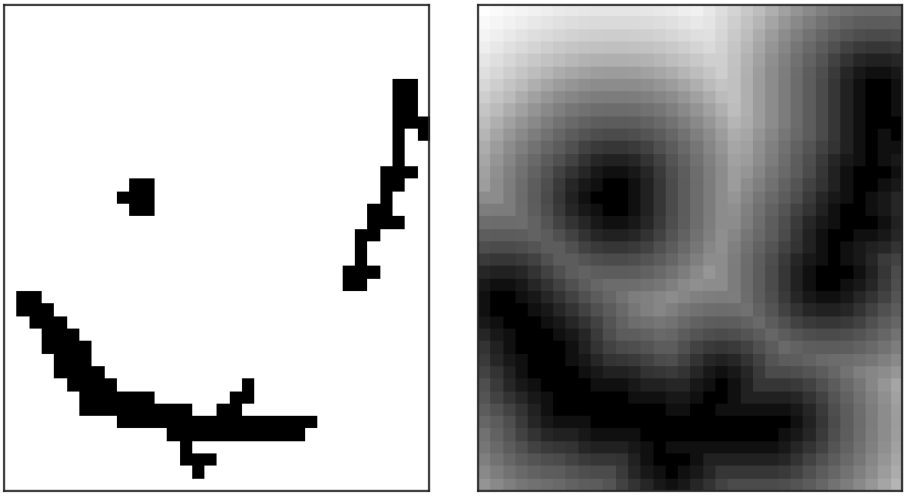
$$J_{\text{wp}} = \frac{1}{H_p.d_n} \sum_{k=n+1}^{n+H_p} \frac{\|X_w - \widehat{X_k}\|^2}{k} \tag{IV.12}$$

IV.3.3.3 Coût d'obstacle

Le coût d'obstacle consiste à pénaliser les trajectoires qui intersectent les obstacles ou qui s'en approchent trop.

A partir de la carte d'obstacle envoyée par le module de reconstruction de l'environnement (voir section IV.2), une carte de distance est calculée. Chaque pixel de la carte de distance contient la distance avec l'obstacle le plus proche, voir la figure IV.5. Cette opération est réalisée en appliquant une transformée de distance euclidienne. On utilise la fonction *distance_transform_edt* du package python scipy.ndimage pour réaliser cette opération.

Ensuite, une fonction de pénalité est appliquée pour toutes les positions de la trajectoire prédite. Cette fonction a pour but de définir une pénalité comprise entre 0 et 1, en fonction de la distance à l'obstacle, en ne considérant que les obstacles proches



(a) Carte d'obstacle (b) Carte de distance

FIGURE IV.5 – La carte d'obstacle (a) et la carte de distance correspondante (b)

[Rochefort et al., 2014]. L'équation de cette fonction s'écrit :

$$f_o\left(d_{\text{obs}}(X_k)\right) = \frac{1 - \tanh\left(a\left(d_{\text{obs}}\left(X_k\right) - b\right)\right)}{2} \tag{IV.13}$$

$$a = \frac{6}{d_{\text{des}} - d_{\text{sec}}} \tag{IV.14}$$

$$b = \frac{1}{2}\left(d_{\text{des}} + d_{\text{sec}}\right) \tag{IV.15}$$

avec

- $d_{\text{obs}}\left(X_k\right)$, la distance entre la position du robot à l'instant t_k et l'obstacle le plus proche. Cette distance est donnée par la carte de distance.
- d_{sec}, la distance de sécurité, en-dessous de laquelle la prise en compte de l'obstacle est maximale
- d_{des}, la distance désirée, au-dessus de laquelle les obstacles ne sont plus considérés.

Le tracé de cette fonction est affiché dans la figure IV.6. La fonction est maximale si $d_{\text{obs}}\left(X_k\right) < d_{\text{sec}}$ et nulle si $d_{\text{obs}}\left(X_k\right) > d_{\text{des}}$ et décroît de manière continue entre ces deux valeurs.

La formulation du coût est la somme des pénalités associées à chaque position de la trajectoire prédite :

$$J_{\text{obs}} = \frac{1}{H_p}\sum_{k=n+1}^{n+H_p} f_{\text{obs}}\left(d_{\text{obs}}(\widehat{X_k})\right). \tag{IV.16}$$

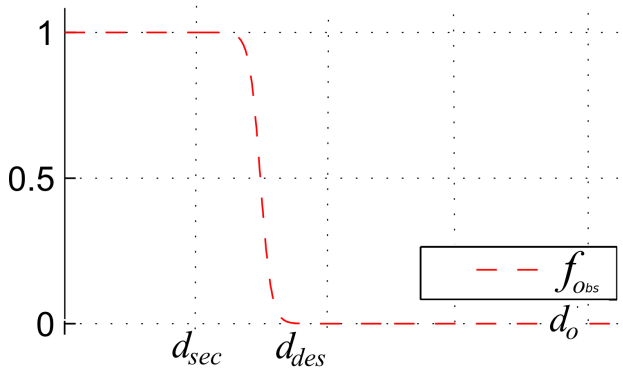


FIGURE IV.6 – Fonction de pénalité

IV.3.3.4 Coût d'exploration

Le coût d'exploration consiste à préférer les trajectoires qui vont explorer des zones encore non visitées. Le but est de maximiser l'aire de la zone observée future.

$G(n)$ est la carte d'exploration actuelle, obtenue par le module de reconstruction de l'environnement (voir section IV.2). $G_{i,j}(n)$ est la valeur de l'élément de coordonnées (i, j) dans la matrice $G(n)$. On définit une nouvelle carte $G(n+H_p)$, carte d'exploration prédite à la fin de l'horizon de prédiction H_p. Cette carte est initialisée avec $G(n)$ et on y ajoute la zone qui est observée sur la trajectoire prédite. Ces cartes sont discrètes, avec une résolution notée r.

Pour définir la zone qui sera explorée sur la trajectoire prédite, il faut définir la zone observée pour chaque position de la trajectoire prédite.

Il faut dans un premier temps définir la zone couverte par le capteur utilisé pour la reconstruction de l'environnement. Les capteurs de profondeur utilisés, présentés dans la section II.2.1.2, ont un champ de vue représenté sous la forme d'un secteur d'angle centré sur l'orientation du robot, défini par l'angle d'ouverture β et la portée du capteur δ. La figure IV.7 montre l'évolution de la zone explorée le long de la trajectoire parcourue.

Pour simplifier les calculs, le secteur d'angle est approximé par un triangle isocèle centré sur l'orientation du robot, voir la figure IV.8a. L'angle au sommet est égal à β et la longueur des cotés adjacents vaut δ. Pour tracer le triangle sur la carte, on calcule les coordonnées des trois sommets du triangle, puis les segments entre les sommets sont tracés en utilisant l'algorithme de Bresenham [Bresenham, 1965].

Le contour du champ de vue est tracé sur la carte d'exploration prédite $G(n + H_p)$ pour toutes les positions de la trajectoire prédite, comme illustré dans les figures (figures IV.8b et IV.8d). Une opération de fermeture est réalisée afin de compléter les zones intérieures non dessinées, voir figures IV.8c et IV.8e.

On obtient donc une carte prédite de l'exploration $G(n+H_p)$ pour chaque trajectoire prédite. Le principe du coût est de maximiser l'aire de la zone qui sera explorée à la fin de la trajectoire prédite, en calculant la différence entre la carte d'exploration courante $G(n)$ et celle prédite $G(n + H_p)$. La formulation du coût est :

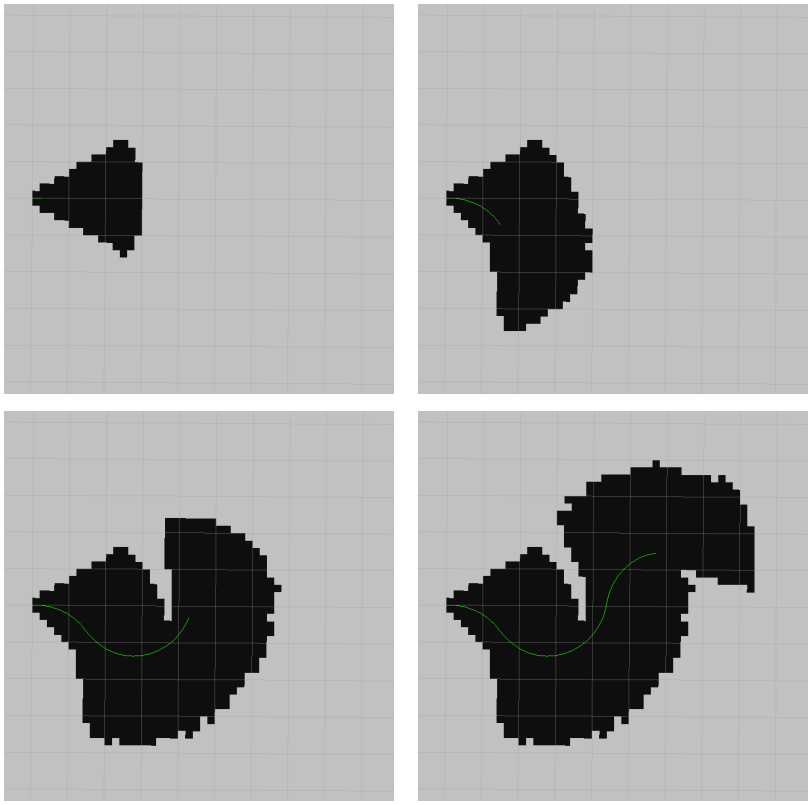


FIGURE IV.7 – Zone explorée (en noir) le long de la trajectoire (en vert)

$$J_{\text{expl}} = \frac{r^2}{H_p.\beta.\delta^2} \sum_i \sum_j G_{i,j}(n) - G_{i,j}(n + H_p) \tag{IV.17}$$

Ce coût est exprimé entre 0 et -1 pour que l'exploration soit traduite comme un coût à minimiser. Il prend la valeur 0 si l'exploration est nulle et est minimisé par la trajectoire qui découvre la plus grande surface non explorée.

IV.3.4 Optimisation de la fonction de coût

La fonction de coût J doit être optimisée afin d'obtenir la commande optimale à appliquer. La fonction qui a été définie dans la partie précédente est non-linéaire et non-convexe, ce qui nécessite d'utiliser un algorithme d'optimisation global.

Pour pouvoir commander le robot en temps-réel, l'algorithme d'optimisation doit donner une solution avec un temps inférieur à la période d'échantillonnage t_e.

Il existe différentes solutions pour effectuer cette optimisation.

Si le temps d'exécution d'une évaluation de la fonction de coût est faible, il est possible d'utiliser l'algorithme DIRECT [Jones et al., 1993]. C'est un algorithme d'optimisation

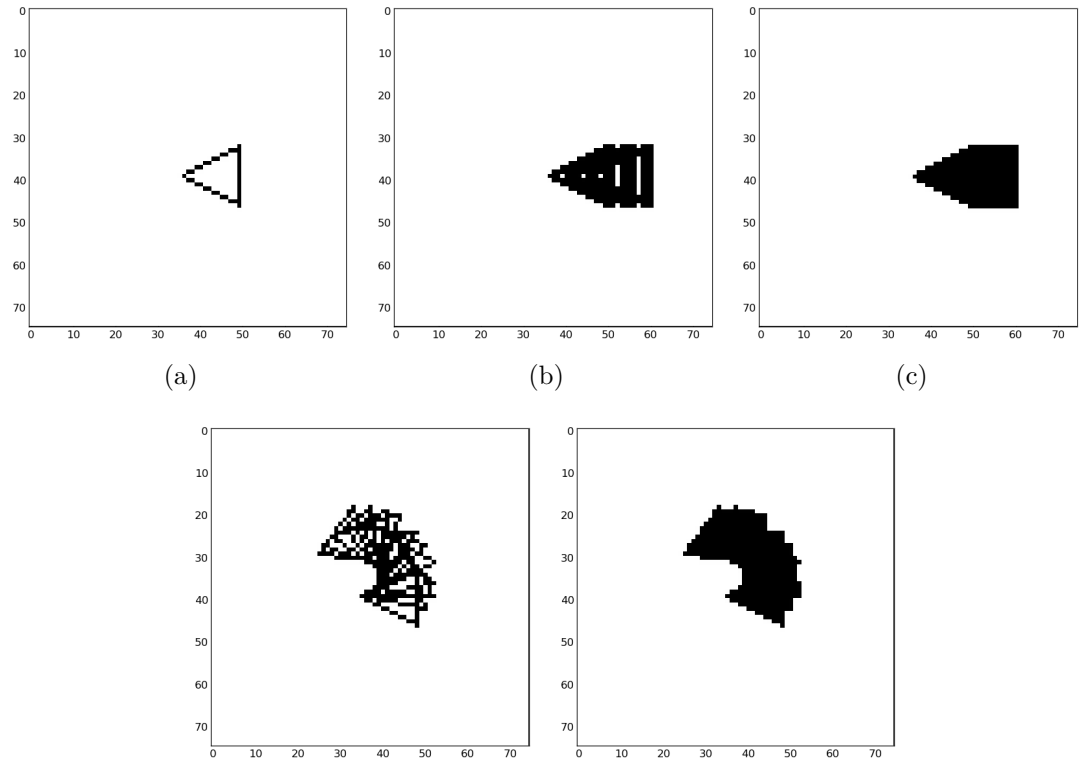


FIGURE IV.8 – Cartes d'exploraiton prédites : forme du champ de vue du capteur (a), prédiction dans le cas d'une translation pure (b),(c), prédiction dans le cas d'un mouvement de rotation et translation (d),(e).

globale qui peut traiter des fonctions non-linéaires et non-convexes comme la fonction de coût décrite dans les parties précédentes. Cependant, le temps d'exécution n'est pas constant.

Dans le cas contraire, une solution possible est de définir un ensemble fixe de séquences de commandes à tester. Pour chaque commande de l'ensemble, le coût est calculé. La commande avec le coût le plus faible est appliquée. Cette méthode a l'avantage de présenter un temps de calcul constant et prévisible mais ne permet pas d'obtenir la solution optimale sur tout l'espace des commandes.

IV.4 Fusion de données multi-capteurs

Pour réaliser sa mission, il est nécessaire que le robot se localise de façon précise. Il doit savoir où il est situé sur la carte de l'environnement afin d'éviter les obstacles et de calculer des trajectoires sûres. Une localisation précise est également indispensable pour reconstruire la carte de l'environnement. En effet, les obstacles détectés sont connus

dans le référentiel du robot. Ils peuvent être insérés dans une carte globale à condition de connaître la position du robot dans cette carte au moment de la détection. Plus la localisation est précise et plus la carte globale reconstruite sera conforme au plan de la pièce.

Or, l'odométrie visuelle nécessite de percevoir une scène suffisamment riche en texture pour pouvoir calculer une localisation précise. Quand la scène est trop peu texturée, l'extraction de points d'intérêts dans les images est difficile. L'algorithme d'odométrie visuelle peut alors calculer des valeurs de localisation aberrantes, ce qui peut s'avérer dangereux pour le robot et nécessiter une reprise du contrôle par l'opérateur. La figure IV.9 montre une situation dans laquelle le calcul de la localisation visuelle est défaillant. On peut remarquer que l'algorithme a calculé une pose qui se trouve derrière le mur. Le module de reconstruction de l'environnement a pris en compte cette mauvaise valeur et un nouveau mur est apparu derrière le premier mur. Dans ce cas, le robot est perdu et la carte de l'environnement obtenue est mauvaise, ce qui conduit à l'échec de la mission d'exploration.

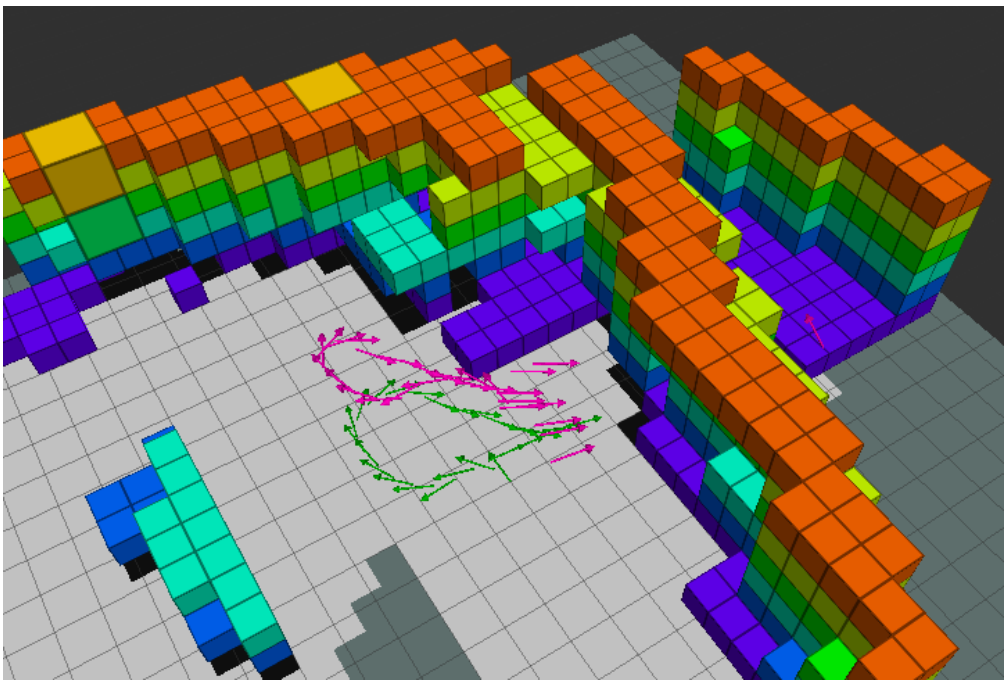


FIGURE IV.9 – Erreur dans le calcul de la localisation visuelle. Les flèches vertes représentent l'odométrie des roues et les flèches violettes l'odométrie visuelle, les cubes colorés sont les obstacles détectés par le module de reconstruction de l'environnement.

Comme l'environnement est inconnu et peut potentiellement présenter des zones difficiles pour la localisation visuelle, utiliser uniquement l'odométrie visuelle pour estimer la localisation du robot nécessite de suivre des trajectoires qui évitent ces zones peu texturées. Cette solution sera étudiée dans le chapitre V. Ici, comme d'autres capteurs

de localisation sont embarqués sur le robot, la solution proposée pour assurer un calcul de localisation robuste est de réaliser une fusion de données multi-capteurs.

Dans notre cas, on dispose de l'odométrie des roues et de la centrale inertielle. Sur le robot utilisé pour les expériences, la vitesse linéaire calculée par l'odométrie des roues est assez précise, par contre la vitesse angulaire estimée ne l'est pas. Les gyromètres de la centrale inertielle donnent une mesure précise de la vitesse de rotation de l'engin. On a donc choisi de prendre la valeur de la vitesse linéaire calculée à partir de l'odométrie des roues et la valeur de vitesse angulaire donnée par la centrale inertielle pour réaliser la fusion de données.

Un filtre de Kalman étendu (EKF) est mis en place pour effectuer la fusion de données entre ces différents capteurs et donner une estimation de la localisation, voir en section IV.4.1. Il permet également de détecter les valeurs aberrantes dans la localisation visuelle et de les corriger, voir en section IV.4.2.

IV.4.1 Filtre de Kalman étendu

Le filtre de Kalman étendu est une extension du filtre de Kalman pour les systèmes non-linéaires. Le principe général est de linéariser la fonction non-linéaire autour de l'estimée courante.

L'état estimé est noté $\widehat{X}$ et sa matrice de covariance associée est notée P. On note $\widehat{U} = [\widehat{v}, \widehat{\omega}]^{\mathrm{T}}$ le vecteur de commande mesuré, provenant de l'odométrie des roues et de la centrale inertielle. Les variances des bruits de mesure sur la vitesse linéaire v et la vitesse angulaire ω sont notées respectivement σ_v^2 and σ_ω^2. La matrice de covariance du bruit d'état est :

$$Q_n = \begin{bmatrix} \sigma_v^2(n) & 0 \\ 0 & \sigma_\omega^2(n) \end{bmatrix} \tag{IV.18}$$

Le processus se déroule en deux étapes : une étape de prédiction et une étape de mise à jour.

IV.4.1.1 Prédiction

Le principe de l'étape de prédiction est d'utiliser l'état estimé de l'instant précédent $\widehat{X}_{n-1|n-1}$ et le vecteur de commande $\widehat{U}_{n-1}$ pour obtenir une estimation de l'état courant $\widehat{X}_{n|n-1}$.

L'état prédit est :

$$\widehat{X}_{n|n-1} = f\left(\widehat{X}_{n-1|n-1}, \widehat{U}_{n-1}\right) \tag{IV.19}$$

avec f, le modèle cinématique du système, voir équation (IV.4).

L'estimation prédite de la covariance est donnée par :

$$P_{n|n-1} = F_{n-1} P_{n-1|n-1} F_{n-1}^{\mathrm{T}} + G_{n-1} Q_{n-1} G_{n-1}^{\mathrm{T}} \tag{IV.20}$$

où F_{n-1} et G_{n-1} sont les matrices jacobiennes de f par rapport à X et U respectivement :

$$F_{n-1} = \left.\frac{\partial f\left(X,U\right)}{\partial X}\right|_{\widehat{X}_{n-1|n-1},\widehat{U}_{n-1}} \tag{IV.21}$$

$$G_{n-1} = \left.\frac{\partial f\left(X,U\right)}{\partial U}\right|_{\widehat{X}_{n-1|n-1},\widehat{U}_{n-1}} \tag{IV.22}$$

Ici, d'après notre modèle (IV.4), ces matrices sont égales à :

$$F_{n-1} = \begin{bmatrix} 1 & 0 & -t_e\widehat{v}_{n-1}\sin\widehat{\theta}_{n-1} \\ 0 & 1 & t_e\widehat{v}_{n-1}\cos\widehat{\theta}_{n-1} \\ 0 & 0 & 1 \end{bmatrix} \tag{IV.23}$$

$$G_{n-1} = \begin{bmatrix} t_e\cos\widehat{\theta}_{n-1} & 0 \\ t_e\sin\widehat{\theta}_{n-1} & 0 \\ 0 & t_e \end{bmatrix} \tag{IV.24}$$

IV.4.1.2 Mise à jour

L'étape de mise à jour du filtre utilise les mesures de l'odométrie visuelle $X_n^{\text{evo}} = [x_n^{\text{evo}}, y_n^{\text{evo}}, \theta_n^{\text{evo}}]^{\text{T}}$ pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

L'innovation est égale à :

$$\widetilde{Y}_n = X_n^{\text{evo}} - \widehat{X}_{n|n-1} \tag{IV.25}$$

et sa covariance est :

$$S_n = P_{n|n-1} + R_n \tag{IV.26}$$

où R_n est la matrice de covariance sur les bruits de mesure de l'odométrie visuelle

$$R_n = \begin{bmatrix} \sigma_x^2(n) & 0 & 0 \\ 0 & \sigma_y^2(n) & 0 \\ 0 & 0 & \sigma_\theta^2(n) \end{bmatrix} \tag{IV.27}$$

Le gain de Kalman optimal est :

$$K_n = P_{n|n-1}S_n^{-1} \tag{IV.28}$$

Finalement, l'état mis à jour et sa covariance sont :

$$\widehat{X}_{n|n} = \widehat{X}_{n|n-1} + K_n\widetilde{Y}_n \tag{IV.29}$$

$$P_{n|n} = \left(I_3 - K_n\right)P_{n|n-1} \tag{IV.30}$$

IV.4.2 Détection des mesures visuelles aberrantes

La détection des valeurs aberrantes dans les données de l'odométrie visuelle est réalisée en utilisant la valeur de l'innovation. Si la norme de l'innovation dépasse un certain seuil ϵ, on considère que la valeur de l'odométrie visuelle est erronée. On fixe alors la valeur de l'état estimé à celle de l'état prédit, soit $\widehat{X}_{n|n} = \widehat{X}_{n|n-1}$ et l'étape de mise à jour n'est pas réalisée [Chen and Patton, 2012]. L'algorithme d'odométrie visuelle est réinitialisé avec cette même valeur.

L'algorithme 1 décrit le processus de filtrage et de correction des données aberrantes.

Algorithm 1 Algorithme EKF

A chaque pas de temps t_n

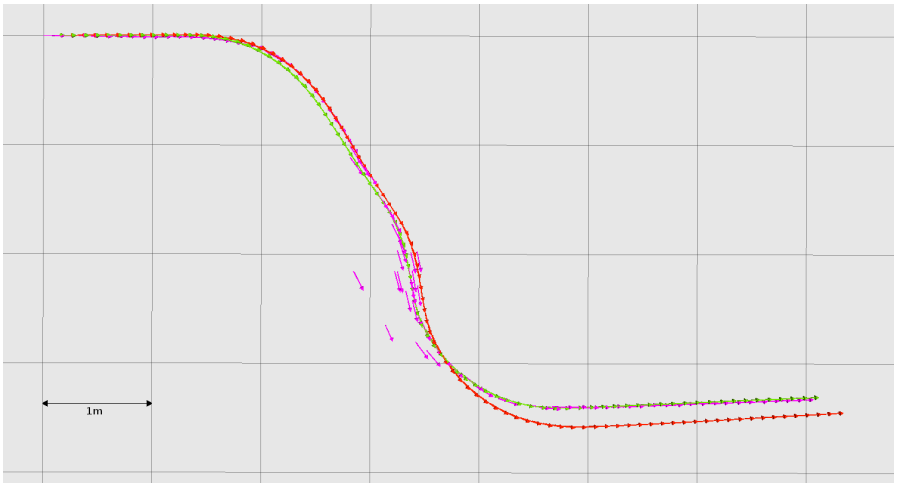
1. Calculer $\widehat{U}_{n-1}$ a partir des données de l'odométrie des roues et de l'IMU
2. Calculer la prédiction $\widehat{X}_{n|n-1}$ et $P_{n|n-1}$ en utilisant (IV.19) et (IV.18)
3. Utiliser les mesures d'odométrie visuelle X_n^{evo} pour calculer l'innovation $\widetilde{Y}_n$ en utilisant (IV.25) et (IV.26)
4. **Si** $\|\widetilde{Y}_n\|^2 > \epsilon$
   Utiliser $\widehat{X}_{n|n-1}$ comme état estimé et réinitialiser eVO avec cette valeur
   Sinon
   Calculer la mise à jour $\widehat{X}_{n|n}$ et $P_{n|n}$ à partir de (IV.29) et (IV.30)
5. Recommencer en 1 avec $n \leftarrow n + 1$

La figure IV.10 montre une trajectoire sur laquelle le calcul de la localisation visuelle est incorrect (a) et l'évolution de la norme de l'innovation pendant le déplacement (b). La norme de l'innovation dépasse en plusieurs points le seuil fixé, ce qui correspond aux valeurs aberrantes de l'odométrie visuelle qu'on peut observer au milieu de la trajectoire sur la figure (a). Ces valeurs aberrantes ont été détectées par le filtre. La localisation estimée n'a donc pas été affectée par ces valeurs aberrantes. De plus, l'algorithme d'odométrie visuelle a été réinitialisé avec la valeur estimée du filtre, ce qui permet à l'algorithme de reprendre son calcul de la localisation normalement. On peut observer une légère dérive entre l'odométrie des roues et la valeur de localisation filtrée sur la fin de la trajectoire, ce qui est dû à la dérive normale des systèmes basés sur l'odométrie.

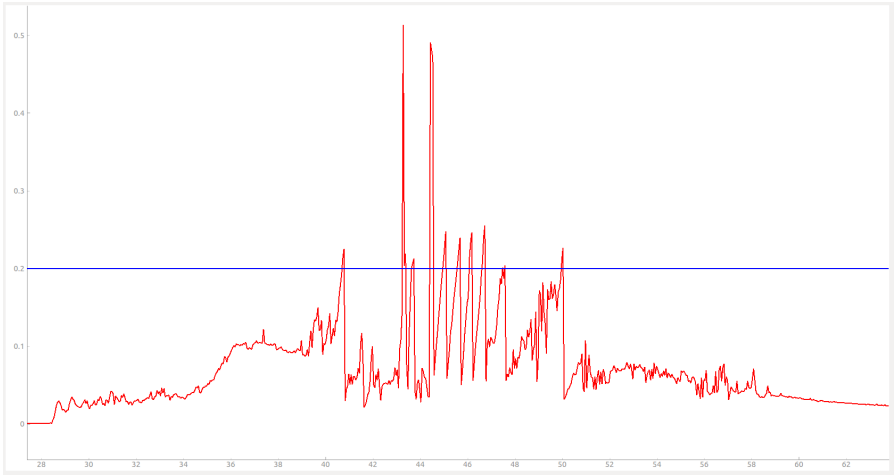
IV.5 Supervision de missions

Lors des premières expériences d'exploration autonome avec le robot, la fonction de coût utilisée prenait en compte le coût d'exploration, le coût d'obstacle et le coût de régulation de vitesse :

$$J = w_{\text{expl}} J_{\text{expl}} + w_{\text{obs}} J_{\text{obs}} + + w_{\text{u}} J_{\text{u}} \tag{IV.31}$$



(a) Odométries : roues (rouge), visuelle (violet) et après filtrage (vert)



(b) Evolution de la norme de l'innovation (rouge) durant le déplacement et seuil ϵ (bleu)

FIGURE IV.10 – Détection des erreurs de localisation visuelle

Des situations de blocage sont apparues lors du déroulement de la mission d'exploration du robot, ce qui a empêché d'achever correctement la mission. En effet, la commande prédictive est une stratégie de commande locale, seules les informations situées dans la zone atteignable sur l'horizon de prédiction sont prises en compte. Or, il arrive alors que le robot ne parvienne pas à détecter les zones non explorées car elles sont à l'extérieur de la zone atteignable. Il n'est donc pas attiré par ces zones non explorées et ne parvient pas à finaliser l'exploration de la pièce. Une stratégie de déblocage a été développée afin de résoudre cette situation, elle est présenté en section IV.5.1.

Cette stratégie s'appuie sur le ralliement de points de passage. Or, il arrive parfois que le robot ne parvienne pas à rejoindre le point qu'il doit rallier dans certaines situations

particulières. Pour remédier à cette situation, une trajectoire globale est calculée afin de guider le robot jusqu'au point de ralliement. Cette procédure est décrite en section IV.5.2.

IV.5.1 Le robot n'explore plus

La première situation de blocage apparaît lorsque le robot se trouve dans une large zone explorée dont les limites sont en dehors de la distance maximale atteignable sur l'horizon de prédiction. Dans ce cas, le coût d'exploration est nul pour toutes les trajectoires prédites. Le robot est alors guidé uniquement par les coûts d'obstacles et de régulation de vitesse. Si la vitesse nominale est non nulle, le robot va continuer à avancer et éviter les obstacles sans nécessairement s'approcher des zones non explorées. Dans ce cas, il ne peut pas finir sa mission d'exploration.

Une stratégie de déblocage a été développée afin d'attirer le robot vers les zones inexplorées en définissant des points à rallier sur la frontière de la zone explorée [Yamauchi, 1997]. Les points sont choisis de telle sorte qu'ils soient suffisamment éloignés d'obstacles, donc accessibles pour le robot et qu'ils soient à la frontière d'une zone inexplorée de taille importante. En effet, si une zone de quelques pixels est inconnue dans la carte, on considère que les pixels prennent la valeur des pixels voisins et il n'est donc pas nécessaire d'aller l'explorer.

Si plusieurs points correspondent à ces critères, le point le plus proche est choisi. Si aucun point n'est trouvé, cela signifie qu'il n'existe plus de zones accessibles à explorer dans la carte, on considère alors que la mission d'exploration est finie et on définit le point de départ comme point à rallier.

Le robot change alors de mode de navigation, il passe du mode exploration au mode ralliement de points de passage. La fonction de coût devient :

$$J = w_{\text{wp}} J_{\text{wp}} + w_{\text{obs}} J_{\text{obs}} + w_{\text{u}} J_{\text{u}} \tag{IV.32}$$

Le coût d'exploration n'apparaît plus dans cette nouvelle fonction. A la place, on intègre le coût de ralliement de points qui a pour but d'attirer le robot vers le point qui a été défini.

Le critère de déclenchement du mode ralliement de points est défini en fonction de l'évolution de la taille de la surface explorée. Si l'aire explorée n'augmente pas pendant une période de temps définie, on considère que le robot est bloqué et on change de mode de navigation.

Quand le point est atteint, le robot repasse en mode exploration. La fonction de coût dans le MPC reprend sa formulation initiale, intégrant le coût d'exploration.

Une situation de blocage et sa résolution sont visibles dans les résultats d'expériences, exposés en partie IV.6.

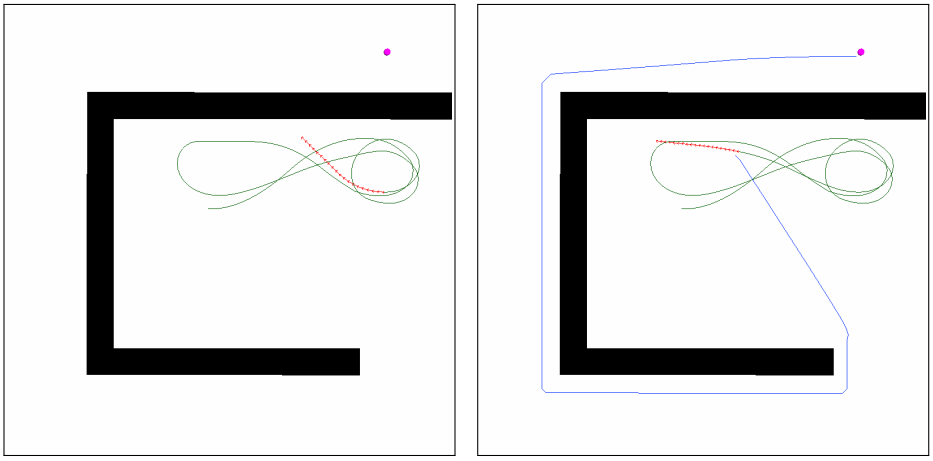
IV.5.2 Le robot ne parvient pas à rallier le point

Un deuxième type de blocage peut apparaître dans le mode de navigation par ralliement de points de passage. En effet, le coût de ralliement de points de passage est formulé de telle façon que le robot cherche systématiquement à se rapprocher du point. Or, en

présence d'obstacles, le robot doit parfois d'abord s'éloigner du point pour parvenir à atteindre le point, comme illustré dans la figure IV.11a.

Le principe de la procédure pour résoudre ce blocage est de chercher un chemin global entre le robot et le point à rallier puis de faire suivre ce chemin au robot afin qu'il atteigne le point qu'il doit rallier.

Un algorithme de planification globale, utilisant l'algorithme de Dijkstra, calcule un chemin entre la position actuelle du robot et le point à rallier. Ce chemin doit passer par des zones déjà explorées et éviter les obstacles, afin de guider le robot vers le point, voir la figure IV.11b.



(a) (b)

FIGURE IV.11 – Situation dans laquelle le robot ne peut pas atteindre le point (a) et calcul de la trajectoire globale (b). La trajectoire du robot est en vert, sa trajectoire prédite optimale en rouge, la trajectoire globale en bleu et le point à rallier est le point violet

Sur la trajectoire globale calculée, des points de passage sont définis périodiquement. Le robot rallie les points un par un pour finalement atteindre le point initial, comme illustré dans la figure IV.12. Une fois ce point atteint, le robot repasse en mode exploration.

La procédure est déclenchée quand la distance entre le point à rallier et le robot ne diminue pas pendant une période de temps définie.

L'algorithme 2 résume la procédure effectuée pour la gestion des blocages quand le robot n'explore plus et quand il ne parvient pas à rallier le point de passage.

IV.6 Expérimentations sur un robot terrestre

L'architecture présentée dans les sections précédentes a été développée et embarquée sur un robot terrestre afin de réaliser des expériences en situation réelle.

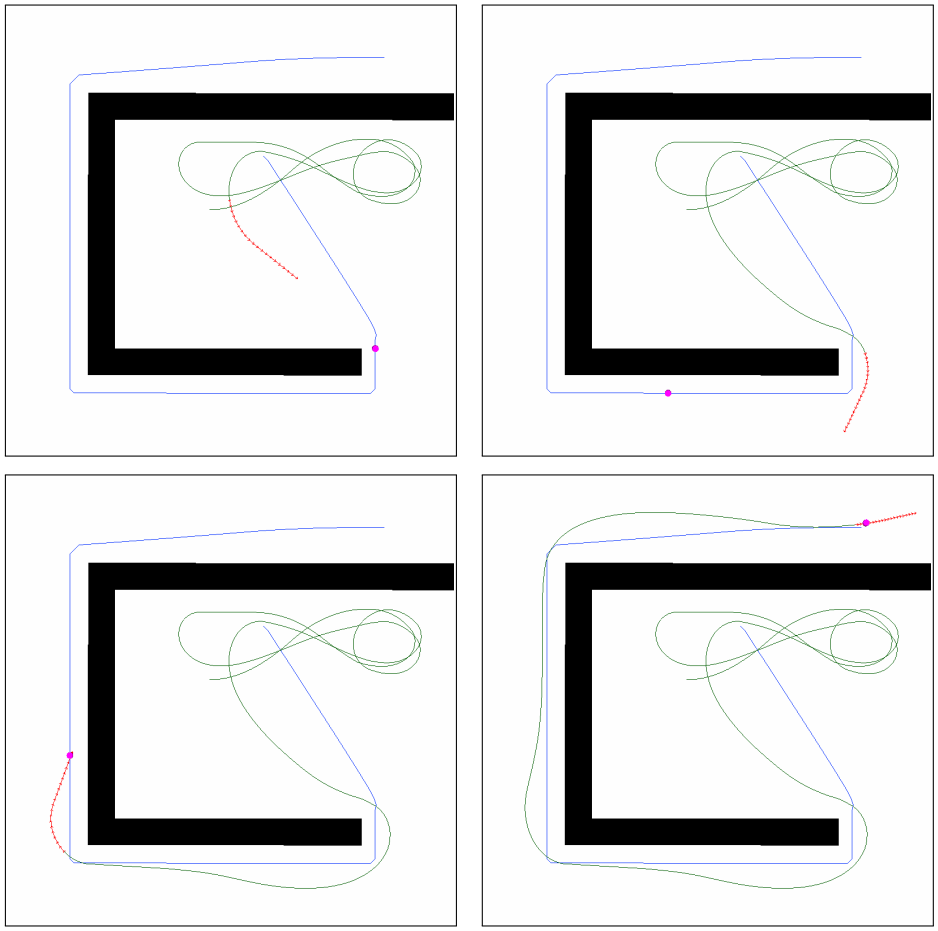


FIGURE IV.12 – Ralliement de points de passage intermédiaires pour atteindre le point initial

La section IV.6.1 décrit les conditions de l'expérience et le paramétrage du robot. La section IV.6.2 montre les résultats obtenus.

IV.6.1 Mise en place

Pour les expérimentations, le robot utilisé est le Summit XL, présenté en section II.2.2. La pièce à explorer est une partie du parking de l'ONERA, d'environ 20 m x 20 m. La figure IV.13 montre une photo de l'environnement avec le robot.

Les valeurs des paramètres sont :

— MPC

- Période d'échantillonnage $t_e = 0.25$ s
- Horizon de commande $H_c = 10$
- Horizon de prédiction $H_p = 20$

Algorithm 2 Gestion des situations de blocage du robot

```
Initialisation : mode := exploration
if mode == exploration then
    if le robot est bloqué then
        mode := ralliement de points
        Calcul d'un point à rallier
        if pas de point trouvé then
            Mission finie, retour au point de départ
        end if
    end if
else mode == ralliement de points
    if le point initial est atteint then
        mode := exploration
    end if
    if le robot est bloqué then
        Calcul d'une trajectoire globale et de la liste des points intermédiaires
        while la liste est non vide do
            Ralliement du premier point de la liste
            if un point intermédiaire est atteint then
                Le point est retiré de la liste
            end if
            if le dernier point de la liste est atteint then
                mode := exploration
            end if
        end while
    end if
end if
```

- Vitesse linéaire maximale $v_{\max} = 0.6$ m.s^{-1}
- Vitesse linéaire minimale $v_{\min} = 0.0$ m.s^{-1}
- Vitesse linéaire nominale $v_0 = 0.6$ m.s^{-1}
- Vitesses angulaires extrémales $\omega_{\max} = \omega_{\min} = 0.6$ rad.s^{-1}

L'horizon de prédiction est fixé plutôt long pour obtenir une différence importante sur le coût d'exploration entre les différentes trajectoires prédites et pour pouvoir détecter les zones inconnues. Les vitesses sont fixées selon les contraintes matérielles du robot. Avec des valeurs plus élevées, les vibrations dues au mouvement du robot sont importantes et risquent de provoquer un déplacement léger des caméras et de fausser la calibration du banc stéréo. La vitesse minimale est fixée à 0 m.s^{-1}, le robot ne peut pas reculer.

— Pondérations

- $w_{\text{wp}} = 15$

FIGURE IV.13 – Photo du robot en cours de mission d'exploration

- $w_{\text{obs}} = 40$
- $w_{\text{expl}} = 15$
- $w_{\text{u}} = 1$
- $w_{\text{v}} = 5$

Les pondérations ont été choisies en effectuant de nombreuses expériences et en modifiant petit à petit les valeurs jusqu'à obtenir le comportement désiré pour le robot. La pondération sur le coût d'obstacle est très élevée par rapport aux autres pondérations. Ce coût est nul en l'absence d'obstacle et n'a donc pas d'influence sur le résultat du coût total, par contre il devient prépondérant quand le robot est proche d'obstacles par rapport aux autres coûts. La pondération sur le coût de régulation de vitesse w_{u} est très faible car on veut que le robot explore au maximum son environnement et donc qu'il ne soit pas trop contraint sur ses commandes.

— Carte
  - Résolution $r = 0.2$ m/pixel
  - Longueur $W = 20$ m
  - Hauteur $H = 20$ m

  La résolution est choisie de telle sorte que la reconstruction de l'environnement soit suffisamment précise pour l'évitement d'obstacles et suffisamment rapide pour une utilisation temps-réel. La longueur et la hauteur de la carte correspondent approximativement à la taille de la pièce à explorer.

— Distance aux obstacles
  - sécurité $d_{sec} = 0.8$ m
  - désirée $d_{des} = 1.2$ m

Les distances aux obstacles sont réglées en fonction de la taille du robot. On veut que la distance minimale entre le robot et les obstacles soit équivalent à sa taille.

— EKF

- Période d'échantillonnage $t_e = 0.05$ s
- Incertitudes
  $\sigma_v^2 = 10^{-2}$ m^2.s^{-2}
  $\sigma_\omega^2 = 10^{-2}$ rad^2.s^{-2}
  $\sigma_x^2 = \sigma_y^2 = 10^{-2}$ m^2
  $\sigma_\theta^2 = 10^{-3}$ rad^2

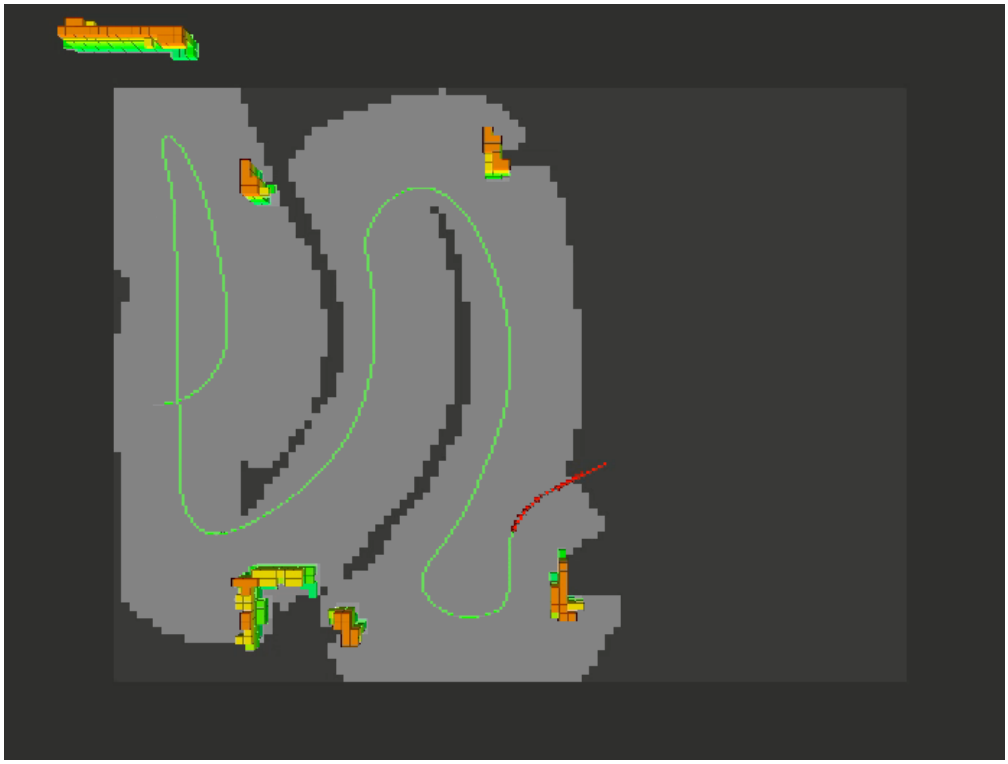
La période d'échantillonnage de l'EKF est la même que celle de la période de fonctionnement de l'algorithme d'odométrie visuelle, qui est de 20 Hz. On a choisi la période qui correspond au capteur le plus "lent". La centrale inertielle et les odomètres des roues publient les mesures avec une fréquence plus élevée, de 125 Hz et 50 Hz respectivement.

Les incertitudes sur les données des capteurs et sur les données envoyées par l'algorithme d'odométrie visuelle sont considérées fixes pendant la mission et ont été estimées au préalable.

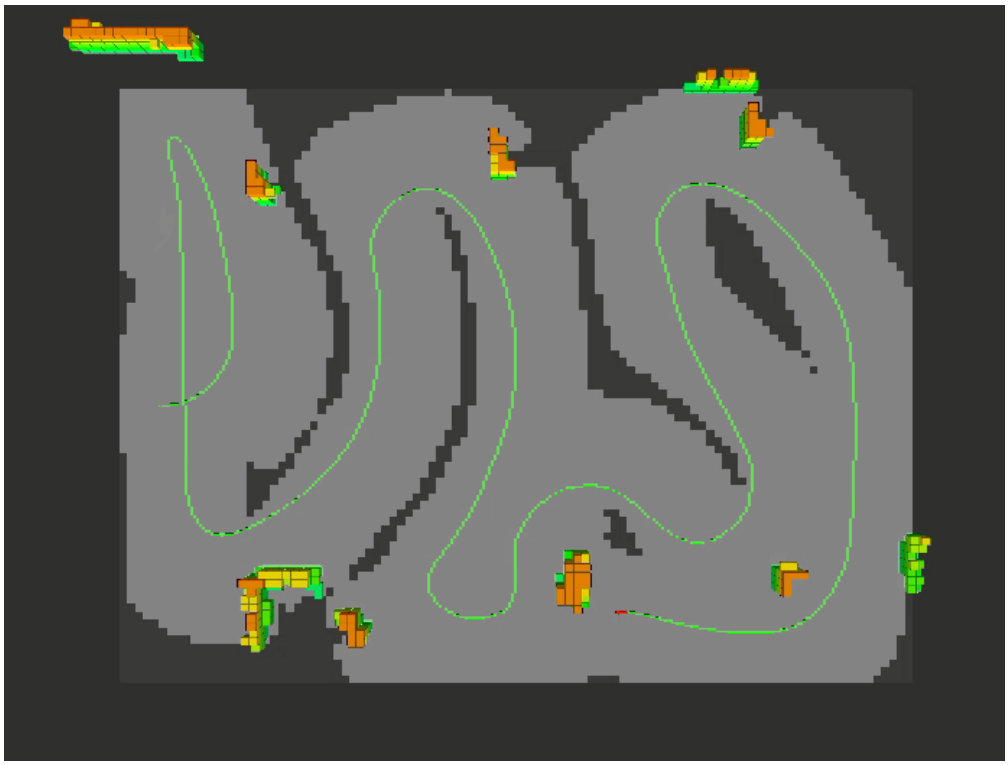
L'optimisation de la fonction de coût est réalisée par l'algorithme DIRECT. La même commande est définie sur tout l'horizon H_c, ce qui donne seulement deux variables à optimiser : v et ω. Cette simplification est effectuée afin d'obtenir un résultat d'optimisation en un temps limité (en moyenne 0.1 s) avec l'algorithme d'optimisation DIRECT (implémentation Python NLopt), ce qui est inférieur au pas de temps du MPC.

IV.6.2 Résultats obtenus

La figure IV.14 montre quatre images extraites d'une mission d'exploration. Sur ces images, la zone explorée est en gris clair et la zone non-explorée en gris foncé. La trajectoire du robot est en vert, la trajectoire prédite optimale en rouge. les obstacles sont les cubes colorés et les points intermédiaires en cas d'action de supervision sont en rose. La figure IV.14a montre le début de la mission. Le robot effectue une trajectoire qui serpente la pièce, ce qui permet une exploration exhaustive. Dans la figure IV.14b, le robot est bloqué dans un coin de la pièce, la commande optimale est la commande nulle. Comme l'aire explorée n'augmente pas, on se retrouve dans la première situation de blocage. Un point à rallier est défini et rejoint, comme le montre la figure IV.14c. Le robot va finalement parvenir à explorer toute la pièce (figure IV.14d), sa mission d'exploration est un succès, il peut ensuite revenir à son point de départ.

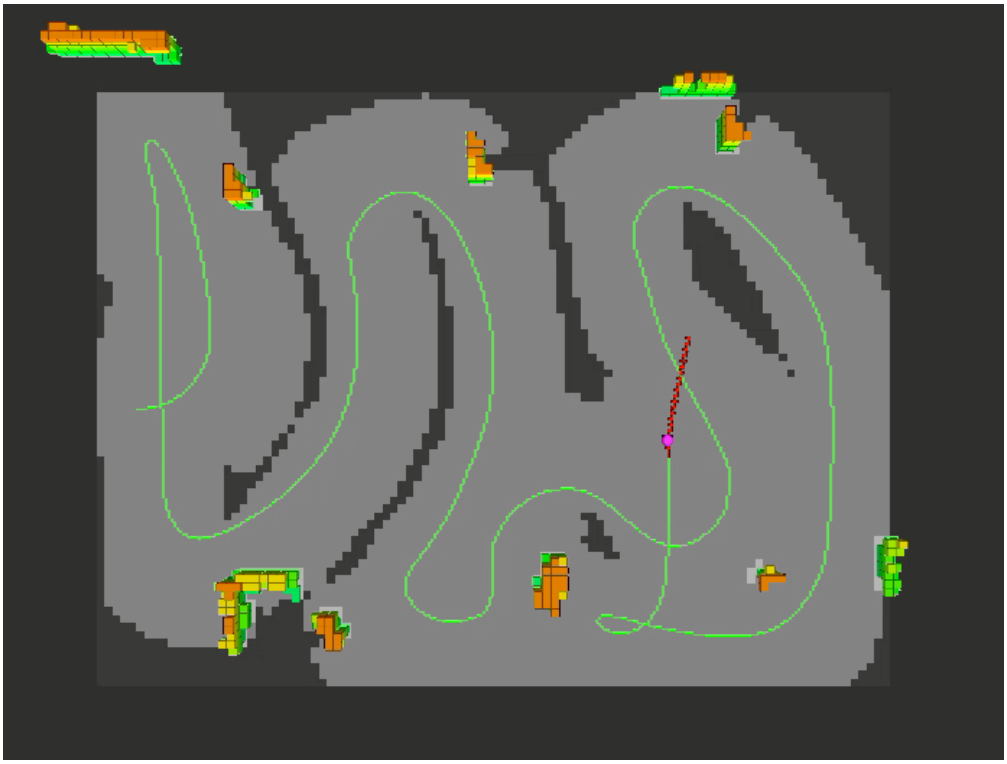


(a) Début de l'exploration

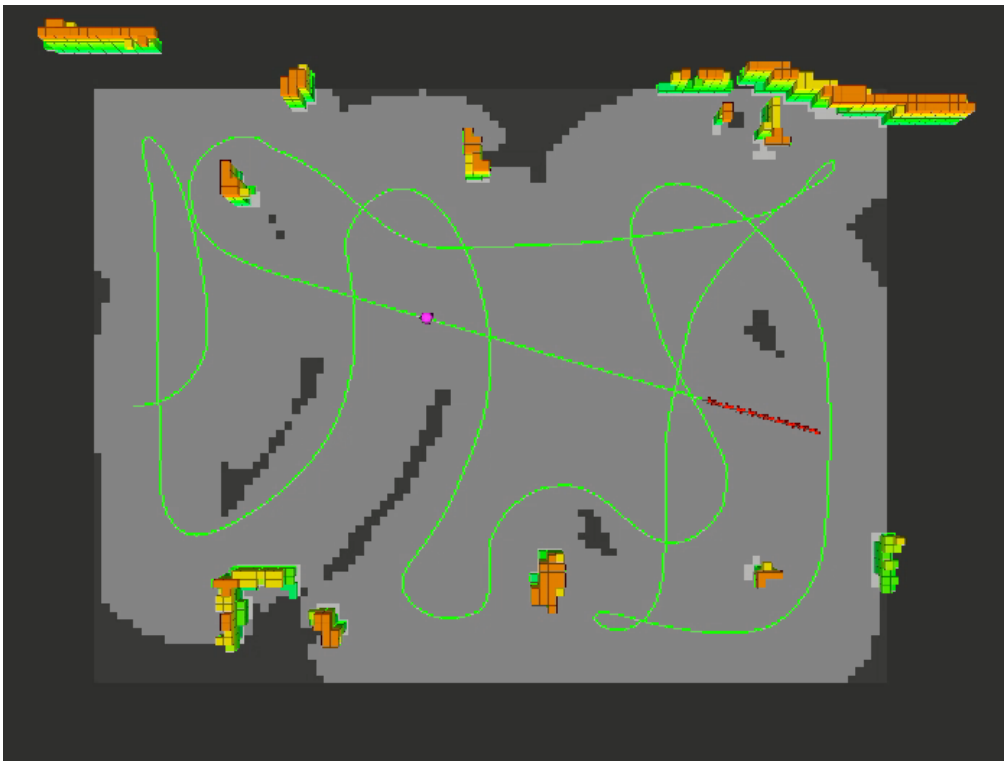


(b) Blocage du robot

FIGURE IV.14 – Images extraites d'une mission d'exploration



(c) Ralliement de points



(d) La pièce est explorée totalement

FIGURE IV.14 – Quatre images extraites à différentes étapes d'une mission d'exploration

Ces expériences montrent que l'architecture qui a été développée permet de remplir les objectifs demandés. Le robot explore toute la pièce et crée la carte du sol, en évitant les obstacles puis revient à son point de départ. Les procédures pour sortir des situations de blocage ont permis au robot de continuer sa mission jusqu'à son terme. La figure IV.15 montre la carte obtenue à la fin d'une mission d'exploration.

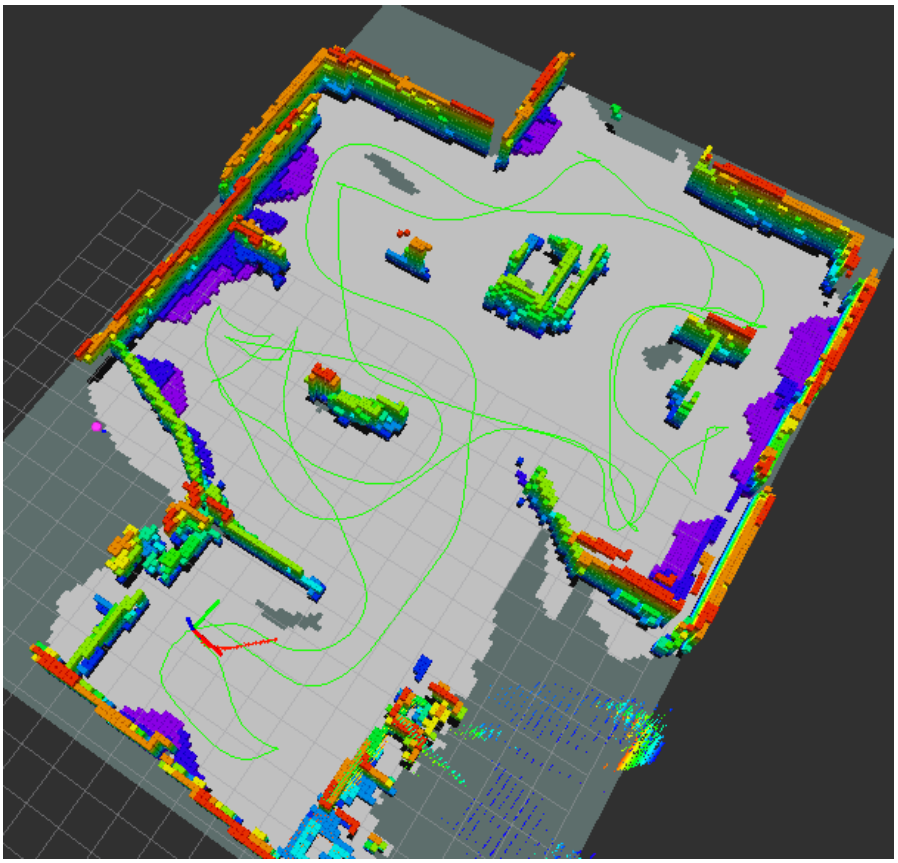


FIGURE IV.15 – Résultat d'une mission d'exploration

Cependant, quelques points sont à signaler :

— On a remarqué dans les expériences que le robot se bloque facilement dans les coins, comme on peut l'observer dans la figure IV.14b. La trajectoire optimale calculée dans ce cas est la trajectoire nulle car la zone derrière le robot est déjà explorée et donc ne l'attire pas et les obstacles devant lui le bloquent. Le robot déclenche alors la procédure de déblocage et passe en mode ralliement de points. Le coût de ralliement de points donne des valeurs de coût beaucoup plus faible pour les trajectoires qui s'approchent du point que pour la trajectoire nulle, ce qui a pour effet de sortir le robot du coin.

— Lorsque le robot effectue des rotations importantes sur lui-même, comme un demi-tour, les vibrations sont importantes. Une légère dérive peut être observée sur

la valeur de l'angle d'orientation du robot. Cette dérive n'est pas suffisamment importante pour être considérée comme une valeur aberrante dans le filtre de Kalman, mais peut à terme donner un calcul de localisation complètement faussé. Ces situations arrivent souvent quand le robot se retrouve dans un coin. Pour pallier ce problème, on peut empêcher le robot d'effectuer de telles rotations en lui permettant de reculer, ce qui nécessiterait de changer la valeur de $v_{\min}$. Il faudrait alors s'assurer que le robot recule uniquement dans des zones déjà explorées afin qu'il ne puisse pas rencontrer d'obstacles non cartographiés, puisqu'il ne peut pas percevoir la scène derrière lui.

— Il peut arriver que le point calculé dans la procédure de déblocage du robot ne puisse finalement pas être atteint à cause d'obstacles à proximité qui n'étaient pas connus au moment du calcul du point. Pour réussir à rallier le point, on a fixé une distance large entre le robot et le point pour que celui-ci soit considéré atteint. On pourrait mettre en place une procédure de calcul d'un nouveau point quand la condition de distance aux obstacles sur un point n'est plus remplie.

IV.7 Conclusion

Dans ce chapitre, on a développé une architecture complète de perception-commande pour effectuer des missions d'exploration autonome sur un robot terrestre.

Une fusion des mesures de localisation provenant des capteurs de vision, des encodeurs des roues et d'une centrale inertielle est réalisée grâce à un filtre de Kalman étendu. L'état estimé obtenu est combiné avec les informations visuelles du capteur de profondeur pour créer une carte de l'environnement qui est ensuite exploitée dans l'algorithme de commande prédictive pour calculer des trajectoires maximisant l'exploration tout en évitant les obstacles. Des procédures ont été développées pour faire face aux situations de blocage du robot. Des expériences on été réalisées en situation réelle et ont permis de démontrer l'efficacité de l'architecture proposée.

On veut développer un système qui soit portable sur tout type de plateformes robotiques, dont les drones. Comme leur capacité de charge est faible, le nombre de capteurs qui peuvent être embarqués est limité. Les capteurs de vision sont les capteurs les plus adaptés pour pouvoir effectuer les tâches de localisation et de reconstruction de l'environnement, mais la localisation visuelle n'est pas fiable dans les environnements peu texturés. La solution présenté dans ce chapitre d'effectuer une fusion de données pour détecter les erreurs dans la localisation visuelle n'est alors pas envisageable. Une autre solution serait de détecter en avance ces zones qui sont problématiques pour la localisation visuelle afin de les éviter par une action de guidage appropriée et ainsi assurer une localisation précise sur toute la trajectoire du robot. Cette approche fait l'objet des chapitres V et VI.

Chapitre V

Vision

Sommaire

Dans ce chapitre, on s'intéresse au problème de la navigation basée vision quand la qualité des images reçues est insuffisante pour permettre un calcul précis de la localisation.

Une première solution, proposée au chapitre IV, consiste à fusionner les données de la localisation visuelle avec les données de localisation provenant d'autres capteurs. Mais cette solution est réalisable à condition d'avoir d'autres capteurs embarqués sur le robot. Dans ce chapitre, on se place dans le cas où le seul capteur disponible est le banc stéréo. Il est donc nécessaire de trouver une autre solution pour s'assurer que la localisation visuelle reste précise.

Nous avons vu en section III.3 une autre approche, qu'on peut relier à la problématique de la commande duale, qui est de chercher à adapter la commande pour optimiser,

à chaque instant, les conditions de fonctionnement de la localisation visuelle. C'est l'approche choisie dans ce chapitre. Plus précisément, notre but dans ce chapitre est de prédire l'apparence future de la scène afin d'adapter la trajectoire du robot pour s'assurer que les images perçues soit suffisamment texturées et ainsi garder une localisation visuelle stable.

La principale contribution de ce chapitre est la définition d'un critère de qualité visuelle des scènes futures simple et pertinent pour la méthode d'odométrie visuelle utilisée dans nos travaux. Cette approche s'inspire des travaux de [Mostegel et al., 2014], présentés dans la section III.3.2. Ce critère, qui est fondé sur la prédiction de la position future des amers visuels utilisés par le module d'odométrie, est décrit en section V.1. Il a été intégré dans une boucle de commande basée sur la commande prédictive afin de réaliser des missions de ralliement de points de passage dans des environnements qui présentent des zones peu texturées, où le calcul de la localisation visuelle risque d'échouer. Ces expériences ainsi que les résultats obtenus sont décrits dans la partie V.2. Un deuxième critère pour évaluer la qualité d'une scène future a été proposé, il adopte une approche par régions afin de diminuer le coût de calcul. Il est présenté dans la partie V.3.

V.1 Qualité d'une scène future pour la localisation

En temps normal, l'algorithme d'odométrie visuelle eVO, présenté en section III.1.4, utilise entre 100 et 150 points afin de calculer la localisation du robot. Si le nombre de points utilisés diminue, le calcul de la localisation devient de plus en plus imprécis. Quand il n'y a plus que quelques points disponibles, le calcul peut devenir complètement faux : dans les environnements peu texturés, un algorithme de localisation visuelle ne peut pas fonctionner normalement.

Une première difficulté se situe au niveau de la localisation 3D des amers visuels par le système stéréoscopique. En effet, quand l'image gauche est peu texturée, le nombre de points extraits est faible. De plus, ces points sont moins fiables, ce qui risque de donner des associations incorrectes avec l'image droite. La détection des associations incorrectes est réalisée grâce à l'algorithme RANSAC (voir III.1.3.3). La procédure fonctionne efficacement quand le nombre de valeurs aberrantes est limité par rapport à l'ensemble des données. Mais quand le nombre d'associations incorrectes est important par rapport au nombre de points total, il est alors impossible de trouver un modèle correct pour l'ensemble des données et de repérer les valeurs aberrantes.

Une deuxième difficulté apparaît lors du calcul de pose. Ce calcul est réalisé en minimisant un critère des moindres carrés. Plus le nombre de points utilisés est faible et plus le calcul de la localisation est imprécis. De plus, si le rejet des associations incorrectes n'a pas fonctionné et que certaines sont utilisées dans ce calcul, le résultat risque d'être complètement faux.

Pour que l'algorithme eVO fonctionne correctement, il faut donc que le nombre d'amers extraits dans les images soit suffisant. Le but de ce chapitre est d'éviter que le robot se retrouve face à des scènes peu texturées, en choisissant sa trajectoire future

de telle sorte qu'on s'assure que le nombre d'amers visibles reste suffisant.

Pour ce faire nous proposons un critère qui cherche à prédire le nombre de points qui seront vus dans les images futures à partir des points connus, déjà triangulés par l'algorithme eVO. Si le nombre de points prédits comme visibles dans la position future est suffisant, on suppose alors que l'algorithme pourra effectuer un calcul correct de la localisation. La formulation du critère est exposée dans la partie V.1.1.

Ce critère a été étudié dans des expérimentations en boucle ouverte le long de trajectoires suivies par un robot afin de vérifier que la prédiction du nombre de points est représentative du nombre de points qui seront réellement vus dans les images futures. Les résultats sont présentés dans la section V.1.2.

V.1.1 Prédiction de la visibilité des amers de eVO

Le but du critère développé est de prédire le nombre d'amers qui seront visibles dans les images futures à partir des points déjà connus. Les incertitudes apparaissant dans le processus sont prises en compte afin d'affiner le résultat de prédiction.

On suppose qu'à l'instant actuel t_n, une liste de points $\mathcal{Y}_n = (Y_i)_{i\in[\![1,M]\!]}$ est connue, avec M le nombre de points. Ces points ont été triangulés par l'algorithme eVO à l'instant t_n. On veut prédire le nombre de points qui seront visibles à l'instant $t_{n+H_{\text{loc}}}$, avec H_{loc}, l'horizon de prédiction de la scène. A $t_{n+H_{\text{loc}}}$, la pose désirée du robot, c'est-à-dire celle qu'on cherche à atteindre par une séquence de commande, est notée $P_{n+H_{\text{loc}}}$. Pour chaque amer 3D en mémoire à l'instant t_n, le but est de prédire s'il sera visible dans l'image qui sera acquise depuis la position $P_{n+H_{\text{loc}}}$.

La prédiction de la visibilité future des amers est effectuée en plusieurs étapes. La première étape consiste à projeter l'amer 3D sur le plan image dans la position future, ce processus est décrit dans la section V.1.1.1. Les incertitudes sur la position du point projeté sont estimées, afin de déduire sa probabilité d'apparaître dans l'image. Ces développements font l'objet de la section V.1.1.2. La formulation du critère final est proposée dans la section V.1.1.4. La figure V.1 illustre le procédé de la prédiction.

V.1.1.1 Position de l'amer 3D dans l'image future

On note p position d'un point d'eVO dans l'image actuelle. Ce point est l'image d'un amer 3D de coordonnées Y dans le repère caméra courant. On note p' la position de la projection du même amer 3D dans l'image future. $P_{n+H_{\text{loc}}}$ est la matrice de transformation entre la pose actuelle et la pose future. Par simplification des notations, on note (R, T) les paramètres de déplacement (matrice de rotation et vecteur de translation) dans cette partie. On a donc :

$$P_{n+H_{\text{loc}}} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \tag{V.1}$$

A t_n, les coordonnées Y de l'amer 3D sont exprimées dans le repère caméra courant $\mathcal{C}_n$. Pour calculer la position du point dans l'image future, il faut d'abord exprimer ce

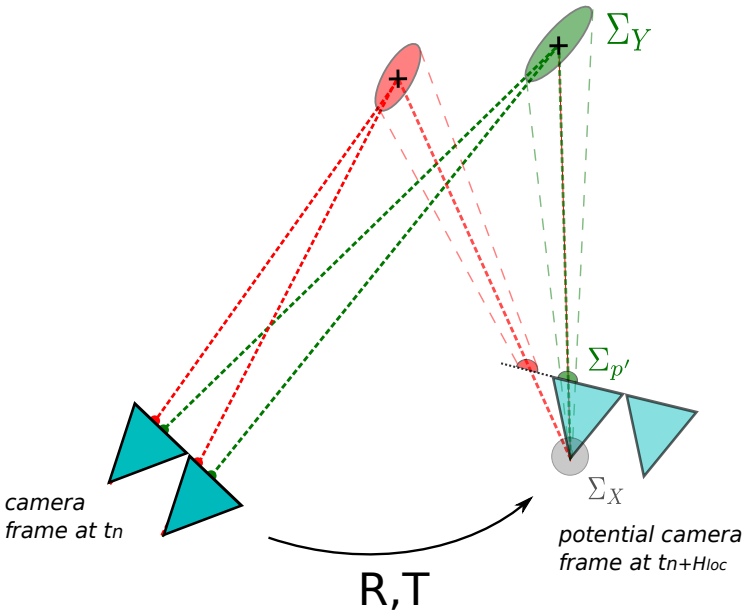


FIGURE V.1 – Schéma représentant le principe du critère développé : à l'instant t_n, deux amers ont été triangulés par eVO. Ils sont projetés dans le plan image futur. L'incertitude sur la position est évaluée. Dans cet exemple, le point vert est considéré comme visible à $t_{n+H_{\text{loc}}}$ et le point rouge ne l'est pas.

point 3D dans le repère de la caméra future $\mathcal{C}_{n+H_{\text{loc}}}$, en appliquant la formule :

$$Y' = R.Y + T \tag{V.2}$$

On peut ensuite projeter le point Y' sur le plan image grâce à la fonction de projection (II.8).

On note $\Theta = (\theta_x, \theta_y, \theta_z, t_x, t_y, t_z)$ le vecteur des paramètres de déplacement, avec $\theta_\bullet$ les angles de rotation et $t_\bullet$, les composantes du vecteur de translation T et $\eta = (u, v, d)$ le vecteur paramétrant l'amer 3D, avec (u, v) les coordonnées de de son image p dans le repère image de la caméra gauche et d la disparité mesurée de son image dans la caméra droite. p' est exprimé en fonction de Θ et η par :

$$p' = \Pi \left[R(\Theta) \cdot \Pi^{-1}(\eta) + T(\Theta) \right] \tag{V.3}$$

Π est la fonction de projection (II.8) et Π^{-1} est la fonction de triangulation (III.1).

V.1.1.2 Calcul des incertitudes

L'incertitude sur la position du point dans l'image future p' est exprimée sous la forme d'une matrice de covariance de dimension 2, notée $\Sigma_{p'}$.

L'incertitude sur la position de p' dans l'image prend en compte les incertitudes relatives au déplacement du robot et celles relatives à la triangulation du point 3D. Les deux matrices de covariances correspondantes sont notées respectivement Σ_Θ et Σ_η. Les incertitudes relatives à Θ et celles relatives à η sont supposées indépendantes.

Elles s'expriment sous la forme :

$$\Sigma_\Theta = \begin{bmatrix} \sigma_{\theta_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\theta_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\theta_z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{t_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{t_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{t_z}^2 \end{bmatrix} \tag{V.4}$$

$$\Sigma_\eta = \begin{bmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_d^2 \end{bmatrix} \tag{V.5}$$

Le choix de ces matrices n'est pas unique. Pour Σ_η, c'est un choix classique, on suppose une performance de détection isotrope dans l'image de gauche et l'indépendance de l'erreur en disparité. Pour Σ_Θ, on a supposé que les différents paramètres de pose sont indépendants entre eux. Dans tous les cas, les calculs qui suivent ne dépendent pas du choix de ces matrices.

On note h la fonction (V.3) exprimant p' en fonction de Θ et η :

$$p' = h\left(\Theta, \eta\right) \tag{V.6}$$

La matrice de covariance sur la position de p' est

$$\Sigma_{p'} = J_{h_\Theta} \cdot \Sigma_\Theta \cdot J_{h_\Theta}^T + J_{h_\eta} \cdot \Sigma_\eta \cdot J_{h_\eta}^T \tag{V.7}$$

avec J_{h_Θ} et J_{h_η} les matrices jacobiennes de h relatives aux paramètres Θ et η respectivement :

$$J_{h_\Theta} = \frac{\partial h}{\partial \Theta} = J_\Pi\left(Y'\right) \cdot J_{Y'}\left(Y\right) \tag{V.8}$$

$$J_{h_\eta} = \frac{\partial h}{\partial \eta} = J_\Pi\left(Y'\right) \cdot R \cdot J_{\Pi^{-1}}\left(\eta\right) \tag{V.9}$$

Les matrices $J_{\Pi^{-1}}$, J_Π et $J_{Y'}$ sont respectivement les matrices jacobiennes des fonctions de triangulation, de projection et de changement de base. Leurs expressions mathématiques sont données dans les paragraphes suivants.

Matrice jacobienne de la fonction de triangulation

L'expression de la fonction de triangulation est donnée en (III.1). Sa matrice jacobienne est :

$$J_{\Pi^{-1}}(\eta) = \frac{\partial \Pi^{-1}(\eta)}{\partial \eta} = \frac{b}{d^2} \cdot \begin{bmatrix} -d & 0 & u-u_0 \\ 0 & -d & v-v_0 \\ 0 & 0 & \alpha \end{bmatrix} \tag{V.10}$$

Matrice jacobienne de la fonction de projection

L'expression de la fonction de projection est donnée en (II.8). Sa matrice jacobienne est :

$$J_{\Pi}(Y') = \frac{\partial \Pi(Y')}{\partial Y} = \frac{\alpha}{z'^2} \cdot \begin{bmatrix} z' & 0 & -x' \\ 0 & z' & -y' \end{bmatrix} \tag{V.11}$$

Matrice jacobienne de la fonction de changement de base

La matrice jacobienne du changement de repère s'écrit :

$$J_{Y'}(Y) = \left.\frac{\partial Y'}{\partial \Theta}\right|_Y = \frac{\partial (R.Y+T)}{\partial \Theta} = \left[\frac{\partial R}{\partial \theta_x, \theta_y, \theta_z} \cdot Y \;\middle|\; \frac{\partial T}{\partial t_x, t_y, t_z} \right] \tag{V.12}$$

On dérive R et T par rapport aux paramètres de déplacement. R est indépendant des paramètres de la translation et T est indépendant des paramètres de rotation.

La dérivée de la matrice de rotation par rapport aux angles de rotation est donnée par :

$$\frac{\partial R}{\partial \theta_x, \theta_y, \theta_z} = \left[R_z R_y \frac{\partial R_x}{\partial \theta_x} \;\middle|\; R_z \frac{\partial R_y}{\partial \theta_y} R_x \;\middle|\; \frac{\partial R_z}{\partial \theta_z} R_y R_x \right] \tag{V.13}$$

V.1.1.3 Probabilité du point d'appartenir à l'image

Cette section décrit l'estimation de la probabilité qu'un point p appartienne à l'image à partir de sa matrice de covariance Σ_p. La matrice de covariance permet de définir une ellipse de confiance à 90 % dans l'image. Le principe du calcul de probabilité est d'évaluer l'aire de l'intersection entre l'ellipse et l'image.

La première étape consiste à exprimer l'ellipse dans un repère permettant de simplifier le calcul de l'aire.

L'équation de l'ellipse dans le repère image est :

$$(X-p)^T \Sigma_p^{-1} (X-p) = s \tag{V.14}$$

$s = 4.605$ pour une ellipse de confiance à 90 %. Cette valeur peut être trouvée dans une table de distribution du χ^2 avec 2 degrés de liberté.

On définit $(\mathcal{E})$ le repère ellipse comme suit : l'origine est le point p, l'axe x est l'axe principal de l'ellipse et y le deuxième axe de l'ellipse. Dans ce repère, l'équation de l'ellipse s'écrit :

$$X'^T \Sigma_D^{-1} X' = s \tag{V.15}$$

avec $\Sigma_D = P^T \Sigma_p P$ matrice diagonale et $X' = P^T (X - p)$. Ici, P est la matrice de passage entre le repère image et le repère ellipse. L'aire de l'ellipse vaut $\mathcal{A}_{ell} = \pi s \sqrt{\lambda_1 \lambda_2}$ avec $\lambda_{1,2}$ les valeurs propres de Σ_p (et les éléments de la diagonale de Σ_D).

Après avoir exprimé l'équation de l'ellipse dans le repère $(\mathcal{E})$, la deuxième étape consiste à calculer l'aire de l'intersection entre l'ellipse et l'image. Pour cela, on va d'abord chercher les points d'intersection entre l'ellipse et les droites définissant les bords de l'image. On obtient un système à résoudre :

$$\begin{cases} X'^T \Sigma_D^{-1} X' = s \\ L_i^T \tilde{X}' = 0 \end{cases} \tag{V.16}$$

avec $L_i = (m_i, n_i, q_i)$ les paramètres de la droite i :

$$m_i.x + n_i.y + q_i = 0 \tag{V.17}$$

Plusieurs cas se présentent alors.

Dans le premier cas, il n'existe pas de solution, ce qui signifie qu'il n'y a pas d'intersection entre l'ellipse et les 4 droites. L'ellipse est soit entièrement à l'extérieur de l'image, soit entièrement à l'intérieur. Un test sur la position du point p est effectué pour définir la position de l'ellipse. Si le point est en dehors de l'image, l'aire de l'intersection entre l'image et l'ellipse est nulle. Au contraire, si le point est dans l'image, l'aire est celle de l'ellipse.

Dans le second cas, la solution du système est non vide, il existe des points d'intersection entre l'image et l'ellipse. L'aire de l'intersection est calculée par une double intégrale sur le domaine D, délimité par l'intersection de l'ellipse et de l'image.

$$\mathcal{A} = \iint_{\mathcal{D}} dxdy \tag{V.18}$$

$$\mathcal{D} = \{X \in \mathbb{R}^2 \mid X^T \Sigma_D^{-1} X \leqslant s \text{ et } L_i^T \tilde{X} \leqslant 0, i \in \{1, 4\}\} \tag{V.19}$$

Cette double intégrale peut être évaluée par un algorithme d'approximations successives (fonction *dblquad* du package scipy.integrate) qui prend un temps variable. Pour simplifier ce calcul et obtenir une solution en temps limité, l'ellipse est approximée par un rectangle de même orientation dont la longueur et la largeur sont réciproquement la longueur du grand axe et du petit axe de l'ellipse.

Le calcul de la double intégration devient donc le calcul de l'aire d'un polygone convexe, formé par les points d'intersection entre les 8 droites formant les deux rectangles

qui appartiennent à $\mathcal{D}$. Pour un polygone convexe, si les m sommets sont rangés dans l'ordre trigonométrique, l'aire est donnée par la formule :

$$\mathcal{A} = \frac{1}{2}\left[x_0\,(y_1 - y_{m-1}) + x_{m-1}\,(y_0 - y_2) + \sum_{i=1}^{m-2} x_i\,(y_{i+1} - y_{i-1})\right] \tag{V.20}$$

Finalement, la probabilité pour un point d'appartenir à l'image est donnée par le rapport entre l'aire de l'intersection sur l'aire totale de l'ellipse.

$$P_r = \frac{\mathcal{A}}{\mathcal{A}_{ell}} \tag{V.21}$$

L'algorithme 3 résume le déroulement du calcul de la probabilité pour le point p d'apparaître dans l'image.

Algorithm 3 Calcul de la probabilité P_r pour un point p d'être dans l'image

Require: position du point p, matrice de covariance Σ_p
  Calcul des paramètres de l'ellipse par diagonalisation de Σ_p
  Test de l'existence de solutions du système (V.16)
  if l'ensemble des solutions est vide **then**
    Test sur la position de p
    if p est dans l'image **then**
      $P_r = 1$
    else
      $P_r = 0$
    end if
  else
    Calcul des paramètres des droites de l'image dans le repère $\mathcal{E}$
    Calcul des points d'intersection des droites des deux rectangles dans $\mathcal{D}$
    Calcul de l'aire du polygone formé par les points d'intersection par (V.20)
    Calcul de la probabilité P_r par (V.21)
  end if
  return P_r

V.1.1.4 Formulation du critère

Le critère pour mesurer la qualité d'une position future pour la localisation visuelle est le nombre de points dont la probabilité d'apparaître dans l'image est supérieure à un seuil s_{proba}.

L'algorithme 4 décrit la procédure globale pour obtenir la prédiction du nombre de points visibles. A l'instant actuel t_n, on connaît la liste de points $\mathcal{Y}_n$, fournie par l'algorithme de localisation visuelle et $P_{n+H_{\text{loc}}}$, la pose désirée à l'instant de prédiction $t_{n+H_{\text{loc}}}$.

Algorithm 4 Prédiction à t_n du nombre de points visibles à $t_{n+H_{\text{loc}}}$

$N \leftarrow 0$
Require: liste des points 3D $\mathcal{Y}_n$, pose future désirée $P_{n+H_{\text{loc}}}$
for Y in $\mathcal{Y}_n$ **do**
Calcul de la position du point image futur p' par (V.3)
Calcul de la covariance sur le point $\Sigma_{p'}$ par (V.7)
Calcul de la probabilité P_r pour le point d'apparaître dans l'image par l'algorithme 3
if $P_r > s_{proba}$ **then**
$N \leftarrow N + 1$
end if
end for
return N

V.1.2 Validation expérimentale

Des expériences en boucle ouverte ont été réalisées afin de valider le critère présenté dans la partie précédente. Le but des expériences est de comparer le nombre de points prédits par l'algorithme par rapport aux nombres de points qui seront réellement vus dans la suite de la trajectoire afin de vérifier si la prédiction est correcte. On veut également vérifier si l'algorithme détecte à l'avance les zones où le nombre de points visibles est faible, situation dans laquelle le calcul de l'odométrie visuelle risque d'être peu précis.

Dans ces expériences, le robot est téléguidé par un opérateur. La même trajectoire a été réalisée trois fois et à chaque passage des marqueurs ont été ajoutés sur les murs afin d'augmenter la texture de la scène et donc le nombre de points potentiellement utilisables par l'algorithme d'odométrie visuelle. La figure V.2 montre trois images extraites à partir de chaque séquence. On peut remarquer que le mur est très peu texturé dans la séquence 1, si le robot lui fait face, le calcul de la localisation visuelle risque d'échouer. Dans les séquences 2 et 3, le mur est plus texturé et ne devrait pas poser de difficultés pour le calcul de la localisation.

Les valeurs données aux paramètres sont :

— Pas de temps : $t_e = 0.25$ s
— Seuil probabilité : $s_{proba} = 0.5$
— Matrices de covariance
$\Sigma_\Theta = \text{diag}(\sigma^2_{\theta_x}, \sigma^2_{\theta_y}, \sigma^2_{\theta_z}, \sigma^2_x, \sigma^2_y, \sigma^2_z)$ et $\Sigma_\eta = \text{diag}(\sigma^2_u, \sigma^2_v, \sigma^2_d)$ avec :
  ◦ incertitudes sur la position : $\sigma^2_x = \sigma^2_y = \sigma^2_z = 0.005$ m
  ◦ incertitudes sur l'orientation : $\sigma^2_{\theta_x} = \sigma^2_{\theta_y} = \sigma^2_{\theta_z} = 0.001$ rad
  ◦ incertitudes sur la position du point dans l'image courante : $\sigma^2_u = \sigma^2_v = 0.2$ pixel
  ◦ incertitudes sur la disparité : $\sigma^2_d = 0.4$ pixel

(a) Séquence 1

(b) Séquence 2

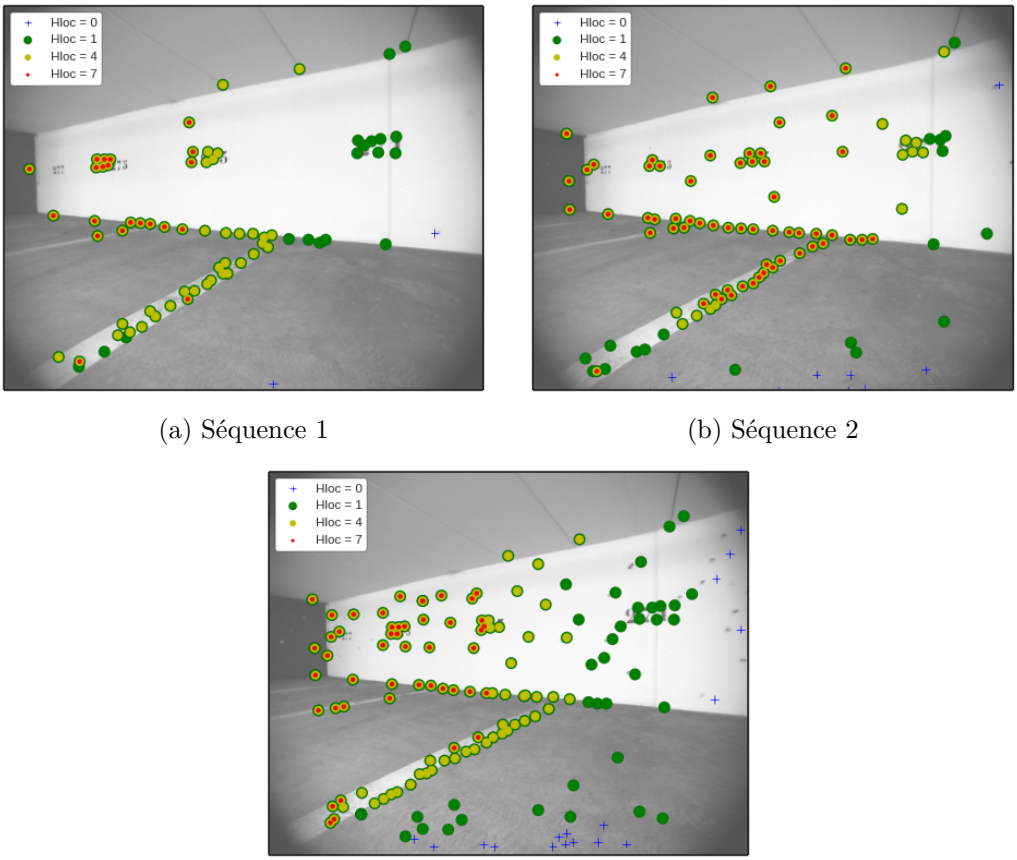
(c) Séquence 3

FIGURE V.2 – Images extraites des trois séquences

La figure V.3 montre le résultat de prédiction à un instant pour chaque séquence. La trajectoire future du robot est une rotation vers la gauche. On voit que les points situés sur la droite de l'image sont progressivement considérés comme non visibles à mesure que l'horizon de prédiction de scène H_{loc} augmente.

L'algorithme de prédiction est exécuté après la réalisation de la trajectoire. La pose désirée $P_{n+H_{\text{loc}}}$ est fixée en prenant la position du robot à $t_{n+H_{\text{loc}}}$ sur la trajectoire effectuée.

La figure V.4 compare le nombre de points prédits avec le nombre de points qui sont réellement vus parmi les points prédits pour un horizon de prédiction de scène $H_{\text{loc}} = 1$. On peut remarquer que le nombre de points réellement vus est toujours proche du nombre de points prédits, mais souvent légèrement inférieur à cette valeur. Ce phénomène peut être expliqué par le fait que certains amers utilisés pour la prédiction n'ont pas d'existence physique réelle. Leur position future théorique est prédite mais ils ne sont donc pas retrouvés dans les images futures. De plus, après le déplacement du robot, l'apparence de certains points change, ce qui ne permet pas de les reconnaitre



(a) Séquence 1

(b) Séquence 2

(c) Séquence 3

FIGURE V.3 – Images extraites des trois séquences avec le résultat de prédiction des points pour différents horizons : en vert, les points qui seront visibles avec $H_{\text{loc}} = 1$, en jaune les points qui seront visibles avec $H_{\text{loc}} = 4$ et en orange, les points qui seront visibles avec $H_{\text{loc}} = 7$. La trajectoire future du robot est une rotation vers la gauche.

dans les images futures.

Par ailleurs, on observe une chute importante du nombre de points au niveau du pas de temps 200, qui a correctement été détectée par l'algorithme de prédiction.

La figure V.5 montre le résultat de la prédiction sur la trajectoire totale pour les trois séquences avec un horizon de prédiction de scène de $H_{\text{loc}} = 1$. Le cercle rouge montre la zone où les marqueurs ont été ajoutés sur le mur. La couleur des marqueurs formant la trajectoire permet d'afficher le nombre de points qui sont prédits visibles par l'algorithme de prédiction : plus la couleur est vers le bleu et plus le nombre de points prédits est faible. On peut observer que le nombre de points prédits dans la zone cerclée en rouge est inférieur dans la première séquence par rapport aux deux autres séquences. L'algorithme de prédiction a correctement prédit un nombre de points faible dans la première séquence et un nombre de points plus élevés dans les deux autres. La différence

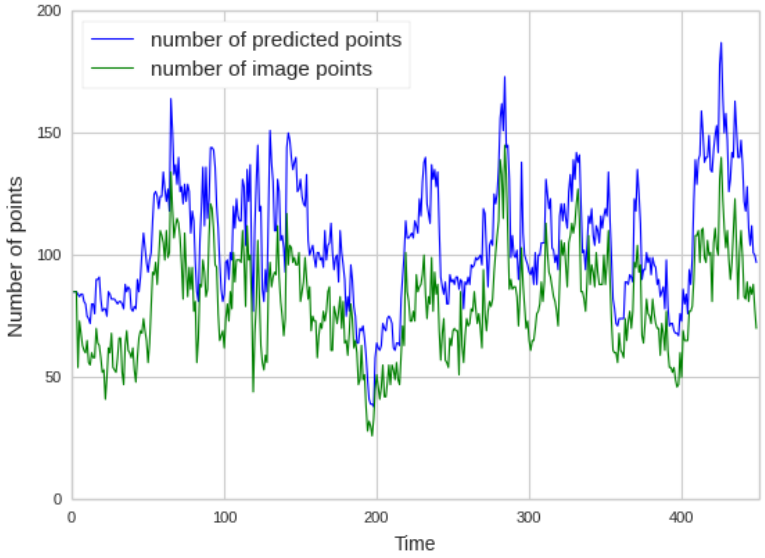


FIGURE V.4 – Comparaison entre le nombre de points prédits et le nombre de points qui seront réellement vus

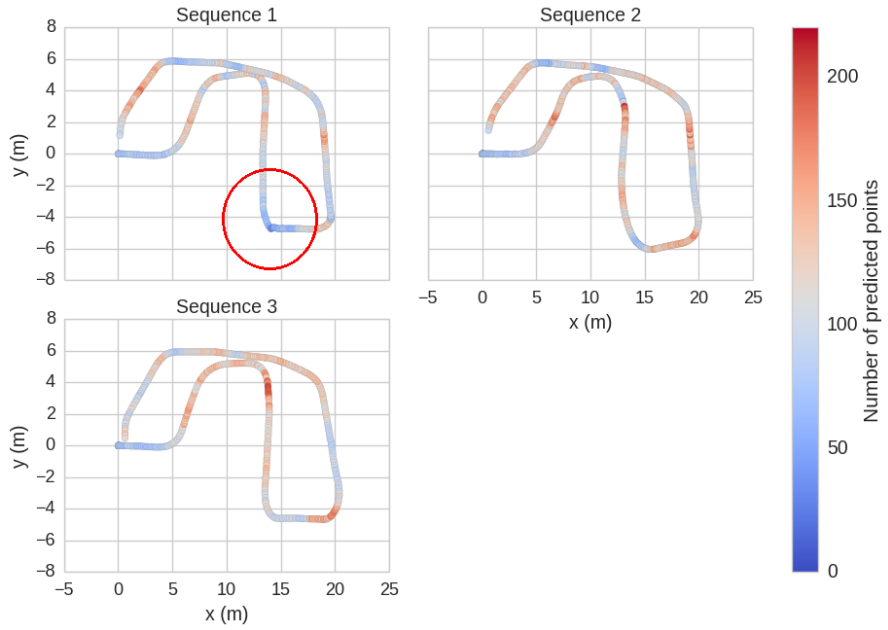


FIGURE V.5 – Résultat de prédiction pour les trois séquences. Le cercle rouge désigne la zone où les marqueurs ont été ajoutés au mur

de texture de la scène entre les trois séquences a ainsi été détectée.

En conclusion, ces expériences ont démontré que l'algorithme développé est efficace

pour prédire le nombre de points qui seront visibles dans les images futures, à partir des points déjà connus et connaissant la trajectoire future du robot. L'algorithme peut détecter une chute importante du nombre de points sur la trajectoire future et donc peut détecter les zones où la qualité de la scène est insuffisante pour permettre une localisation visuelle précise.

Par la suite, ce critère a été intégré dans une boucle de commande afin de réaliser des missions de navigation par points de passage dans des environnements qui présentent des zones peu texturées. Les expériences réalisées sont présentées dans la partie V.2.

V.2 Expériences de ralliement de points de passage

Le but des expériences présentées dans cette partie est de réaliser des missions de ralliement de points de passage avec un robot dans un environnement qui présente des zones peu texturées. Le critère sur la qualité de localisation, développé dans la partie précédente, est intégré à la boucle de commande pour que le robot cherche à rallier son objectif tout en s'assurant que la localisation visuelle reste précise.

Le critère sur la qualité d'une scène pour la localisation est utilisé pour formuler un nouveau coût afin de l'intégrer à la fonction de coût de la boucle de commande MPC, initialement présentée en section IV.3. La nouvelle boucle de commande obtenue est présentée dans la section V.2.1. La mise en œuvre des expériences est décrite dans la section V.2.2 et les résultats sont présentés dans la section V.2.3.

V.2.1 Intégration du critère dans une boucle de commande

Le critère défini dans la partie V.1.1 donne le nombre de points prédits visibles à l'instant $t_{n+H_{\text{loc}}}$ avec H_{loc}, l'horizon de prédiction de scène. On veut utiliser ce critère afin de favoriser les trajectoires qui présentent un nombre élevé de points prédits à $t_{n+H_{\text{loc}}}$ et pénaliser celles qui ont un faible nombre de points prédits.

Ce critère est formulé sous la forme d'un coût afin d'être intégré dans la fonction de coût de la boucle MPC. Pour cela, la valeur du coût doit être comprise entre 0 et 1 et elle doit décroître avec l'augmentation du nombre de points prédits.

Nous proposons le critère suivant :

$$J_{\text{loc}} = 1 - \frac{N(X_{n+H_{\text{loc}}}, \mathcal{Y}_n)}{N_{\max}(t_n)} \tag{V.22}$$

$N(X_{n+H_{\text{loc}}}, \mathcal{Y}_n)$ est le nombre de points prédits à $t_{n+H_{\text{loc}}}$ et $N_{\max}(t_n)$ est le nombre de points 3D connus à t_n.

Deux objectifs sont considérés pour les expériences : le ralliement de points de passage et la prise en compte de la qualité de localisation. La fonction de coût est donc formulée comme la somme de deux coûts :

$$J = w_{\text{wp}} J_{\text{wp}} + w_{\text{loc}} J_{\text{loc}} \tag{V.23}$$

La figure V.6 illustre le fonctionnement de la commande prédictive pour cette fonction de coût. Sur cet exemple, la trajectoire optimale n'est pas la trajectoire qui s'approche le plus du point à rallier mais celle qui s'en approche tout en gardant des points 3D dans son champ de vue.

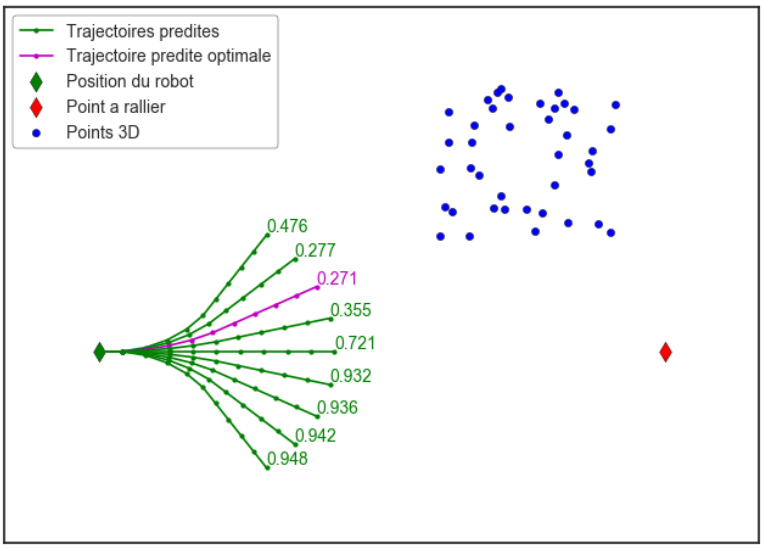


FIGURE V.6 – Calcul de la trajectoire optimale pour une mission de ralliement de points avec considération de la qualité de la localisation future

L'architecture du système utilisée pour les expériences est affichée sur la figure V.7. Pour se localiser, le robot utilise uniquement les images provenant du banc stéréo et l'algorithme d'odométrie visuelle eVO. Le module de prédiction de scène prend en entrée les amers 3D triangulés par eVO et la pose future donnée par l'algorithme de commande MPC afin de prédire le nombre d'amers qui seront visibles dans la pose future.

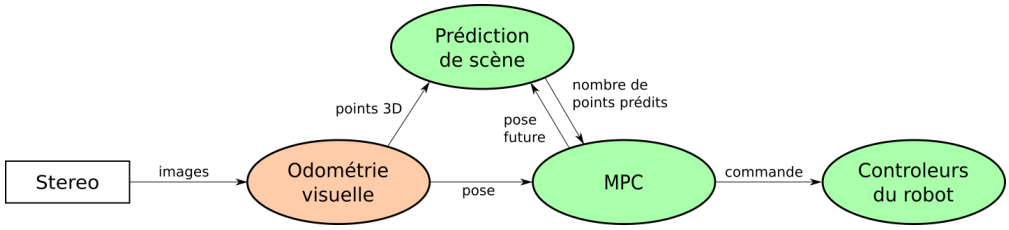


FIGURE V.7 – Architecture du système pour le ralliement de points avec considération de la qualité de localisation future

V.2.2 Mise en place des expérimentations

Le robot doit rallier un point devant lui, situé devant un mur sans texture, comme illustré à la figure V.8. Si le robot avance directement droit sur le point, il risque de

percevoir uniquement le mur sans texture dans son champ de vue et la localisation visuelle risque d'être incorrecte. Le but de ces expériences est de vérifier si l'ajout du critère sur la qualité de localisation dans la stratégie de commande permet d'améliorer le comportement du robot par rapport à la zone peu texturée.

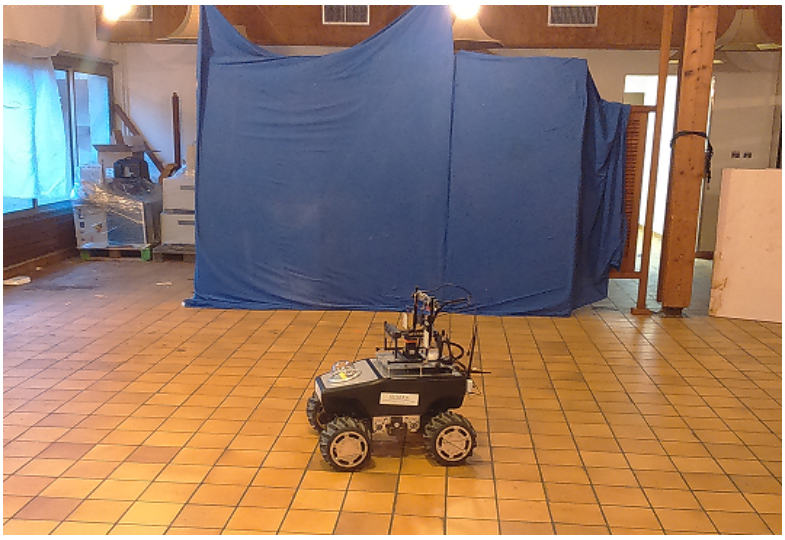


FIGURE V.8 – L'environnement expérimental, le mur sans texture est derrière le robot

Le robot utilisé est le Robotnik Summit XL, sa description est disponible en section II.2.2. La vérité terrain est fournie par l'odométrie des roues et la centrale inertielle. L'odométrie des roues estime la distance parcourue tandis que la centrale inertielle mesure l'orientation du robot. L'odométrie visuelle présente une dérive sur l'estimation de la position qui augmente avec la distance parcourue. Dans ces expériences, les trajectoires effectuées sont courtes, on suppose que la dérive sur l'estimation de la position par l'odométrie des roues est négligeable.

On utilise les paramètres fixés dans les expériences de validation du critère sur la localisation visuelle V.1.2. Pour la boucle de commande, on a fixé $H_c = 5$ et $H_p = 8$. Les horizons sont fixés avec des valeurs plus petites que celles utilisées dans les expériences d'exploration. Les objectifs considérés dans ces expériences ne nécessitent pas de fixer des horizons plus longs.

Le point à rallier est situé à 5.8 m devant le robot. On considère que le point est atteint si le robot se trouve à une distance inférieure à 0.5 m du point.

La vitesse linéaire maximale du robot est fixée à $v_{\max} = 0.4$ m.s^{-1}. Les vitesses angulaires extrémales sont $\omega_{\max} = 0.5$ rad.s^{-1} et $\omega_{\min} = -0.5$ rad.s^{-1}.

Le temps de calcul du coût de qualité de localisation étant assez important, il n'est plus possible d'utiliser l'algorithme d'optimisation DIRECT comme dans les expériences d'exploration décrites dans le chapitre précédent IV.6. On a donc défini un ensemble de trajectoires à tester. Il contient 7 valeurs de vitesses angulaires, réparties sur l'intervalle $[\omega_{\min}; \omega_{\max}]$. On ajoute à cet ensemble la commande nulle $(v = 0, \omega = 0)^T$.

Le point de départ est le même pour toutes les réalisations.

V.2.3 Résultats obtenus

Une première série d'expériences a été réalisée pour fixer l'horizon de prédiction de scène H_{loc}. On a choisi de fixer la valeur à $H_{\text{loc}} = 4$. Avec une valeur plus petite, la prédiction est à trop court-terme. Avec une valeur plus élevée, le robot a peu de points dans son champ de vue futur pour la plupart des trajectoires et préfère généralement suivre une trajectoire rectiligne.

Après avoir fixé la valeur de H_{loc}, des expériences ont été réalisées dans le but de fixer les pondérations sur les coûts w_{wp} et w_{loc}. On fixe $w_{\text{wp}} = 1 - w_{\text{loc}}$. Pour la suite, on s'intéresse uniquement au réglage de la pondération w_{loc}.

La figure V.9 présente les trajectoires réalisées pour différentes pondérations w_{loc} comprises entre 0.0 et 0.5. Quand $w_{\text{loc}} = 0.0$, la fonction de coût ne prend en compte que le coût de ralliement de points, le coût sur la qualité de scène n'est pas considéré. On peut remarquer sur la figure que la trajectoire calculée par eVO est incorrecte. Le robot a dû être stoppé par l'opérateur.

Dans les expériences suivantes, la pondération w_{loc} a été graduellement augmentée. On observe que le robot choisit des trajectoires moins directes vers le point, mais qui permettent de percevoir plus de points 3D dans son champ de vision et donc d'obtenir un calcul de localisation plus précis. Avec $w_{\text{loc}} = 0.1$, la localisation visuelle est encore incorrecte mais présente une dérive plus faible que dans le cas précédent. Avec $w_{\text{loc}} = 0.2$ ou 0.3, la localisation visuelle est proche de la vérité terrain mais le robot ne parvient pas à atteindre le point. Quand $w_{\text{loc}} = 0.4$ ou 0.5, la localisation visuelle est correcte et le robot parvient à rejoindre le point.

Si on donne à w_{loc} une valeur plus élevée, l'importance du critère sur la qualité de localisation devient plus forte que celle du critère de ralliement de points. Le robot ne cherche plus à rejoindre son objectif mais préfère rester sur des positions où le nombre de points prédits est important.

La figure V.10 montre le nombre de points utilisés dans l'algorithme d'odométrie visuelle à chaque pas de temps. Pour chaque trajectoire, le nombre de points diminue quand le robot approche du mur. La diminution du nombre de points est plus importante pour les trajectoires réalisées avec une valeur petite sur w_{loc} car le robot s'approche du mur peu texturé de face alors que les trajectoires réalisées avec une valeur de w_{loc} présentent un nombre de points minimal plus élevé car le robot approche du mur avec un angle d'incidence plus grand, ce qui permet de percevoir sur une partie de l'image, une scène plus texturée et donc plus de points.

Le nombre de points 3D minimal sur chaque trajectoire est affiché sur la figure V.11. On peut observer que ce nombre augmente avec w_{loc}. Ces deux figures montrent que le critère sur la qualité de la localisation future permet au robot d'adapter sa trajectoire afin d'augmenter le nombre de points utilisés dans l'algorithme d'odométrie visuelle.

Pour chaque valeur de pondération w_{loc}, l'expérience a été répétée dix fois. L'erreur en translation sur la distance parcourue entre l'odométrie visuelle et la vérité terrain a été calculée. La figure V.12 affiche la moyenne et le maximum obtenus pour chaque

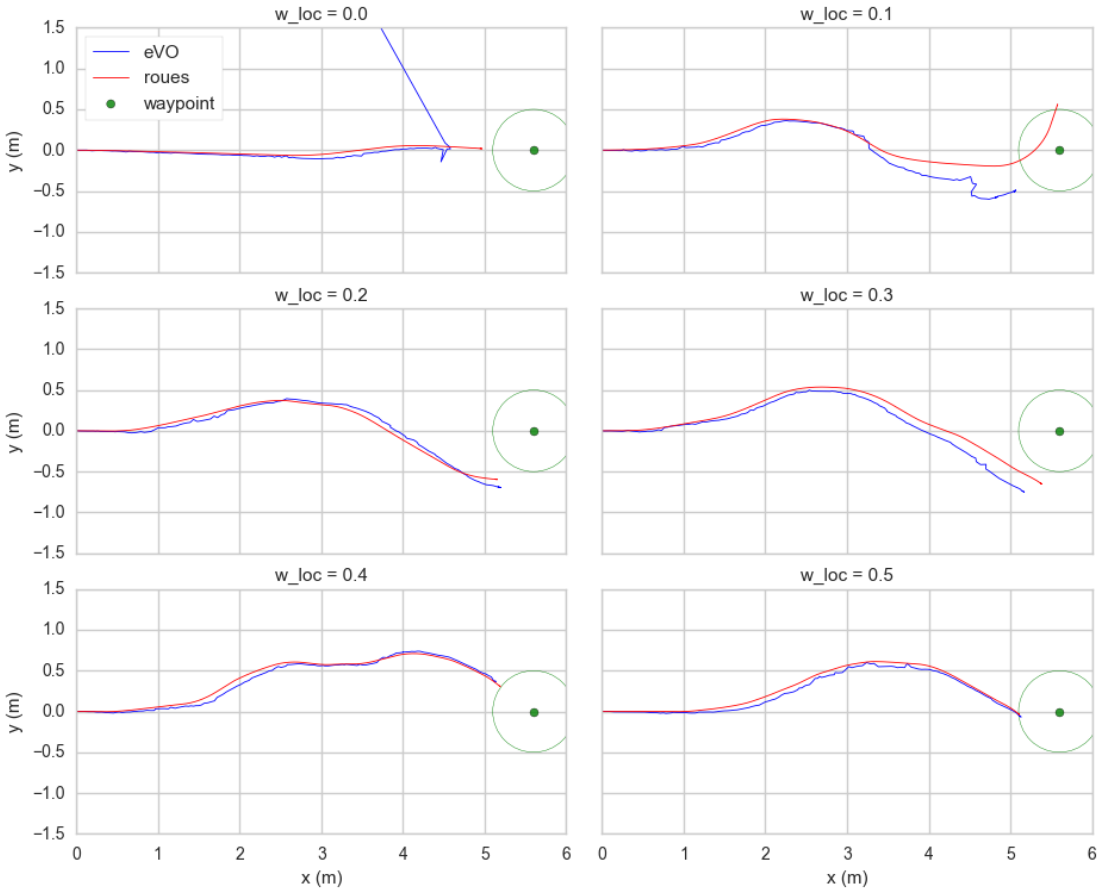


FIGURE V.9 – Trajectoires réalisées par le robot pour différentes pondérations w_{loc}. La courbe bleue est la trajectoire calculée par eVO, la courbe rouge est la vérité terrain, le point vert est le point à rallier et le cercle vert représente la zone dans laquelle le point est considéré atteint

pondération. Quand w_{loc} est inférieur à 0.4, l'erreur maximale est importante, supérieure à 150 % et atteignant même 400 %. Dans ces cas, la localisation visuelle est incorrecte et on peut considérer que le robot est perdu. La valeur maximale de l'erreur diminue quand w_{loc} est supérieur ou égal à 0.4 et présente des valeurs inférieures à 50 %. La moyenne de l'erreur présente également une diminution avec l'augmentation de w_{loc}. On en conclut que plus la valeur de w_{loc} est élevée et plus la localisation visuelle est précise. De plus, les erreurs importantes sur la localisation visuelle apparaissent uniquement quand w_{loc} est inférieur à 0.4.

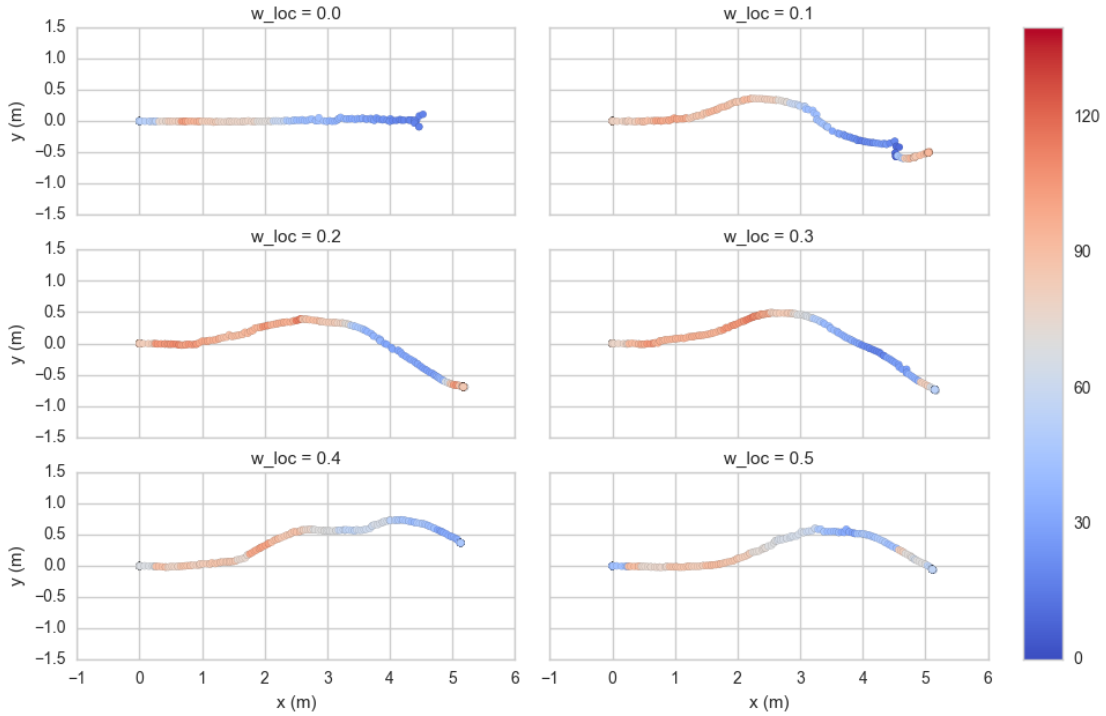


FIGURE V.10 – Trajectoires réalisées par le robot avec le nombre de points triangulés par eVO à chaque pas de temps

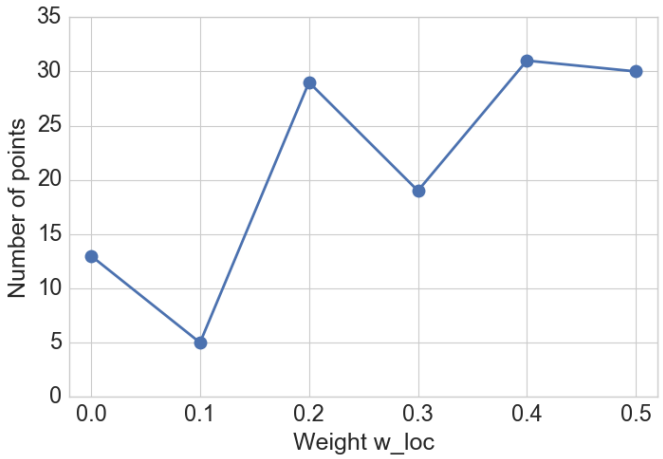


FIGURE V.11 – Nombre minimal de points triangulés par eVO sur chaque trajectoire

V.2.4 Temps de calcul

Le temps de calcul de ce critère est trop important par rapport à la période d'échantillonnage utilisée dans la boucle de commande ($t_e = 0.25$ s). Le nombre de commandes

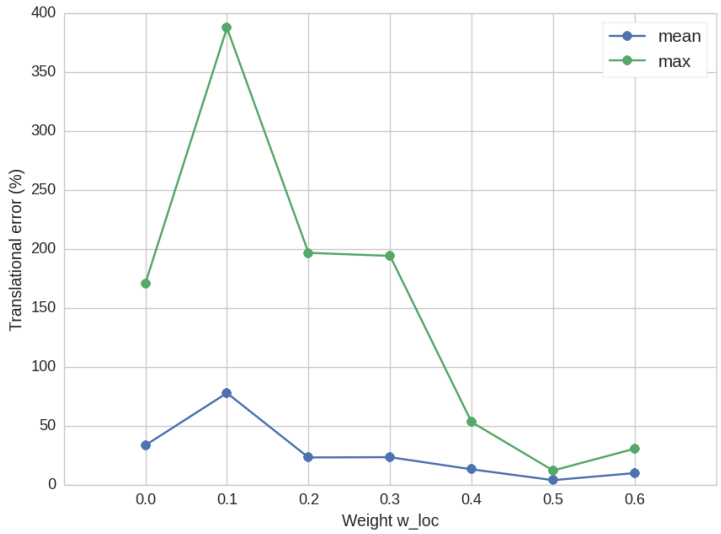


FIGURE V.12 – Erreur sur la distance parcourue entre l'odométrie visuelle et la vérité terrain

testées doit donc être limité afin de s'assurer que l'optimisation soit effectuée dans le temps imparti. Le temps moyen de calcul du coût J pour une trajectoire est de 0.015 s. Le temps de calcul est exclusivement dû au calcul du coût de localisation qui est largement supérieur au temps de calcul du coût de ralliement de points. Cette valeur de temps autorise à tester jusqu'à 16 trajectoires dans la boucle de commande.

Mais, le temps de calcul du coût de qualité de localisation est également très variable, car il dépend du nombre de points utilisés pour effectuer la prédiction. Plus la liste de points à prédire est grande et plus le temps de calcul est long. Le temps moyen de calcul pour le coût de qualité de localisation est de 0.015 s avec un écart-type de 0.006 s. Le maximum obtenu est de 0.029 s. Pour s'assurer que le calcul de la commande optimale soit assez rapide, nous avons donc diminué le nombre de trajectoires testées à 8.

V.2.5 Conclusion

Ces expériences montrent que l'ajout du critère sur la qualité de localisation améliore le comportement du robot face aux zones peu texturées. En effet, il adapte sa trajectoire afin de percevoir plus de points dans son champ de vision, ce qui permet de diminuer l'erreur sur la localisation visuelle. Avec un choix adapté des pondérations sur les coûts dans la fonction de coût, le robot parvient à rallier son objectif tout en gardant une localisation visuelle précise.

Le critère développé est efficace mais présente des temps de calculs importants, ce qui restreint le nombre de commandes candidates dans le MPC. En conséquence, nous avons décidé de développer une nouvelle stratégie pour évaluer la qualité d'une scène future pour la localisation. Cette stratégie est présentée dans la partie V.3.

V.3 Critère de qualité de localisation basé régions

Le premier critère développé, qui cherche à prédire la visibilité future des points 3D, présente des limites de fonctionnement non négligeables, décrites dans la section V.3.1. Une nouvelle méthode a été proposée afin d'évaluer la qualité d'une scène future pour la localisation, elle est présentée dans la section V.3.2. Des expériences ont été réalisées pour montrer l'apport de cette nouvelle méthode, elles sont décrites dans la section V.3.3.

V.3.1 Limites du critère sur la visibilité des points de eVO

Comme nous l'avons vu dans la section V.2.4, le critère de qualité de localisation basée sur la prédiction de points présente un coût de calcul élevé, ce qui limite le nombre de trajectoires qu'on peut évaluer dans l'approche de commande prédictive.

Par ailleurs, les amers 3D utilisés dans l'algorithme de prédiction ont été triangulés par eVO à partir des images reçues par le système stéréo. Ils sont donc bornés en distance et en angle à cause des limites de fonctionnement du système stéréo. Ceci limite la prédiction à long terme car rapidement tous les points connus sont prédits hors du champ de vue. De plus, les points potentiellement visibles dans le futur sont encore trop loin pour être détectés dans les images.

Par ailleurs, le nombre de points extraits dans les algorithmes d'odométrie visuelle stéréo est délibérément limité afin que le temps de calcul de la localisation ne soit pas trop important. Quand le nombre de points extraits est suffisant pour calculer la localisation, il n'est plus nécessaire d'en extraire de nouveaux. Or, ce comportement n'est pas compatible avec une recherche plus large, si possible exhaustive, de l'information qui sera visible dans les images futures.

Le but de cette partie est de proposer une nouvelle méthode pour évaluer la qualité d'une scène future. L'information est cherchée sur l'image entière et ne se limite plus aux amers 3D déjà connus. De plus, elle est organisée en régions afin de limiter la quantité de calculs et ainsi réduire le temps d'exécution. Le principe repose sur l'extraction d'amers 2D dans les images et l'interpolation de leur profondeur en utilisant l'information déjà connue sur les amers 3D.

La définition du critère est présentée dans la section V.3.2. Il a été testé sur des données expérimentales afin de démontrer son efficacité, les résultats sont décrits dans la section V.3.3.

V.3.2 Définition du critère basé régions

L'algorithme prend en entrée les images gauche et droite, la position actuelle des points extraits dans l'image gauche à la dernière image-clé et leur profondeur associée ainsi que la pose actuelle. Il est également nécessaire de définir la position future désirée.

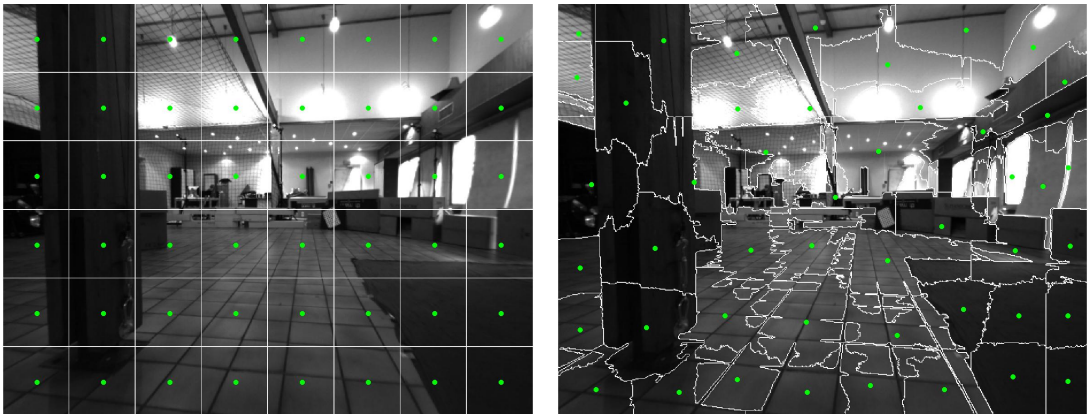
Le calcul du critère se décompose en plusieurs étapes. La première étape consiste à découper l'image gauche en régions. Pour cela, deux méthodes sont proposées, une méthode par rectangle et une méthode par superpixels. Elles sont présentées dans la section V.3.2.1. Ensuite, des points sont extraits dans l'image gauche, voir en section V.3.2.2.

Pour chaque région, on estime une profondeur à partir des points déjà connus ou en triangulant de nouveaux points. Ces développements sont présentés dans la section V.3.2.3. Enfin, la prédiction de la visibilité de chaque région est réalisée et le nombre de points supposés visibles est calculé, les explications se trouvent dans la section V.3.2.4.

V.3.2.1 Création des régions

L'image gauche est divisée en régions. Pour cela, deux méthodes ont été testées. La première est la division de l'image en rectangles de taille fixe. On a choisi de diviser la longueur en 8 et la largeur en 6, ce qui donne 48 rectangles. La figure V.13a montre l'exemple d'une image divisée par cette méthode. La deuxième méthode est la méthode par superpixels. Un superpixel est une partie de l'image, plus grande qu'un pixel normal, qui présente une couleur et une luminosité uniformes. La méthode SLIC a été utilisée [Achanta et al., 2012]. Le résultat est affiché dans la figure V.13b.

Ensuite, pour les deux méthodes, le point médian de chaque région est calculé. Ce point se situe à la médiane des coordonnées des points composant chaque région. Pour la première méthode, ce point est au centre du rectangle. Les superpixels peuvent présenter des formes diverses, qui peuvent être non convexes. Le point médian n'a pas de localisation particulière dans le superpixel et peut même se retrouver en dehors de la région dans certains cas. Les figures V.13a et V.13b affichent le point médian de chaque région.



(a) Division par rectangles (b) Division par superpixels

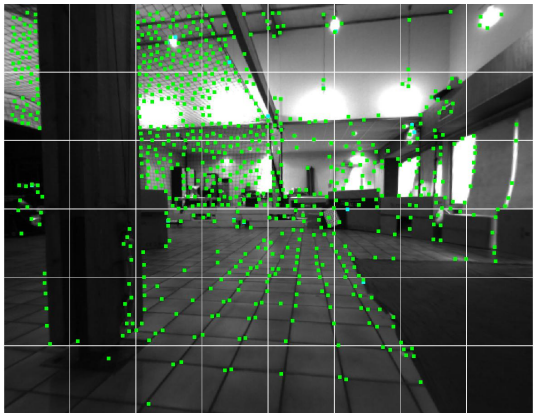
FIGURE V.13 – Division de l'image par deux méthodes

Les étapes suivantes sont les mêmes pour les deux méthodes de division de l'image.

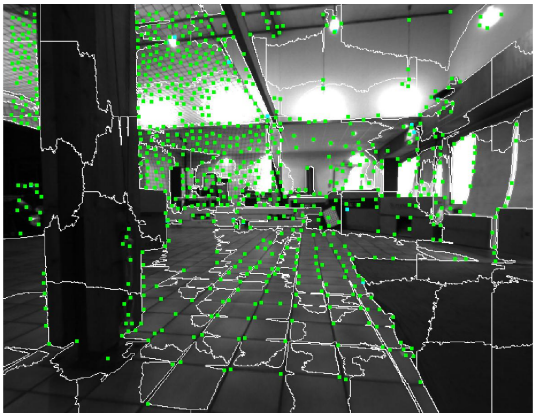
V.3.2.2 Extraction de points

Des points de Harris sont extraits dans l'image gauche. Pour chaque région de l'image, le nombre de points extraits appartenant à la région est compté. Dans la figure V.14, des

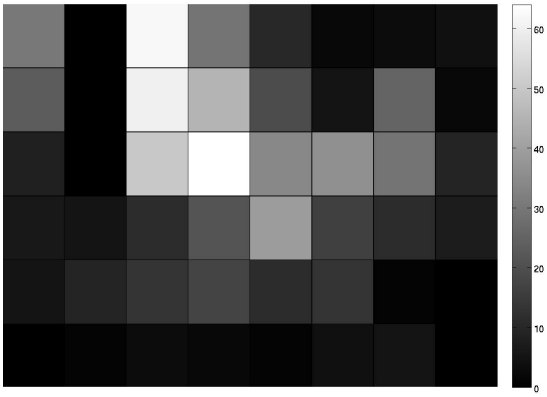
images avec les points extraits sont affichées. Pour chaque méthode, on affiche le nombre de points de chaque région.



(a) Division par rectangles



(b) Division par superpixels



(c) Division par rectangles

(d) Division par superpixels

FIGURE V.14 – Extraction de points et comptabilisation du nombre de points pour chaque région

V.3.2.3 Estimation de la profondeur

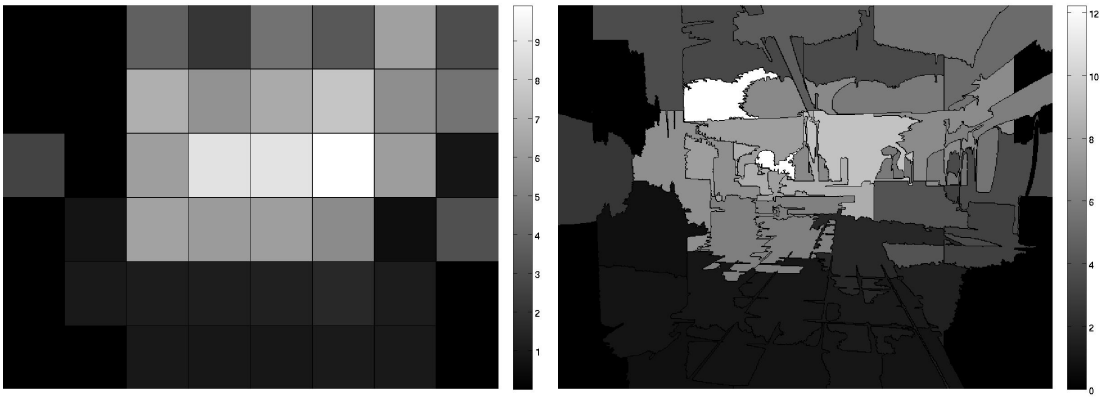
On suppose que chaque région dans l'image représente une zone plane et fronto-parallèle de la scène. Tous les points de cette région sont donc supposés être à la même profondeur. On va chercher à estimer la profondeur de chaque région à partir des amers 3D déjà connus ou en triangulant de nouveaux points.

Si des amers 3D, triangulés par l'algorithme d'odométrie visuelle, apparaissent dans la région, la profondeur de la région est estimée par la médiane des profondeurs des amers 3D. Si aucun amer 3D n'apparaît dans la région, mais que des amers 2D ont été extraits à l'étape précédente, une procédure d'association avec des points de l'image droite est exécutée dans le but d'obtenir une valeur de profondeur. Si des associations de points

sont trouvées dans la région, la profondeur médiane des amers 3D correspondants est assignée à la région.

Si aucune association n'est trouvée, ou si aucun amer 2D n'apparaît dans la région, aucune profondeur ne peut être calculée pour cette région et elle n'est donc plus considérée par la suite.

La figure V.15 montre le résultat de profondeur obtenu pour les deux méthodes.



(a) Division par rectangles (b) Division par superpixels

FIGURE V.15 – Estimation de la profondeur de chaque région

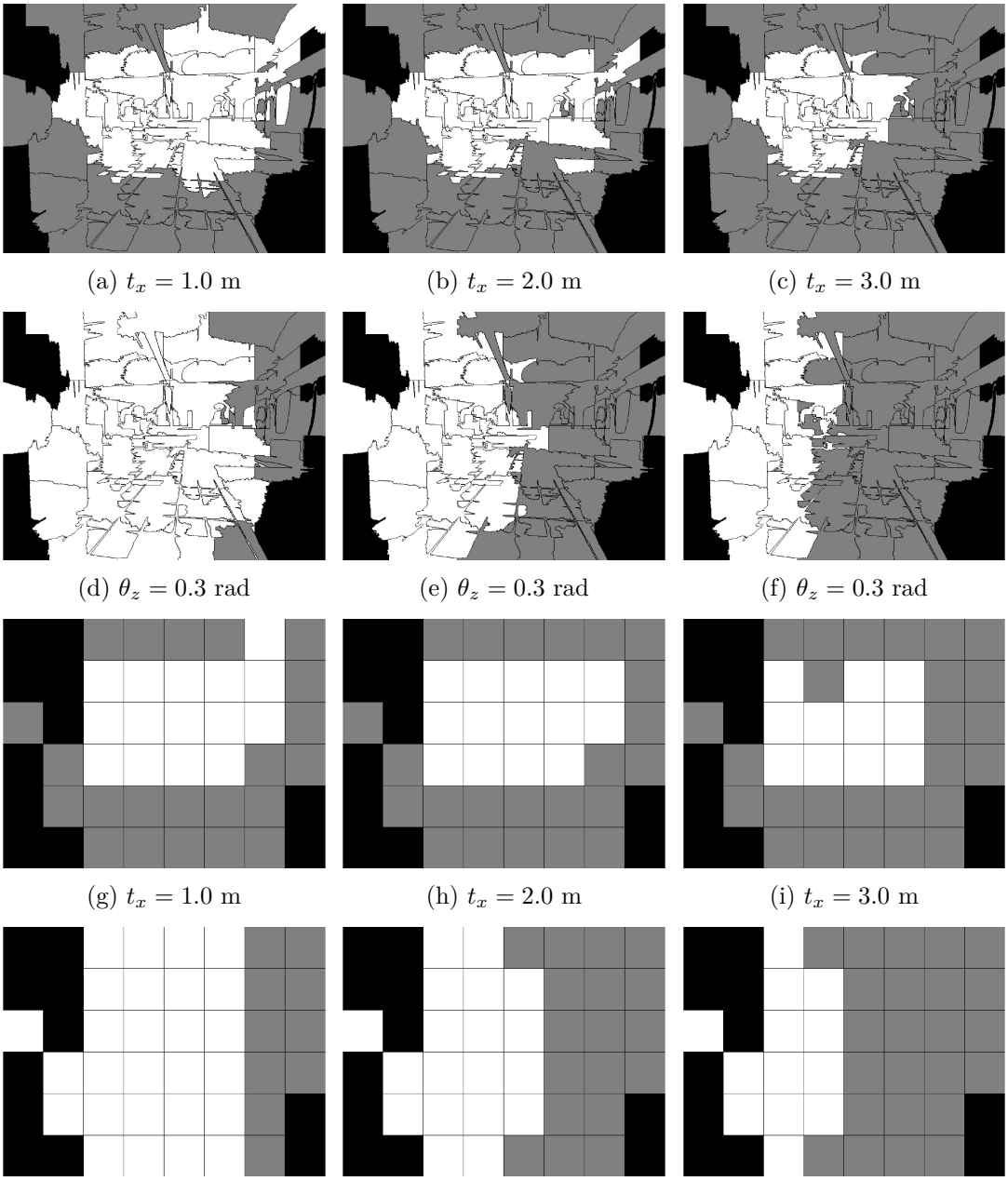
V.3.2.4 Prédiction de la visibilité

Pour chaque région où une profondeur a été estimée, on assigne la valeur obtenue au point médian. On obtient ainsi un point 3D. Ce point 3D est projeté sur le plan image futur, comme réalisé avec le premier critère (voir la section V.1.1.1). Ici, les incertitudes sur la position du point ne sont pas prises en compte.

Si le point est prédit visible dans l'image future, toute la région est supposée visible dans l'image future et donc tous les points qui se trouvent dedans également. Si le point est considéré non visible alors la région n'est pas considérée. Le critère final est le nombre total de points appartenant à des régions qui sont prédites visibles à la position future.

La figure V.16 montre le résultat de prédiction pour les deux méthodes pour plusieurs déplacements futurs. Plus le déplacement est important et plus le nombre de régions qui sont prédites non visibles est important. Dans le cas d'un mouvement de translation vers l'avant, ce sont les régions sur les bords de l'image qui disparaissent en premier. Dans le cas d'une rotation vers la gauche, ce sont les régions situées sur la droite de l'image qui sont considérées non visibles en premier.

Avec la méthode superpixels, la prédiction est plus précise car les superpixels sont définis comme des régions homogènes de l'image qui correspondent souvent à des régions planes de la scène. Au contraire, la méthode de division par rectangles donne un résultat moins précis car les rectangles découpent la scène en zones qui ne sont pas cohérentes avec les discontinuités de profondeur. Par exemple, sur les images de la figure V.13, on peut



(a) $t_x = 1.0$ m (b) $t_x = 2.0$ m (c) $t_x = 3.0$ m

(d) $\theta_z = 0.3$ rad (e) $\theta_z = 0.3$ rad (f) $\theta_z = 0.3$ rad

(g) $t_x = 1.0$ m (h) $t_x = 2.0$ m (i) $t_x = 3.0$ m

(j) $\theta_z = 0.3$ rad (k) $\theta_z = 0.3$ rad (l) $\theta_z = 0.3$ rad

FIGURE V.16 – Prédiction de la visibilté de chaque région pour différents mouvements prédits. Les régions noires sont les régions où aucune information de profondeur n'est connue, les régions blanches sont les régions qui sont prédites visibles après le déplacement et les régions grises sont les régions qui disparaissent du champ de vue.

observer un poteau sur la gauche. Tous les superpixels présentent un bord coïncidant avec le bord du poteau, ils correspondent soit à un morceau du poteau, soit à la scène derrière le poteau. Au contraire, on peut trouver dans le même rectangle, une partie du poteau et une partie de la scène derrière, qui ne sont pas à la même profondeur.

V.3.3 Validation expérimentale

Ce deuxième critère a été testé en simulation sur des données d'expériences. Seules les images provenant de ces jeux de données sont utilisées. Toute la chaîne de perception-commande est exécutée afin d'obtenir la commande optimale à chaque pas de temps. L'algorithme fonctionne en boucle ouverte, les commandes obtenues ne sont pas appliquées. Le but de cette partie est de comparer les commandes calculées suivant les différentes méthodes.

Les tests sont effectués avec le logiciel Matlab sur un ordinateur possédant un processeur Intel Xeon CPU W3530. Ce processeur est moins performant que ceux utilisés sur les robots, les algorithmes peuvent présenter des différences de temps d'exécution.

Dans cette partie, trois méthodes sont comparées : celle sur la visibilité des points de eVO présentée en début de chapitre (V.1.1) et les deux méthodes sur la division de l'image par régions, par rectangles ou par superpixels. Dans cette partie, on appellera méthode points 3D, la méthode qui prédit la visibilité des points 3D de eVO, méthode rectangles, la méthode de division de l'image par les rectangles et méthode superpixels, celle utilisant la division par superpixels.

Pendant ces tests, seul le coût sur la qualité de localisation J_{loc} est considéré dans la fonction de coût. Les valeurs des paramètres sont :

— $H_{\text{loc}} = H_p$

— $H_c = H_p$ if $H_p \leq 8$ and $H_c = 8$ if $H_p > 8$

Plusieurs horizons de prédiction de scène H_{loc} ont été testés afin de comparer les différences de comportement par rapport à la valeur de ce paramètre.

La vitesse linéaire est fixée à $v = 0.25$ m.s^{-1}. Seule la vitesse angulaire ω est optimisée. L'ensemble des valeurs possibles est borné par -0.4 et 0.4 rad.s^{-1} avec un pas de 0.1 rad.s^{-1}, ce qui donne une optimisation sur un ensemble de 9 valeurs. $\omega = 0$ correspond à une vitesse angulaire nulle, ce qui correspond à une trajectoire rectiligne. Si $\omega < 0$, la trajectoire présente une rotation vers la droite et si $\omega > 0$, la rotation est vers la gauche.

A chaque nouvelle paire d'images reçue, le coût de qualité de localisation J_{loc} est calculé pour chaque commande et pour les trois méthodes de prédiction de la scène future. On calcule ensuite la commande optimale pour chaque méthode, qui correspond à la commande qui minimise J_{loc}. Seule la vitesse angulaire optimale ω^* obtenue est donnée dans la suite, puisque la vitesse linéaire est fixe. Les résultats d'optimisation et les temps d'exécution de chaque méthode sont comparés.

La figure V.17 montre le temps de calcul pour chaque méthode à chaque pas de temps sur la trajectoire entière. Pour les méthodes points 3D et superpixels, le temps de calcul est toujours supérieur à celui de la méthode rectangles et il est très variable. De

plus, pour la méthode points 3D, les valeurs maximales sont très élevées. Au contraire, la méthode rectangles présente un temps de calcul très stable avec des valeurs maximales faibles.

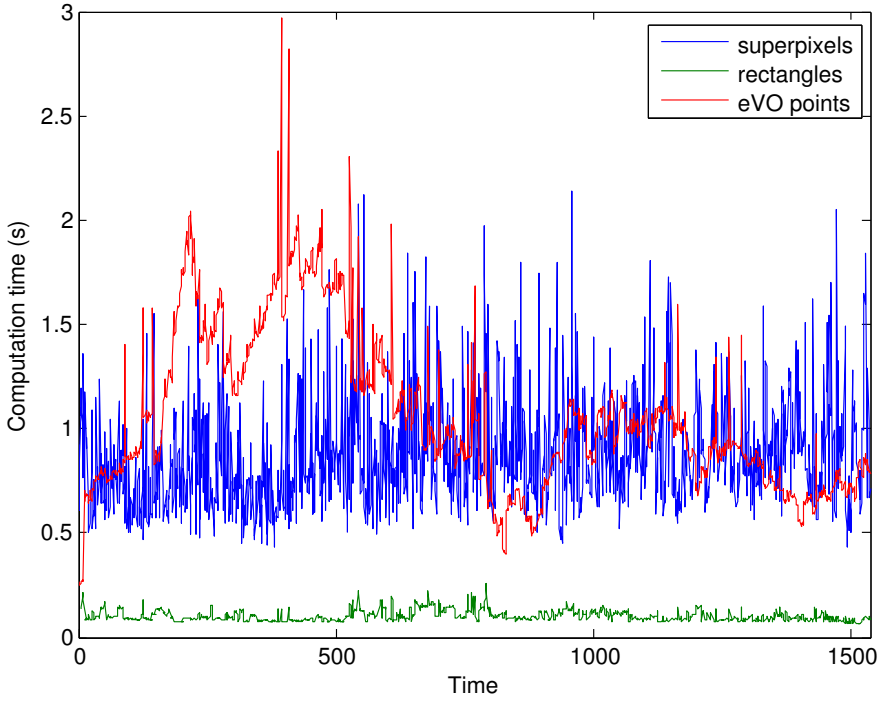


FIGURE V.17 – Temps de calcul de chaque méthode sur une trajectoire

Le temps de calcul a été mesuré sur plusieurs trajectoires avec des horizons de prédiction de scène différents. Des valeurs statistiques sont affichées dans le tableau V.1. La moyenne et l'écart-type sont plus élevés pour les méthodes points 3D et superpixels que la méthode rectangles. La valeur maximale moyenne est également plus importante pour ces deux méthodes. On retrouve le résultat observé sur une trajectoire dans la figure V.17. Le temps minimal moyen pour la méthode superpixels est 0.461 s, ce qui est supérieur à la valeur maximale moyenne de la méthode rectangles. Le temps de calcul pour la méthode superpixels est donc toujours supérieur à celui de la méthode rectangles, car la division de l'image en superpixels nécessite un temps de calcul important qui domine le coût des autres étapes.

Ces résultats montrent que la méthode rectangles améliore de manière significative le temps de calcul comparé à la méthode points 3D. De plus, le temps de calcul obtenu est très stable, ce qui permet de facilement prévoir le temps de calcul total de l'optimisation et donc de définir plus de trajectoires à tester dans la commande MPC. Par contre, la méthode superpixels n'améliore pas le temps de calcul mais présente tout de même des valeurs maximales inférieures à celles de la méthode point 3D.

Nous passons maintenant à une comparaison des trois méthodes en termes de com-

	Rectangles	Superpixels	Points 3D
Moyenne	0.104	0.893	0.888
Ecart-type	0.038	0.252	0.343
Maximum	0.277	2.158	2.472
Minimum	0.057	0.461	0.216

TABLE V.1 – Comparaison du temps de calcul (en secondes) pour les trois méthodes, valeurs moyennes sur plusieurs trajectoires.

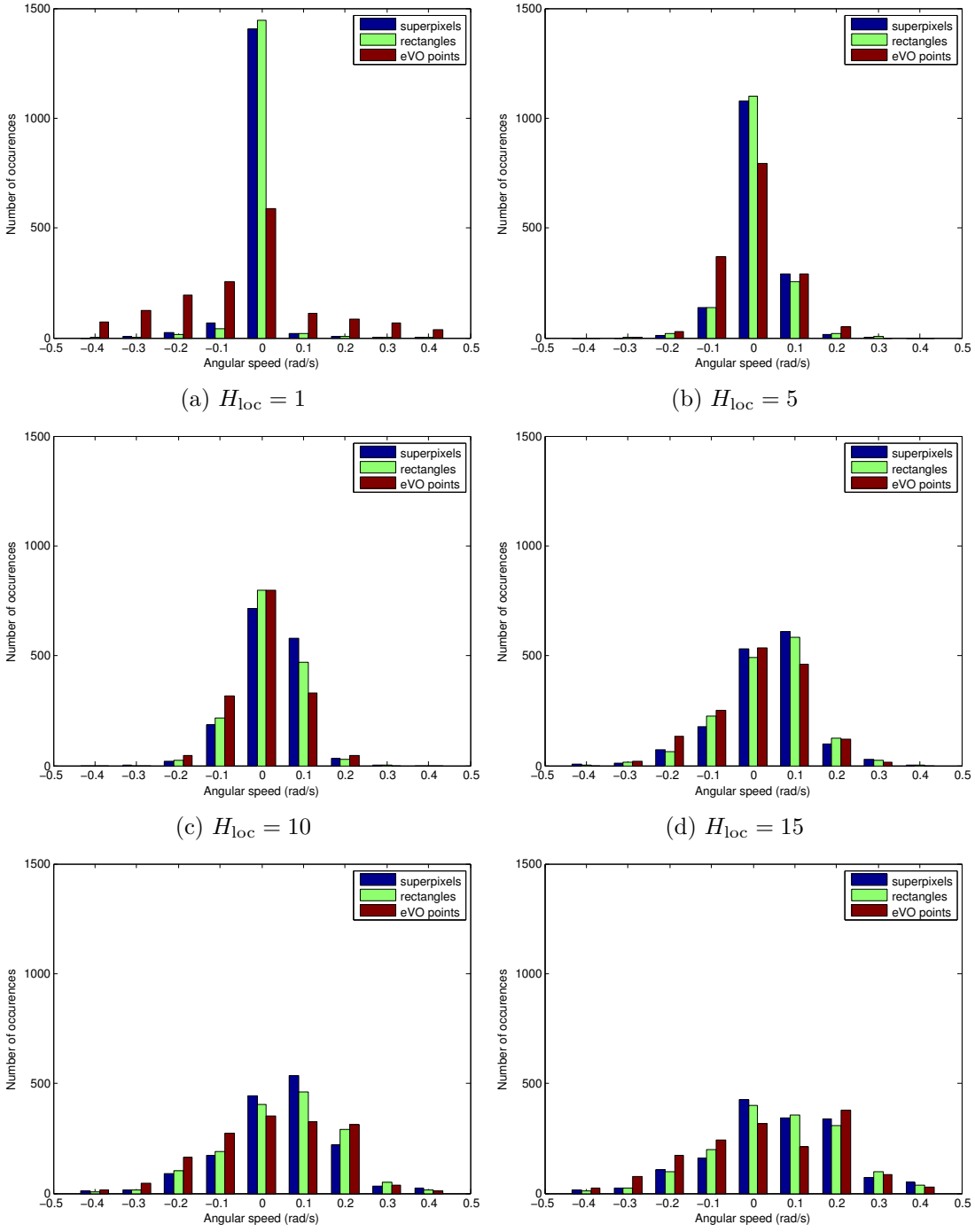
mande sélectionnée au final. La figure V.18 montre le nombre d'apparition de chaque valeur de commande optimale ω^*, pour les trois méthodes et pour différents horizons de prédiction.

Si $H_{\text{loc}} = 1$, on peut remarquer que les méthodes rectangles et superpixels donnent une commande optimale $\omega^* = 0$ sur presque toute la trajectoire (figure V.18a). En effet, quand l'horizon de prédiction de scène est faible, le déplacement entre la position actuelle et la position future est petit. Pour une trajectoire rectiligne ($\omega = 0$), la part de l'image qui disparait après le mouvement est donc très limitée. Comme les points médians ne sont pas près des bords de l'image, ils sont très majoritairement prédits visibles à la pose future. Par conséquent, presque toutes les régions sont prédites visibles et le coût J_{loc} est faible. Pour un mouvement de rotation ($\omega \neq 0$), les points médians des régions situées au bord de l'image opposé à la direction de rotation disparaissent. Ces régions sont donc prédites non visibles après le mouvement et le coût J_{loc} est donc plus important. La commande optimale est donc dans la plupart des cas la commande avec la vitesse angulaire $\omega = 0$, qui donne une trajectoire rectiligne.

Au contraire, les points 3D extraits par l'algorithme d'odométrie visuelle peuvent être extraits n'importe où sur l'image et donc potentiellement près des bords. Il est donc possible que plus de points disparaissent avec un mouvement rectiligne qu'un mouvement de rotation. Ce qui explique pourquoi les commandes optimales sont moins concentrées sur la valeur $\omega^* = 0$ avec la méthode points 3D.

Quand la valeur de H_{loc} augmente, les deux méthodes de division de l'image donnent des commandes optimales sur un ensemble de valeurs plus large. On observe que les résultats deviennent similaires à ceux obtenus avec la méthode points 3D.

En comparant les commandes optimales obtenues à chaque pas de temps, on observe que la différence entre la commande calculée par la méthode rectangles et celle calculée par la méthode points 3D est faible, dans la plupart des cas elle est inférieure ou égale à 0.1 rad.s^{-1}, comme le montre la figure V.19. Sur ce graphique, si on ne prend pas en compte le cas $H_{\text{loc}} = 1$, on peut observer que la différence entre la commande calculée par la méthode rectangles et la méthode points 3D est de 0 rad.s^{-1} dans 60 % des cas. Ce pourcentage est de 30 % pour une différence de 0.1 rad.s^{-1}. Ainsi, dans plus de 90 % des cas, la différence est inférieure ou égale à 0.1 rad.s^{-1}. Les résultats sont similaires quand on compare la méthode superpixels et la méthode points 3D. Ces résultats montrent que les trois méthodes donnent des performances comparables quand H_{loc} est supérieur à 5.



(a) $H_{\text{loc}} = 1$ (b) $H_{\text{loc}} = 5$

(c) $H_{\text{loc}} = 10$ (d) $H_{\text{loc}} = 15$

(e) $H_{\text{loc}} = 20$ (f) $H_{\text{loc}} = 25$

FIGURE V.18 – Histogrammes des commandes optimales pour différents horizons H_{loc} sur une trajectoire

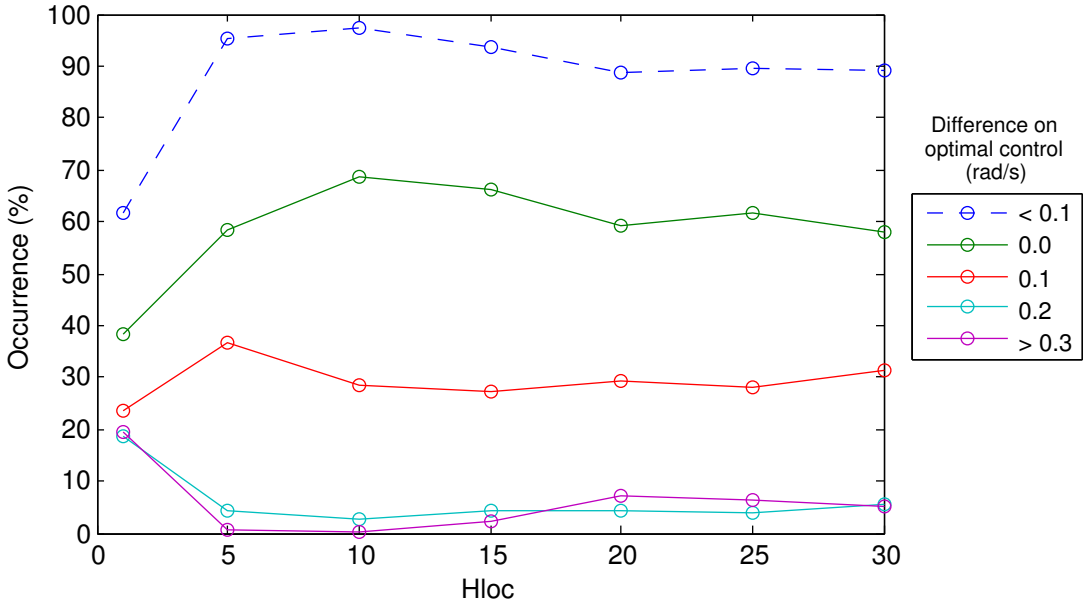


FIGURE V.19 – Proportion des différences sur les valeurs de commande optimale obtenue par les méthodes rectangles et points 3D pour différents horizons de prédiction de scène H_{loc}

En conclusion, ces tests ont montré que la méthode rectangles améliore significativement le temps de calcul par rapport à la première méthode proposée, cherchant à prédire la visibilité des points de eVO, tout en gardant des performances similaires. Le temps de calcul est plus faible et constant. La méthode superpixels donne des performances similaires également, mais ne permet pas d'améliorer le temps de calcul.

Ce gain en temps de calcul donnerait la possibilité de définir un ensemble plus grand de commandes en entrée de l'algorithme MPC, tout en gardant la même période d'échantillonnage t_e. Ce qui signifie que plus de trajectoires pourraient être évaluées par l'algorithme et ce qui permettrait d'améliorer la précision dans la commande du robot.

Cependant, cette nouvelle méthode utilise toujours l'information 3D obtenue à partir d'une triangulation de points extraits dans les images provenant du système stéréo, ce qui limite intrinsèquement la portée et l'angle de l'information utilisable. De plus, pour les trois méthodes, le critère mesure la perte de points après le déplacement par rapport au nombre de points déjà connus. Ces méthodes ne cherchent pas à prédire ce qui pourrait être vu dans les images futures et qui n'a pas encore été perçu dans les images déjà reçues. La possibilité d'extraire de l'information dans les parties de l'image qui seront vues seulement après le déplacement n'est pas prise en compte. Ce phénomène est encore plus important avec les mouvements de rotation car la part de l'image future non vue est plus importante. Dans le cas de rotation importante, il peut même n'y avoir pas d'intersection entre la scène perçue à l'instant actuel et la scène future. Il est donc impossible dans ce cas de faire une prédiction de la qualité de localisation future. Cette

problématique n'a pas été considérée dans les travaux de cette thèse. Elle nécessite d'avoir un a priori sur l'environnement ce qui n'est pas le cas ici car l'environnement est supposé inconnu au début de la mission.

V.4 Conclusion

Dans ce chapitre, nous avons mis en place une première méthode afin d'évaluer la qualité d'une scène future pour la localisation visuelle. Elle consiste à prédire la visibilité future des points 3D issus de l'algorithme d'odométrie visuelle. Cette méthode a été validée par des expériences puis intégrée à une boucle de commande afin de réaliser des missions de ralliement de points de passage dans des environnements pouvant présenter des zones peu texturées et où le calcul de la localisation visuelle risque d'échouer. Des expériences en situation réelle ont été réalisées et ont permis de démontrer que l'ajout du critère sur la qualité de localisation permet d'améliorer le comportement du robot face aux zones peu texturées.

Une deuxième méthode d'évaluation de la qualité d'une scène future a été proposée afin de réduire le coût de calcul, qui est trop élevé et variable pour la première méthode. Cette nouvelle méthode est basée sur une approche de division de l'image par régions. Deux types de divisions ont été testés, par superpixels ou par rectangles. Les expériences ont montré que la méthode de division de l'image par des rectangles permet de diminuer fortement le temps d'exécution tout en gardant des performances similaires.

Les expériences proposées dans ce chapitre sont basées sur des scénarios simples de ralliement de points de passage. Dans la suite des travaux, des missions avec des objectifs plus complexes ont été considérées. Le but est de reprendre les expériences d'exploration autonome réalisées au chapitre IV, sans utiliser la fusion de données multi-capteurs et donc en intégrant le critère sur la qualité d'une scène pour la localisation. Un deuxième type de mission considéré est la réalisation de missions de ralliement de points de passage avec évitement des obstacles mobiles. Ces différentes expériences font l'objet du chapitre VI.

Chapitre VI

Expérimentations avancées avec couplage vision/commande

Sommaire

Dans ce chapitre, l'objectif est de réaliser des missions de navigation complexe sur des plateformes robotiques équipées uniquement de caméras pour percevoir l'environnement. Deux types de missions sont considérées.

Le premier type de mission considéré est l'exploration autonome dans des environnements inconnus et encombrés, comme réalisé dans le chapitre IV. La pièce explorée peut présenter des zones peu texturées où le calcul de la localisation visuelle est susceptible d'échouer. Le but est de prendre en compte le critère sur la qualité d'une scène future, développé dans le chapitre V afin de s'assurer que la localisation visuelle reste correcte. Ces expériences sont décrites dans la partie VI.1.

Le deuxième type de mission considéré est le ralliement de points de passage avec détection et évitement d'obstacles mobiles. La détection des obstacles ainsi que la localisation sont assurées uniquement à partir des informations visuelles embarquées. La section VI.2 décrit ces expériences.

VI.1 Exploration autonome avec prise en compte de la qualité de localisation

Le but de cette partie est de réaliser des missions d'exploration autonome en environnement inconnu et encombré. Contrairement aux expériences réalisées au chapitre IV, on considère dans cette partie que l'unique capteur disponible sur le robot est son banc stéréo. Il n'est donc pas possible d'effectuer une fusion de données multi-capteurs comme dans les expériences décrites au chapitre IV, la localisation est uniquement calculée à partir des informations visuelles.

Pour garantir que la localisation visuelle reste précise, il est donc nécessaire de prendre en compte la prédiction de la qualité de localisation future dans la commande. Dans la littérature, la qualité de la localisation future n'a été prise en compte que dans le cadre de missions de navigation par points de passage [Mostegel et al., 2014, Sadat et al., 2014]. Les références portant sur des missions d'exploration, décrites dans la partie III.3.2.3, s'intéressent uniquement à la réduction active des incertitudes et à la résolution du compromis entre exploration de zones inconnues et retour sur des zones connues pour diminuer les incertitudes sur la localisation du robot [Kim and Eustice, 2013, Bryson and Sukkarieh, 2008].

Pour assurer que le robot choisisse une trajectoire assurant un calcul précis de la localisation, on intègre donc le critère sur la qualité de localisation visuelle développé au chapitre V. On a utilisé le critère par point, cherchant à prédire la visibilité des amers issus de l'algorithme d'odométrie visuelle, décrit dans la partie V.1. Notons qu'on a montré dans le chapitre V que le deuxième critère développé, le critère par région, basé sur une prédiction de la visibilité de régions de l'image, est plus performant en termes de temps de calcul. Cependant, les expériences décrites dans cette partie ont été réalisées avec le critère par point car le module utilisant ce critère avait déjà été développé et embarqué sur le robot pour les expériences de ralliement de points, décrites dans la partie V.2. Il était donc plus simple de réaliser une première série d'expériences en utilisant les algorithmes déjà embarqués.

L'architecture du système mise en place pour la réalisation de ces expériences est décrite en section VI.1.1. La mise en œuvre des expérimentations est expliquée dans la section VI.1.2. Les résultats des expériences sont décrits dans la section VI.1.3.

VI.1.1 Architecture du système

L'architecture du système mise en place pour réaliser les expériences d'exploration avec prise en compte de la qualité de la localisation est représentée sur la figure VI.1.

La reconstruction de l'environnement est réalisée à partir des données envoyées par le capteur de profondeur, ce module est décrit dans la section IV.2. La localisation est calculée uniquement par l'algorithme d'odométrie visuelle eVO (voir en section III.1.4) à partir des images reçues par le banc stéréo. Le module de prédiction de scène est identique à celui mis en place pour les expériences de ralliement de points, décrites dans la partie V.2. Il prend en entrée la liste des amers 3D envoyée par l'algorithme

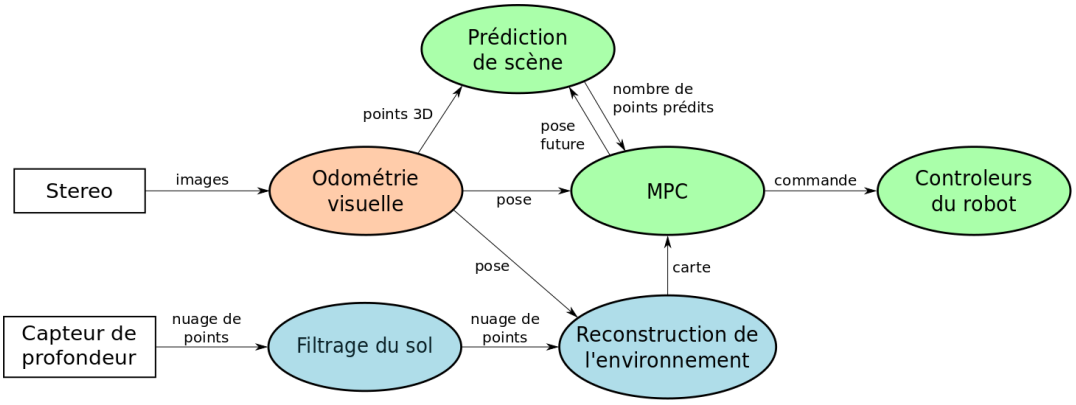


FIGURE VI.1 – Architecture du système pour l'exploration autonome avec considération de la qualité de localisation future

d'odométrie visuelle et la pose future envoyée par le module MPC et renvoie le nombre d'amers prédits comme étant visibles à la position future.

Le module MPC utilise une nouvelle fonction de coût, qui comprend le coût d'exploration, décrit en partie IV.3.3.4, le coût d'obstacle, décrit en partie IV.3.3.3 et le coût de qualité de localisation, décrit en partie V.2.1. Sa formulation est :

$$J = w_{\text{expl}} J_{\text{expl}} + w_{\text{obs}} J_{\text{obs}} + w_{\text{loc}} J_{\text{loc}} \tag{VI.1}$$

La figure VI.2 illustre le fonctionnement de la boucle de commande avec cette nouvelle fonction de coût. Le robot cherche simultanément à explorer les zones inconnues (en blanc), éviter les obstacles (en rouge) et garder des amers 3D dans son champ de vue (points bleus). La trajectoire optimale est la trajectoire qui répond au mieux simultanément à ces trois objectifs.

VI.1.2 Mise en œuvre des expérimentations

Dans ces expériences, le robot doit explorer une pièce inconnue et encombrée qui possède un mur sans texture, comme on peut le voir dans la figure VI.3. La localisation du robot est donnée uniquement par l'algorithme d'odométrie visuelle à partir des images reçues par le banc stéréo. On veut vérifier si l'ajout du critère dans la boucle de commande permet de mener les missions à terme et de garder une localisation précise tout au long du déplacement du robot dans la pièce.

La pièce mesure environ 11x11 m^2. Deux obstacles sont disposés dans la pièce et resteront à la même place pour toutes les expériences réalisées. La pièce est également bordée d'obstacles.

Le robot utilisé est le Turtlebot, présenté en section II.2.2. Le processeur installé sur le robot n'étant pas assez puissant pour embarquer tous les algorithmes, les modules de

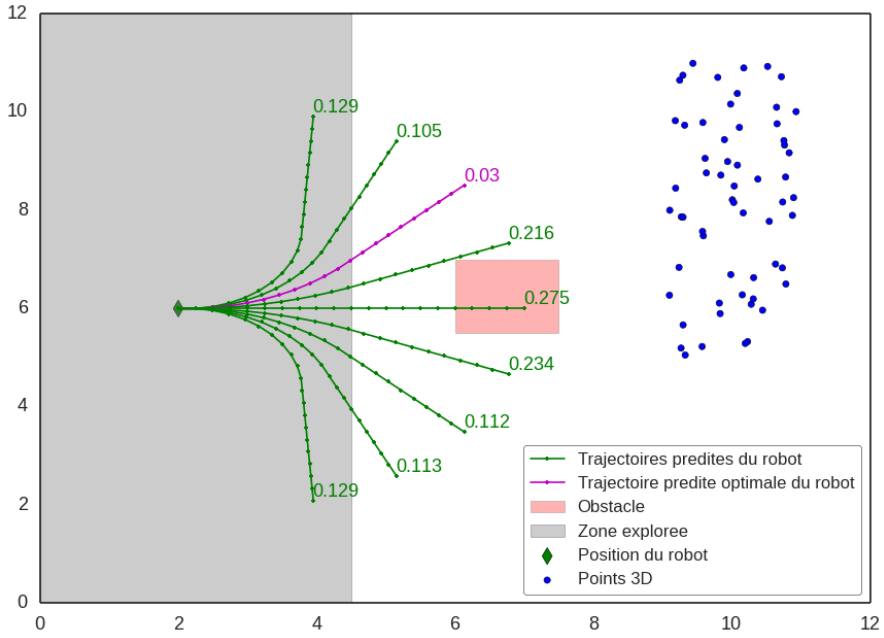


FIGURE VI.2 – Calcul de la trajectoire optimale pour une mission d'exploration autonome avec évitement d'obstacles et considération de la qualité de la localisation future

commande prédictive et de prédiction de la qualité visuelle sont exécutés sur une autre station (voir section II.2.2).

La vérité terrain sur la localisation est obtenue grâce à un système de capture de mouvement, qui permet d'obtenir une position et une orientation très précises du robot, l'erreur en position est inférieure à 0.3 mm et l'erreur angulaire est inférieure à 0.05°. Il est ainsi possible de comparer la trajectoire estimée par l'algorithme d'odométrie visuelle avec la vraie trajectoire et de calculer l'erreur sur la localisation visuelle.

Les valeurs des paramètres relatifs au critère de qualité de localisation sont ceux utilisés dans la partie V.1.2.

La portée du capteur de profondeur est limitée à $\delta = 2.5$ m. Ce capteur a une portée plus importante normalement, mais on limite volontairement la portée pour limiter la capacité d'exploration du capteur et ainsi augmenter le temps de déroulement de la mission. La résolution de la carte est $r = 0.2$ m/pixel.

Pour le module MPC, les paramètres sont les suivants :

— Horizons : $H_p = 20$, $H_c = 8$

— Distance aux obstacles : $d_{sec} = 0.6$ m, $d_{des} = 1.0$ m

— Limites de vitesse $v_{max} = 0.25$ m.s^{-1}, $\omega_{max} = 0.4$ rad.s^{-1}

— Poids initiaux sur les coûts d'exploration et d'obstacle : $w^0_{\text{expl}} = 0.3$, $w^0_{\text{obs}} = 0.7$

L'horizon de prédiction a une valeur supérieure à celle utilisée dans les expériences de ralliement de points car il est nécessaire d'avoir une prédiction à plus long terme pour

FIGURE VI.3 – Environnement dans lequel se déroule la mission d'exploration, le mur peu texturé est derrière le robot

un objectif d'exploration. Les distances aux obstacles ont été choisies en fonction de la taille du robot, la distance de sécurité est le double du diamètre du robot. Les vitesses maximales sont également fixées en fonction des caractéristiques du robot.

Le réglage des trois pondérations dans la fonction de coût (VI.1) est plus complexe que le réglage de deux pondérations, réalisé par essais et erreurs dans les expériences précédentes. On a choisi de fixer les pondérations sur le coût d'exploration w_{expl} et sur le coût d'obstacle w_{obs}, en choisissant les valeurs utilisées dans les expériences d'exploration réalisées dans le chapitre IV. Ces valeurs ont permis d'obtenir un résultat d'exploration correct. Seule la valeur de w_{loc} va donc être modifiée pendant les expériences. On veut conserver le même rapport entre les deux pondérations w_{expl} et w_{obs} pour garder la même importance relative entre les deux objectifs correspondants.

La somme des trois pondérations est fixée à 1. Les pondérations w_{expl} et w_{obs} sont modifiées afin de respecter cette contrainte quand la valeur du poids sur la qualité de localisation w_{loc} est non nul. Les nouvelles valeurs de w_{expl} et w_{obs} sont calculées de telle sorte que le rapport entre ces deux pondérations soit préservé.

Si $w_{\text{loc}} = 0.0$, on a :

$$w^0_{\text{expl}} + w^0_{\text{obs}} = 1 \tag{VI.2}$$

Quand $w_{\text{loc}} > 0$, w_{expl} et w_{obs} sont recalculés pour respecter la contrainte :

$$w_{\text{expl}} + w_{\text{obs}} + w_{\text{loc}} = 1 \tag{VI.3}$$

Les nouvelles valeurs de w_{expl} et w_{obs} sont obtenues par :

$$w_{\text{expl}} = \frac{w^0_{\text{expl}}}{w^0_{\text{expl}} + w^0_{\text{obs}} + w_{\text{loc}}} \tag{VI.4}$$

$$w_{\text{obs}} = \frac{w^0_{\text{obs}}}{w^0_{\text{expl}} + w^0_{\text{obs}} + w_{\text{loc}}} \tag{VI.5}$$

Le rapport entre w_{expl} et w_{obs} est inchangé :

$$\frac{w_{\text{expl}}}{w_{\text{obs}}} = \frac{w^0_{\text{expl}}}{w^0_{\text{obs}}} \tag{VI.6}$$

L'optimisation de la fonction de coût (VI.1) est réalisée en discrétisant l'espace des commandes. Un ensemble de 10 commandes est défini, il contient la commande nulle $U = (v = 0, \omega = 0)^T$ et 9 commandes $U = (v = v_{max}, \omega)^T$ avec les vitesses angulaires ω définies uniformément sur l'intervalle $[-\omega_{max}, \omega_{max}]$.

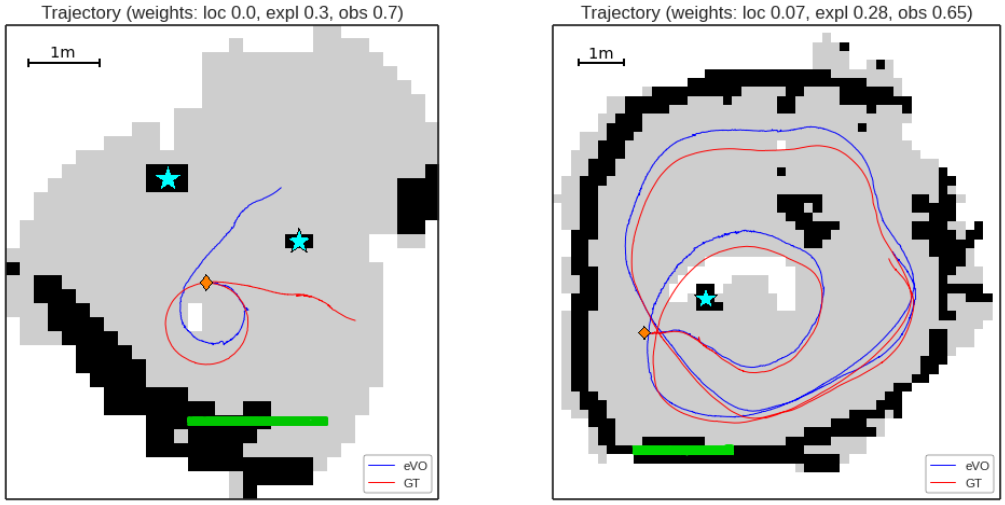
VI.1.3 Résultats

Deux résultats d'expériences sont affichés sur la figure VI.4. Pour la première figure, le poids w_{loc} a été fixé à zéro. Dans la seconde figure, le poids a été fixé à 0.07 avec un horizon de prédiction de scène $H_{\text{loc}} = 1$. Le mur sans texture est à la droite du robot au début de la mission d'exploration. Dans le premier cas, le robot tourne directement sur ce mur, parce qu'il cherche à maximiser la zone explorée et éviter l'obstacle sur sa gauche. Cette trajectoire est donc la trajectoire optimale à suivre. On peut remarquer sur la figure que la trajectoire calculée par l'algorithme d'odométrie visuelle dérive fortement par rapport à la vérité terrain quand le robot approche du mur. A cause de l'erreur sur la localisation, la reconstruction de l'environnement est mauvaise, le même obstacle a été détecté à deux endroits différents.

Dans la seconde expérience, le robot commence la mission à la même position. Il préfère suivre une trajectoire rectiligne, pour éviter l'obstacle et ne pas se retrouver face au mur peu texturé. L'erreur sur la localisation reste faible tout au long de la trajectoire effectuée et le robot parvient à achever sa mission d'exploration, la pièce est presque entièrement explorée.

Ce premier résultat montre que l'ajout du critère sur la qualité de localisation dans la stratégie de commande permet au robot de choisir une trajectoire qui lui évite de se retrouver face au mur et donc permet d'éviter les erreurs dans le calcul de la localisation visuelle.

Dans une deuxième série d'expérimentations, 15 essais ont été réalisés avec différents horizons de prédiction de scène H_{loc} et différents poids w_{loc}. Le robot commence toujours au même point de départ et l'environnement n'est pas modifié entre les différents essais. Le but de ces expériences est de confirmer le premier résultat obtenu et de chercher le meilleur paramétrage pour H_{loc} et w_{loc} dans le but d'obtenir un résultat d'exploration complet tout en gardant la localisation la plus précise possible. Cinq expériences ont



(a) $w_{\text{loc}} = 0$ (b) $w_{\text{loc}} = 0.07$

FIGURE VI.4 – Deux résultats de missions d'exploration. La courbe bleue est la trajectoire calculée par l'algorithme d'odométrie visuelle et la courbe rouge est la trajectoire obtenue par le système de capture de mouvement. La zone explorée est en gris et les obstacles sont en noir. Le diamant orange est le point de départ et la ligne verte représente le mur peu texturé. L'étoile bleue représente le même obstacle, vu à deux endroits différents dans la figure (a).

été réalisées avec $w_{\text{loc}} = 0.0$. Certaines de ces expériences ont dû être arrêtées par l'utilisateur car la localisation visuelle a dérivé de façon trop importante pour permettre une navigation sûre dans l'environnement. Dans les dix autres expériences, w_{loc} est supérieur à zéro.

La comparaison des résultats est effectuée sur deux critères : la réalisation de l'objectif d'exploration en étudiant la couverture de la zone et la précision de la localisation, en étudiant l'erreur entre la trajectoire calculée par l'odométrie visuelle et celle calculée par le système de capture de mouvement.

La figure VI.5 montre les résultats obtenus sur la couverture de zone. Elle est exprimée sous la forme du pourcentage de la surface explorée par rapport à la surface totale. La surface totale est la surface définie par les limites données à la carte d'exploration au début de la mission. Cette surface, définie comme un rectangle, est plus grande que la zone atteignable par le robot car les obstacles au bord de la pièce empêchent le robot d'explorer les coins, comme on peut le constater dans la figure VI.4b. C'est pourquoi la couverture de zone ne dépasse jamais 80 %. On peut considérer qu'au dessus de 70 %, la zone est globalement explorée.

Sur cette figure, on peut constater que l'exploration a échoué dans les cas où $w_{\text{loc}} = 0.0$, la couverture de zone est inférieure à 30 %. Dans certains cas, la mission a dû être arrêtée prématurément à cause des erreurs dans la localisation, ce qui n'a pas permis

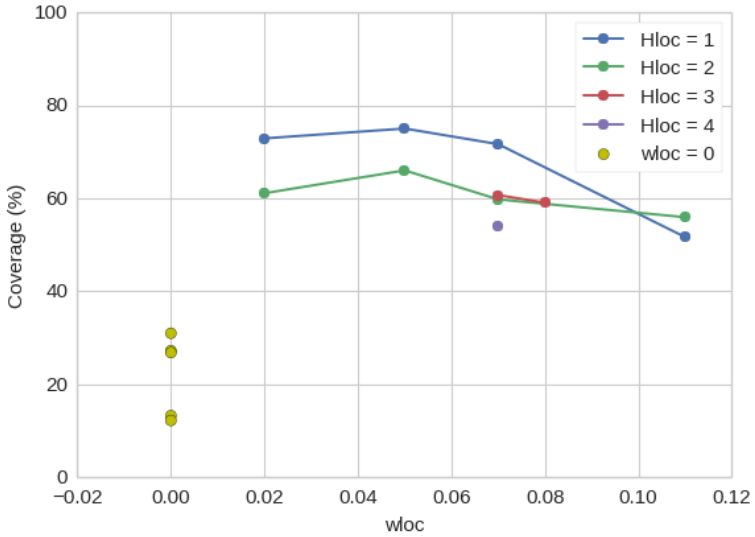


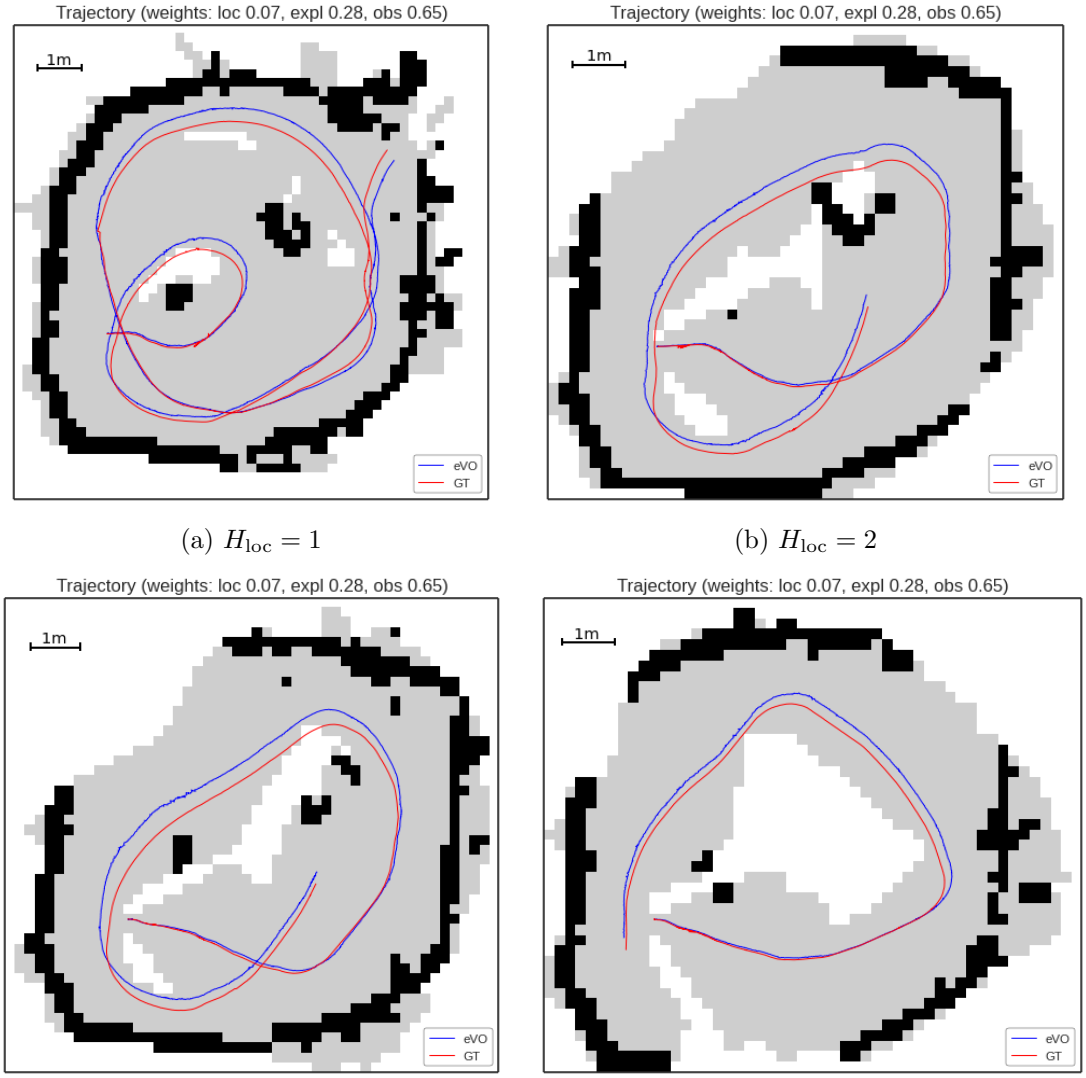
FIGURE VI.5 – Couverture de la zone pour différents horizons H_{loc} et différents poids w_{loc}, en pourcentage de la zone totale à explorer

de réaliser l'objectif d'exploration. Dans les expériences avec w_{loc} supérieur à 0.0, la couverture de la zone est comprise entre 50 % and 80 %.

La couverture maximale est obtenue avec un horizon de prédiction de scène $H_{\text{loc}} = 1$ et diminue quand H_{loc} augmente, pour un poids w_{loc} fixe. La figure VI.6 montre la carte d'exploration obtenue lors de la réalisation de quatre missions d'exploration avec $w_{\text{loc}} = 0.07$ et H_{loc} compris entre 1 et 4.

On peut constater que la trajectoire réalisée par le robot est de plus en plus rectiligne à mesure que l'horizon H_{loc} augmente et que le robot explore de moins en moins les coins de la pièce. Ce phénomène peut être expliqué en examinant la définition du critère de prédiction de la scène. Ce critère utilise les amers 3D déjà vus et triangulés par l'algorithme d'odométrie visuelle. Ces amers sont donc obligatoirement situés en face des caméras. Le nombre d'amers qui sortent du champ de vision après le déplacement est donc plus important avec un mouvement de rotation qu'un mouvement de translation. Par conséquent, le coût J_{loc} est plus faible pour une trajectoire rectiligne que pour une trajectoire avec rotation. Ce phénomène est de plus en plus important avec l'augmentation de H_{loc}. Le coût de qualité de localisation devient alors très pénalisant pour les trajectoires avec rotation importante, c'est pourquoi le robot choisit majoritairement des trajectoires rectilignes. Dans le cas $H_{\text{loc}} = 4$, on peut voir que le robot tourne uniquement quand il s'approche d'obstacles, car dans ce cas, les trajectoires rectilignes sont pénalisées par le coût d'obstacle.

L'objectif d'exploration de zones inconnues, où par définition aucun amer 3D n'est connu, est antagoniste à l'objectif de conservation d'une localisation précise, qui consiste à rester dans des zones connues où les amers 3D sont nombreux. On veut que le robot ait la possibilité d'effectuer des rotations pour remplir son objectif d'exploration et par-



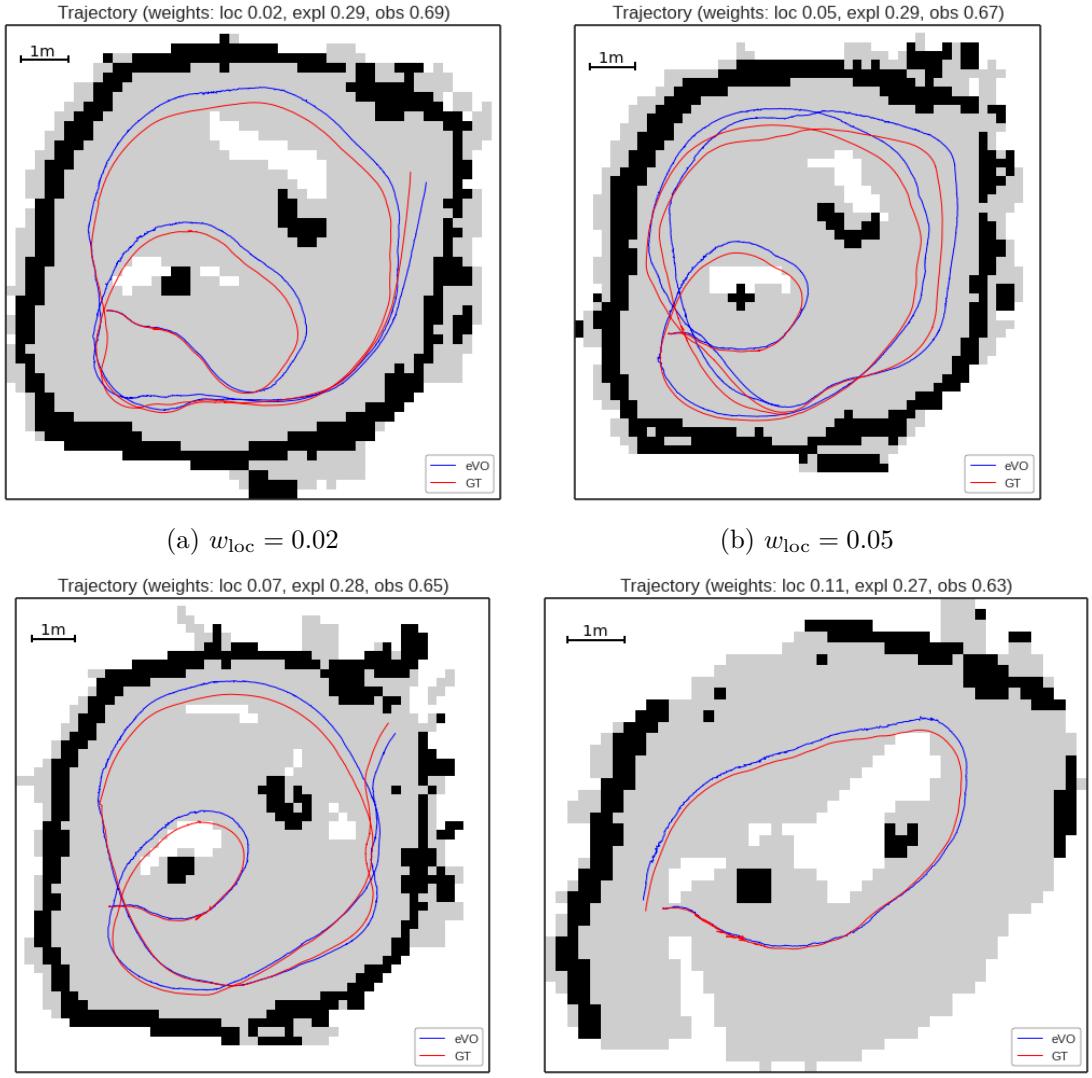
(a) $H_{\text{loc}} = 1$ (b) $H_{\text{loc}} = 2$

(c) $H_{\text{loc}} = 3$ (d) $H_{\text{loc}} = 4$

FIGURE VI.6 – Résultat de quatre expériences avec différentes valeurs sur H_{loc} et $w_{\text{loc}} = 0.07$

ticulièrement aller explorer les coins de la pièce. $H_{\text{loc}} = 1$ est donc la valeur optimale à fixer pour remplir l'objectif d'exploration.

Dans la figure montrant les résultats de couverture de zone VI.5, pour un horizon fixé H_{loc}, on peut constater que la couverture de zone diminue quand le poids sur le coût de qualité de localisation w_{loc} augmente. En effet, quand la valeur de ce poids augmente, l'importance de l'objectif d'exploration est réduit par rapport à l'importance de l'objectif de qualité de localisation. Le robot préfère limiter le nombre d'amers sortants du champ de vue après le mouvement plutôt qu'explorer des zones inconnues. La figure VI.7 montre



(a) $w_{\text{loc}} = 0.02$ (b) $w_{\text{loc}} = 0.05$

(c) $w_{\text{loc}} = 0.07$ (d) $w_{\text{loc}} = 0.11$

FIGURE VI.7 – Résultat de quatre expériences avec différentes valeurs sur w_{loc} et $H_{\text{loc}} = 1$

les cartes d'exploration obtenues lors de la réalisation de quatre expériences avec $H_{\text{loc}} = 1$ et avec quatre valeurs différentes sur le poids w_{loc}. Quand w_{loc} est compris entre 0.02 et 0.07 (figure VI.7a, VI.7b et VI.7c), la mission d'exploration est réussie et la localisation visuelle reste précise. Si le poids w_{loc} est plus grand (figure VI.7d), l'objectif d'exploration n'est pas atteint car le poids sur le coût de qualité de localisation est trop élevé par rapport au poids sur le coût d'exploration, le robot est moins attiré par les zones inconnues et plus par les zones connues et texturées, dans lesquelles de nombreux amers 3D sont connus.

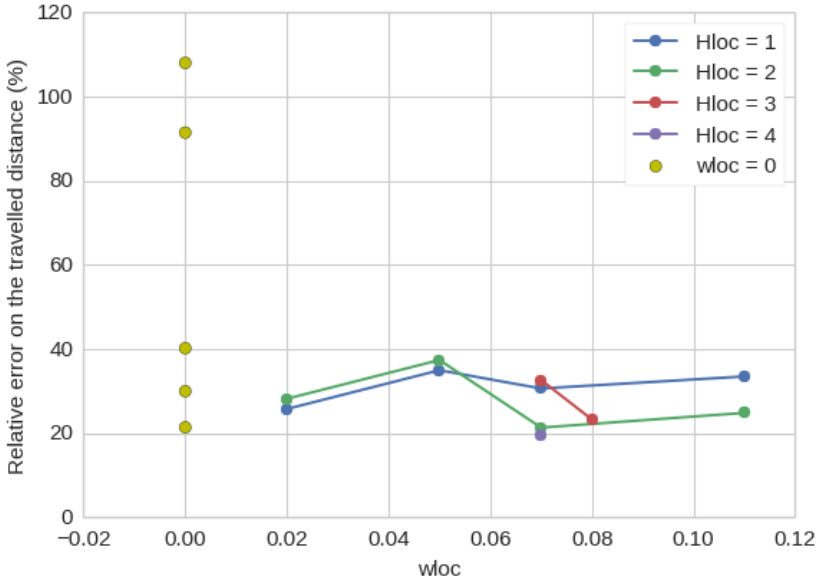


FIGURE VI.8 – Erreur relative entre la distance parcourue calculée par l'algorithme d'odométrie visuelle et la vérité terrain, pour plusieurs horizons H_{loc} et différents poids w_{loc}, en pourcentage de la distance parcourue totale

Pour évaluer l'erreur sur la localisation visuelle, on calcule l'erreur relative entre la distance parcourue estimée par l'odométrie visuelle et celle estimée par le système de capture de mouvement. L'erreur est affichée sur la figure VI.8 pour chaque expérience réalisée. Deux expériences avec $w_{\text{loc}} = 0.0$ présentent une erreur élevée (90 % et 110 %). Pour toutes les autres expériences, l'erreur est bornée, avec un maximum de 40 %. Ces résultats montrent que la localisation visuelle présente une dérive dans tous les cas, mais que celle-ci est plus limitée quand on ajoute le coût sur la qualité de localisation dans la commande.

La figure VI.9 montre la distance maximale parcourue sur la trajectoire entre deux pas de temps de l'algorithme d'odométrie visuelle. La période de fonctionnement de l'algorithme est d'environ 0.20 s et la vitesse linéaire maximale est de 0.25 m.s^{-1}. La distance maximale que le robot peut parcourir entre deux pas de temps est donc d'environ 0.05 m. Elle peut être légèrement supérieure car la période d'acquisition des images présente des petites variations. Avec $w_{\text{loc}} = 0.0$, la distance maximale est, dans trois cas, largement supérieure à cette limite, ce qui démontre une erreur importante dans l'estimation de la localisation par l'algorithme d'odométrie visuelle. Dans les autres cas, cette erreur est inférieure ou proche de la limite, il n'y a donc pas eu de dérive importante dans le calcul de l'odométrie visuelle.

Ces résultats démontrent que l'ajout du critère sur la qualité de localisation dans la boucle de commande permet au robot d'éviter de se diriger vers les zones où le calcul de la localisation visuelle risque d'échouer. De plus, pour obtenir les meilleures performances

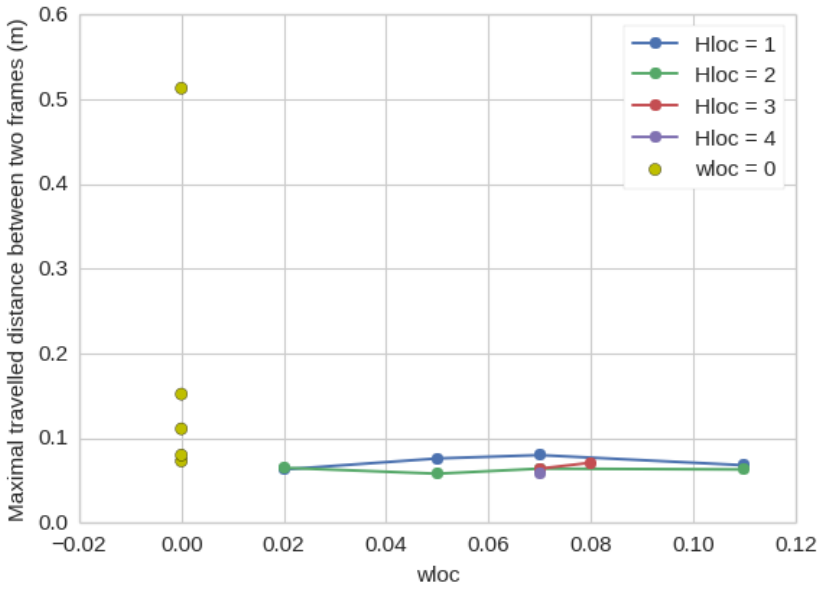


FIGURE VI.9 – Distance parcourue maximale calculée par l'algorithme d'odométrie visuelle entre deux pas de temps, en mètres, pour différents horizons H_{loc} et différents poids w_{loc}

en termes d'exploration et de précision de localisation, les paramètres $H_{\text{loc}} = 1$ et w_{loc} compris entre 0.02 et 0.07 semblent être le choix le plus adéquat.

VI.1.4 Discussion

Ces résultats d'expérience montrent l'efficacité de la méthode proposée pour s'assurer que la localisation visuelle n'est pas mise en défaut par les zones peu texturées est efficace. Le robot parvient à explorer son environnement et garde une localisation précise.

Ces expériences ont été réalisées uniquement avec le premier algorithme développé pour évaluer la qualité d'une scène future pour la localisation, le critère par point, qui cherche à prédire la visibilité des amers issus de l'algorithme d'odométrie visuelle (voir en partie V.1).

Le deuxième critère développé, le critère par région, présenté dans le chapitre précédent V.3 a montré des performances similaires par rapport au critère par point pour un temps de calcul largement inférieur. Dans de futurs travaux, on pourrait réaliser ces expériences en intégrant ce critère par région. Il permettrait d'embarquer tous les algorithmes directement sur le robot ou de proposer un ensemble de commandes plus grand à évaluer dans l'algorithme MPC. Réaliser ces expériences permettrait également de comparer la performance des deux critères développés pour estimer la qualité d'une scène future pour la localisation dans des expériences en boucle fermée.

Ces expériences ont montré que le robot se retrouve face au compromis entre explorer de nouvelles zones encore inconnues, poussé par le coût d'exploration, et de rester sur

des zones connues pour garder le maximum d'amers dans son champ de vue, poussé par le coût de qualité de localisation. On retrouve le compromis relevé dans les références d'exploration "active" [Kim and Eustice, 2013, Bryson and Sukkarieh, 2008]. Dans les travaux présentés ici, pour s'assurer que le robot réponde aux deux objectifs de mission, il a fallu fixer l'horizon de prédiction de scène H_{loc} à 1 et régler les pondérations liées à ces deux objectifs de façon adéquate.

VI.2 Ralliement de points avec évitement d'objets mobiles

Le deuxième scénario considéré est la réalisation de missions de ralliement de points de passage avec évitement des objets mobiles. Le seul capteur embarqué sur le robot est un banc stéréo. Toute la chaîne développée pour répondre à l'objectif de la mission utilise donc à nouveau uniquement les informations visuelles. La partie VI.2.1 décrit l'architecture du système mis en place pour réaliser les expériences ainsi que les différents modules développés pour la détection et la prise en compte des objets mobiles dans la commande. Les résultats des expérimentations sont exposés dans la partie VI.2.2.

VI.2.1 Description de l'architecture du système

La figure VI.10 montre le schéma de l'architecture mise en place pour réaliser ce type de missions.

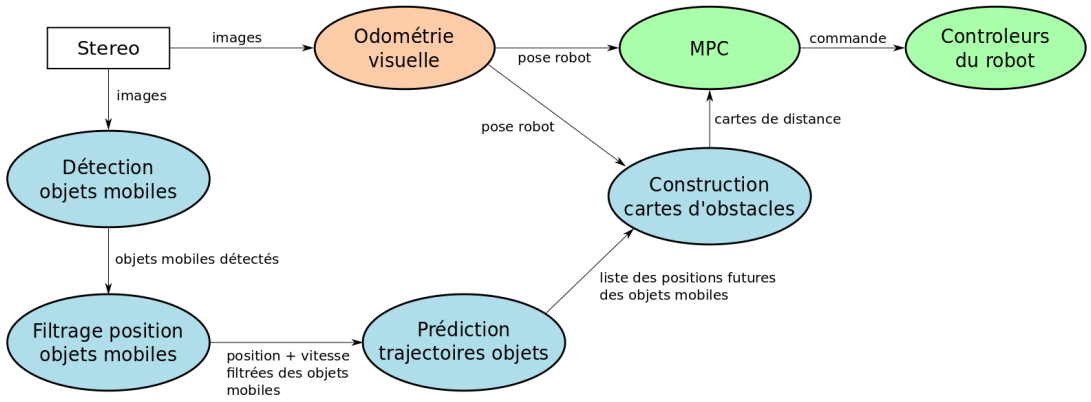


FIGURE VI.10 – Architecture du système pour le ralliement de points avec évitement des obstacles mobiles

La localisation est assurée par l'algorithme d'odométrie visuelle eVO (voir en section III.1.4). Dans cette partie, on considère que les images reçues sont suffisamment texturées pour pouvoir calculer la localisation du robot de façon précise. La détection des objets mobiles est réalisée uniquement à partir des images reçues par le banc stéréo, elle est présentée dans la section VI.2.1.1. Un filtre de Kalman permet d'obtenir une meilleure estimation de la position des objets mobiles, il est décrit dans la sec-

tion VI.2.1.2. A partir de l'estimation de la position et de la vitesse, la trajectoire future des objets mobiles est prédite, comme expliqué en section VI.2.1.3. Des cartes d'obstacles sont créées, elles représentent à chaque pas de temps l'occupation prédite du sol par tous les objets mobiles détectés, voir en section VI.2.1.4. Enfin, ces informations sont envoyées à une boucle de commande basée sur la commande prédictive dans le but de rallier le point de passage tout en anticipant la trajectoire future des objets mobiles afin de les éviter. Les explications sont données dans la section VI.2.1.5.

VI.2.1.1 Détection des objets mobiles

La détection des objets mobiles est basée sur l'utilisation d'algorithmes de stéréovision denses. Le principe est de prédire l'image entre deux instants consécutifs sous l'hypothèse d'une scène rigide. L'image prédite est ensuite comparée avec l'image observée pour former une image résiduelle, qui après seuillage, donne les objets mobiles potentiels. Cette méthode a été développée par Maxime Derome [Derome, 2017], lors de sa thèse réalisée à l'ONERA DTIS. Les principales étapes de cette méthode sont rappelées ci-dessous.

Les images stéréo sont utilisées pour estimer les paramètres de déplacement (matrice de rotation R et vecteur de translation T) entre les instants t_{n-1} et t_n par odométrie visuelle. Ces paramètres sont utilisés pour compenser le mouvement de la caméra entre ces deux instants.

Les images stéréo sont également utilisées pour calculer une carte de profondeur dense en utilisant la méthode ACTF (Accurate Coarse To Fine) [Sizintsev and Wildes, 2010]. Les points des deux images sont triangulés en utilisant l'équation (III.1) pour obtenir l'estimation de la position 3D du point $\hat{Y}_n$.

Tous les calculs suivants sont réalisés uniquement sur les images reçues par la caméra gauche. On appelle I_n l'image gauche à l'instant t_n. La prédiction d'image utilise l'image gauche courante combinée avec la carte de profondeur estimée et les paramètres de déplacement afin de prédire l'image gauche I_{n-1}^{pred} à l'instant t_{n-1}.

Pour compenser le mouvement de la caméra et revenir à l'instant précédent, les coordonnées d'un pixel dans l'image précédente sont prédites sous l'hypothèse d'une scène statique par :

$$\begin{pmatrix} u_{n-1}^{pred} \\ v_{n-1}^{pred} \end{pmatrix} = \Pi_{n-1}\left(R^{-1}.(\hat{Y}_n - T)\right) \tag{VI.7}$$

avec Π_{n-1} la fonction de projection (voir equation (II.8)) à t_{n-1}.

L'image prédite I_{n-1}^{pred} est obtenue en interpolant les intensités de l'image I_{n-1} à ces positions prédites. A partir de I_{n-1}^{pred} et I_{n-1}, on calcule le résidu $\Phi_r(u, v)$ pour chaque pixel. Cette opération est réalisée par l'algorithme de flot optique eFolki [Plyer et al., 2016]. Elle présente un coût de calcul important et est réalisée sur un GPU pour pouvoir être réalisée en temps-réel. Les incertitudes sont considérées et propagées afin d'estimer la matrice de covariance de l'image résiduelle, de manière similaire à ce qui a été présentée en section V.1.1.2.

Ensuite, une statistique χ^2 est calculée. On obtient les détections par seuillage du résultat. Des étapes de traitement supplémentaires permettent de nettoyer les fausses détections et de calculer des boites englobantes autour des zones où la détection de mouvement est confirmée.

La figure VI.11 illustre la chaîne de détection complète.

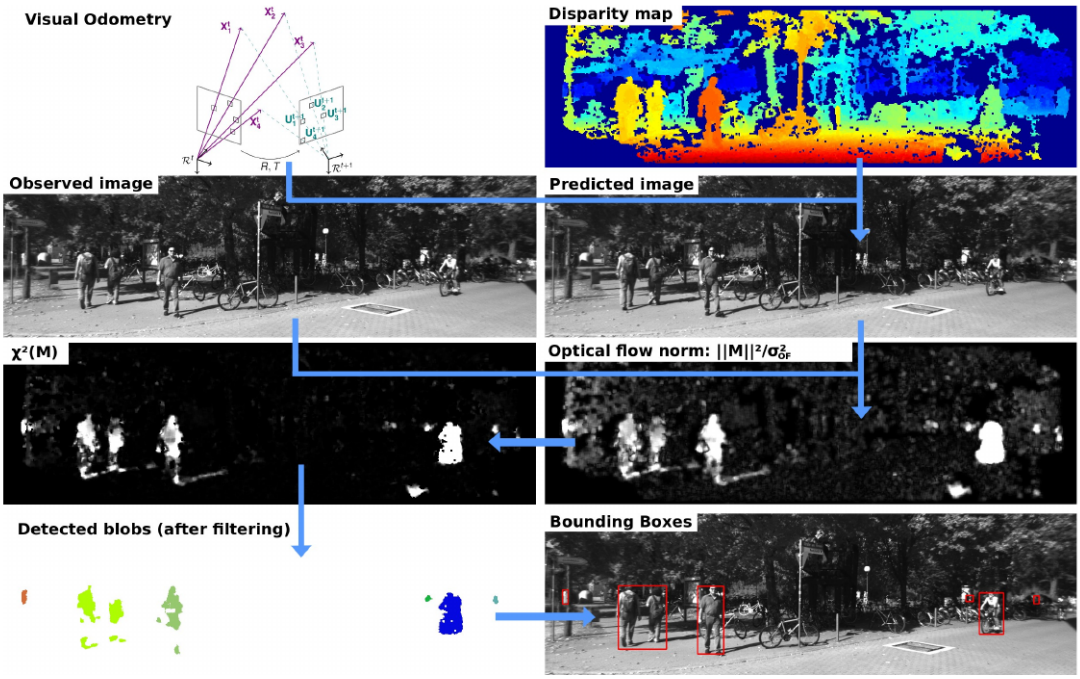


FIGURE VI.11 – Chaîne de détection des objets mobiles par vision stéréo. L'odométrie visuelle et le calcul de profondeur permettent de reconstruire l'image à l'instant précédent. Le flot optique entre l'image prédite et observée donne un résidu, pondéré en fonction des incertitudes de mesure puis seuillé pour donner les détections. Ensuite les boites englobantes sont calculées. Illustration issue de [Derome et al., 2016]

Quand plusieurs objets sont détectés, chacun se voit attribuer un numéro. L'algorithme cherche à associer temporellement les objets détectés en utilisant une méthode des plus proches voisins. Cette association peut être uniquement établie entre deux instants consécutifs. Si un objet connu n'est pas détecté pendant un pas de temps, l'algorithme n'est pas capable de déterminer l'association temporelle. Elle sera alors réalisée par le module de filtrage, présenté dans la partie suivante.

L'algorithme vérifie également la cohérence temporelle des observations : un objet détecté doit l'avoir été au pas de temps précédent et la distance entre la position actuelle et la position prédite de l'objet doit être inférieure à un certain seuil. Si un objet détecté ne remplit pas ces conditions, il est considéré comme une fausse détection et n'est pas envoyé au module de filtrage.

VI.2.1.2 Filtrage de la position des objets mobiles

Un objets mobile détecté est modélisé comme un cylindre de rayon r et de hauteur h. A chaque pas de temps, la sortie du module de détection est un vecteur noté :

$$Z_n = \left[\xi_n^T, r_n, h_n\right]^T \tag{VI.8}$$

$\xi_n = (x_n, y_n)^T$ est la position estimée du centre de gravité du cylindre dans le repère du robot. La composante sur l'axe z n'est pas considérée car le cylindre est supposé se déplacer parallèlement au plan du sol.

Comme ces mesures peuvent être très bruitées à cause d'ambiguïtés géométriques dans le module de détection, une opération de filtrage de la position et des paramètres du cylindre est réalisée par un filtre de Kalman. Elle permet également d'obtenir une estimation de la vitesse de l'objet, qui est nécessaire pour estimer la trajectoire future de l'objet mobile. Le vecteur d'état dans le filtre de Kalman à l'instant t_n est noté

$$\mathcal{X}_n = \left[\xi_n^T, V_n^T, r_n, h_n\right] \tag{VI.9}$$

et sa covariance associée est notée P_n.

Le vecteur de mesure est le vecteur Z_n (VI.8) et la matrice de covariance du bruit de mesure est notée R_n :

$$R_n = \begin{pmatrix} \sigma_x^2.I_2 & 0 & 0 \\ 0 & \sigma_r^2 & 0 \\ 0 & 0 & \sigma_h^2 \end{pmatrix} \tag{VI.10}$$

Q_n est la matrice de covariance du bruit de modèle.

$$Q_n = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma_v^2.I_2 & 0 & 0 \\ 0 & 0 & \sigma_s^2 & 0 \\ 0 & 0 & 0 & \sigma_s^2 \end{pmatrix} \tag{VI.11}$$

On suppose que la vitesse du cylindre est constante entre t_{n-1} et t_n, ce qui donne le modèle d'état :

$$\begin{cases} \mathcal{X}_n = A\mathcal{X}_{n-1} + w_n \\ Z_n = C\mathcal{X}_n + b_n \end{cases} \tag{VI.12}$$

avec w_n et b_n, les bruits de modèle et d'observation,

$$A = \begin{pmatrix} I_2 & t_e.I_2 & 0 & 0 \\ 0 & I_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{VI.13}$$

et

$$C = \begin{pmatrix} I_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{VI.14}$$

t_e est la période d'échantillonnage. I_2 est la matrice identité de dimension 2.

Ce modèle est linéaire, on peut donc utiliser un filtre de Kalman classique. Comme le filtre de Kalman étendu présenté dans la partie IV.4.1, le processus se décompose en deux étapes : une étape de prédiction, qui estime l'état courant à partir du modèle du système et de l'état précédent et une étape de mise à jour, qui affine l'estimation en utilisant le vecteur de mesure Z_n. Le système linéaire considéré ici permet de simplifier l'estimation de la covariance sur l'état.

— Etape de prédiction

$$\widehat{\mathcal{X}}_{n|n-1} = A.\mathcal{X}_{n-1|n-1} \tag{VI.15}$$

$$P_{n|n-1} = A.P_{n-1|n-1}.A^T + Q_n \tag{VI.16}$$

— Etape de mise à jour

— Calcul de l'innovation et de sa covariance

$$\tilde{Y}_n = Z_n - C.\widehat{\mathcal{X}}_{n|n-1} \tag{VI.17}$$

$$S_n = C.P_{n|n-1}.C^T + R_n \tag{VI.18}$$

— Gain de Kalman optimal

$$K_n = P_{n|n-1}.C^T.S_n^{-1} \tag{VI.19}$$

— Mise à jour de l'état et de sa covariance

$$\widehat{\mathcal{X}}_{n|n} = \widehat{\mathcal{X}}_{n|n-1} + K_n\tilde{Y}_n \tag{VI.20}$$

$$P_{n|n} = (I - K_n.C)\, P_{n|n-1} \tag{VI.21}$$

Quand l'objet mobile n'est plus détecté par le module de détection, seule l'étape de prédiction est réalisée. La position de l'objet est prédite en utilisant la dernière vitesse estimée. Quand de nouvelles mesures arrivent, le procédé reprend son fonctionnement nominal, incluant l'étape de mise à jour.

Plusieurs objets peuvent être pris en compte simultanément. Lorsque le module de détection fonctionne correctement, c'est lui qui réalise l'association temporelle entre un objet pisté et une nouvelle détection. En cas de non détection, l'association est faite dans l'étape de filtrage en comparant la position prédite de l'objet pisté et celle de la détection la plus proche. Si la distance est inférieure à un certain seuil (fixé à 0.5 m) on associe l'objet pisté et la nouvelle détection. Le calcul de la position reprend alors un

fonctionnement normal. Si un objet n'est plus détecté pendant un certain temps (1.0 s), il est oublié.

Le temps de calcul de l'algorithme de détection des objets mobiles peut donner un retard non négligeable entre l'instant où les images sont reçues et l'instant où la mesure Z_n est envoyée au module de filtrage. Ce retard, noté Δt, est estimé en comparant les différents instants de réception des informations. La position mesurée de l'obstacle est alors corrigée selon l'équation :

$$\xi_n = \xi_n^r + \Delta t.\widehat{V}_{n-1} \tag{VI.22}$$

avec ξ_n^r la position reçue et ξ_n la position corrigée. $\widehat{V}_{n-1}$ est la vitesse estimée par le filtre de Kalman au pas de temps précédent. Cette opération est réalisée à la réception des mesures, avant l'étape de mise à jour dans le filtre de Kalman.

VI.2.1.3 Prédiction de la trajectoire des objets mobiles

Pour pouvoir anticiper le mouvement d'un objet mobile, sa trajectoire future est prédite sur l'horizon de prédiction H_p de la boucle de commande MPC. Pour cela, on utilise la dernière vitesse estimée dans le filtre de Kalman $\widehat{V}_n$, supposée constante sur l'horizon H_p. La liste des positions futures prédites $(\tilde{\xi}_i)_{i\in[\![n+1,n+H_p]\!]}$ sur l'horizon de prédiction est obtenue par :

$$\tilde{\xi}_n = \widehat{\xi}_n \tag{VI.23}$$

$$\forall k \in [\![n, n+H_p-1]\!], \tilde{\xi}_{k+1} = \tilde{\xi}_k + t_e.\widehat{V}_n \tag{VI.24}$$

On note $\Xi_k = \left\{\tilde{\xi}_k^{(j)}, j = [\![1, N_{\text{obs}}]\!]\right\}$ l'ensemble des positions prédites des N_{obs} obstacles à l'instant t_k.

VI.2.1.4 Création des cartes d'obstacles

Pour chaque pas de temps t_k sur l'horizon de prédiction H_p, la carte d'occupation contenant les N_{obs} obstacles détectés est construite à partir de l'ensemble des positions prédites Ξ_k. Les cylindres sont supposés être perpendiculaires au plan du sol, l'occupation de l'obstacle j sur le sol est donc un disque de centre $\tilde{\xi}_k^{(j)}$ et de rayon $\widehat{r}_n$, dernière estimation effectuée du rayon par le filtre de Kalman. Pour inclure l'objet dans la carte d'occupation, on considère le carré englobant ce cercle. Cette opération est réalisée pour tous les objets mobiles détectés et pour chaque pas de temps t_k.

On obtient donc H_p cartes d'occupation contenant chacune les N_{obs} objets mobiles prédits. On calcule ensuite pour chaque pas de temps t_k, la carte de distance en utilisant une transformée de distance euclidienne, comme effectué dans la partie IV.3.3.3.

Dans les expériences réalisées, les obstacles fixes n'ont pas été considérés, mais ils peuvent facilement être pris en compte en les ajoutant à la carte d'occupation.

La hauteur h du cylindre n'est pas considérée, car le robot se déplace sur le plan du sol. Elle peut être utilisée pour des missions de navigation sur des drones, dans

lesquelles il est possible de contourner un obstacle en passant en-dessous ou au-dessus. Dans ce cas, il faut créer des grilles d'occupation en 3D. Un tel système est décrit dans l'article [Roggeman et al., 2017c].

VI.2.1.5 Commande prédictive pour l'évitement

Pour prendre en compte les obstacles mobiles dans la commande, un coût d'obstacle mobile a été formulé. Il s'inspire du coût d'obstacle décrit dans la section IV.3.3.3. Ce coût pénalise les trajectoires dont les positions futures risquent d'entrer en collision avec les futures positions des objets mobiles. X_k est la position prédite du robot à l'instant t_k. Le coût d'obstacle mobile est défini par :

$$J_{\text{obs\_m}} = \frac{1}{H_p} \sum_{k=n+1}^{n+H_p} f_{\text{obs}} \left(d_{\text{obs}} \left(X_k, \Xi_k \right) \right). \tag{VI.25}$$

f_{obs} est la fonction de pénalité (IV.15) décrite en section IV.3.3.3. $d_{\text{obs}} \left(X_k, \Xi_k \right)$ est la distance entre la position prédite du robot et l'obstacle prédit le plus proche à l'instant t_k. Cette distance est obtenue à partir de la carte de distance calculée pour le pas de temps t_k.

La fonction de coût de la commande MPC pour le ralliement de points avec évitement des obstacles mobiles est :

$$J = w_{\text{wp}} J_{\text{wp}} + w_{\text{obs\_m}} J_{\text{obs\_m}} \tag{VI.26}$$

J_{wp} est le coût de ralliement de points, défini dans la section IV.3.3.2.

La figure VI.12 illustre le fonctionnement de la commande MPC avec cette fonction de coût. Le robot cherche à s'approcher du point tout en considérant la trajectoire future de l'obstacle mobile. La trajectoire optimale n'est donc pas la trajectoire qui s'avance le plus vers le point à rallier car celle-ci s'approche trop près des positions prédites de l'obstacle mobile. Le robot préfère une trajectoire qui permet de garder une plus grande distance avec les positions futures de l'objet mobile.

VI.2.1.6 Algorithme

L'algorithme 5 résume la procédure de détection et d'évitement des objets mobiles.

VI.2.2 Expérimentations

La boucle de commande développée a été embarquée sur un robot mobile afin de réaliser des expériences de ralliement de points avec évitement des obstacles mobiles en situation réelle. La section VI.2.2.1 explique la mise en œuvre des expériences sur le robot mobile. La section VI.2.2.2 expose les résultats obtenus.

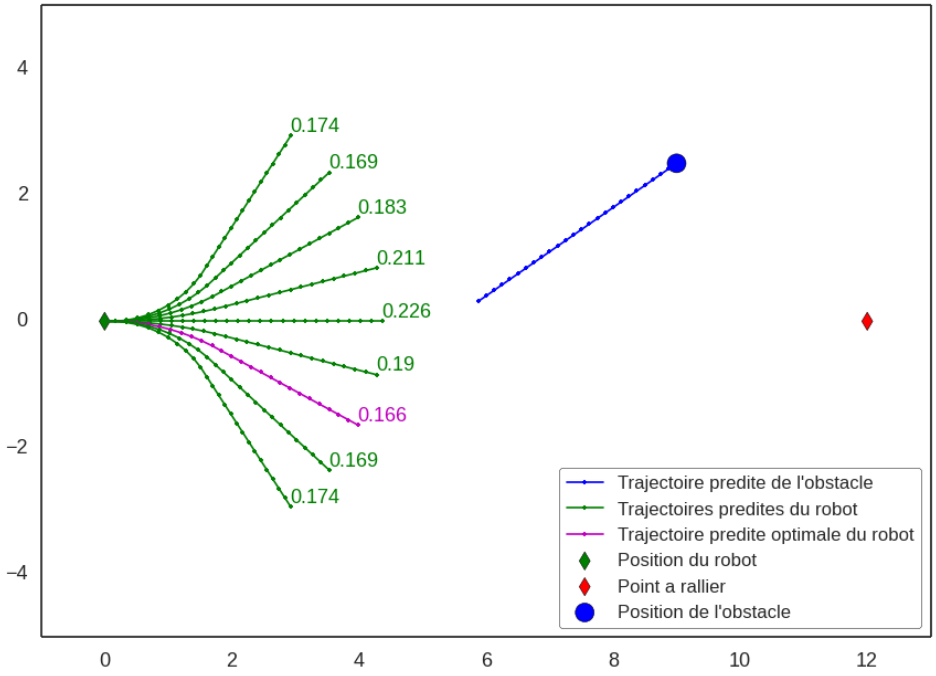


FIGURE VI.12 – Exemple de calcul de la trajectoire optimale pour une mission de ralliement de points avec évitement des obstacles mobiles

VI.2.2.1 Mise en œuvre des expérimentations

Le scénario considéré simule une mission de navigation en environnement urbain. L'environnement est une route large, sur laquelle le robot peut manœuvrer pour éviter les objets mobiles. Ces derniers sont des piétons, avec une vitesse de marche de 1.5 m.s^{-1}.

Le robot utilisé est le Robotnik Summit XL, présenté dans la section II.2.2. Il doit atteindre un point situé à 10 ou 15 mètres devant sa position de départ suivant les expériences.

La figure VI.13 présente une photo de l'environnement considéré pour ces expériences.

Les paramètres utilisés sont les suivants :

— Horizons : $H_p = 30$, $H_c = 10$

— Limites de vitesse : $v_{max} = 0.5$ m.s^{-1}, $\omega_{max} = 0.5$ rad.s^{-1}

— Résolution de la carte d'obstacle : $r = 0.15$ m/pixel

— Distance aux obstacles mobiles : $d_{sec} = 2.0$ m, $d_{des} = 4.0$ m

— Poids sur le coût d'obstacle mobile : $w_{\text{obs\_m}} = 100$

— Poids sur le coût de ralliement de points : $w_{\text{wp}} = 1$

Les horizons sont longs car on veut anticiper largement la trajectoire des objets mobiles. Les vitesses maximales sont fixées en fonction des capacités du robot. Les distances aux obstacles sont élevées pour s'assurer que le robot reste loin des obstacles. Le poids

Algorithm 5 Boucle de détection et évitement des obstacles mobiles

A chaque pas de temps t_n

1. Détection des objets mobiles à partir des images stéréo
2. Filtrage de la position et estimation de la vitesse de chaque objet mobile connu et prédiction de la position des objets qui ne sont plus détectés
3. Prédiction de la trajectoire de chaque objet à partir de sa position et de sa vitesse estimées
4. Création des H_p cartes d'obstacles et calcul des cartes de distance correspondantes
5. Calcul de la commande optimale

FIGURE VI.13 – Photo du robot dans son environnement de navigation

sur le coût d'obstacle mobile est largement supérieur à celui sur le coût de ralliement de points. Le coût d'obstacle mobile est nul en l'absence d'obstacle, il a donc de l'influence sur le résultat d'optimisation de la fonction de coût uniquement quand des obstacles ont été détectés dans la zone atteignable sur l'horizon de prédiction. Dans ce cas, l'objectif d'évitement des obstacles est largement prépondérant à l'objectif de ralliement de point.

L'optimisation de la fonction de coût de la commande MPC est réalisée par discrétisation de l'espace des commandes. Un ensemble de 12 commandes est défini afin de s'assurer que le temps de calcul soit inférieur à la période d'échantillonnage t_e. Il contient la commande nulle $U = (v = 0, \omega = 0)^T$ et 11 commandes avec une vitesse linéaire égale à v_{max} et des vitesses angulaires réparties uniformément sur l'intervalle $[-\omega_{max}, \omega_{max}]$.

VI.2.2.2 Résultats

Plusieurs expériences ont été réalisées avec différents cas de figure.

Dans la première expérience, le robot doit rejoindre un point situé 10 mètres devant lui. Un seul objet mobile traverse la trajectoire future du robot avec un angle de 45°. La figure VI.14 montre différentes étapes de la trajectoire réalisée par le robot. Au début de la mission, le robot avance vers le point à rallier. Un objet mobile est détecté, la trajectoire optimale prédite est adaptée en conséquence. L'obstacle est correctement évité et le robot peut reprendre une trajectoire en direction du point à rallier.

Dans la deuxième expérience, le robot doit rejoindre un point situé à 15 mètres devant lui. Deux objets mobiles traversent la trajectoire du robot. Le premier arrive face à lui tandis que le second s'approche avec un angle de 30°. Le robot détecte tardivement le premier obstacle. Il parvient tout de même à l'éviter mais il ne respecte pas la distance de sécurité imposée ($d_{sec} = 2.0$ m). Le deuxième obstacle est détecté suffisamment tôt pour que le robot l'évite en respectant la distance de sécurité. Finalement, le robot parvient à rejoindre son point de passage.

La détection des objets mobiles est réalisée en détectant les mouvements apparents dans l'image. Or, quand un objet mobile se déplace vers la caméra, le mouvement apparent dans l'image est petit et donc difficilement détectable. C'est pourquoi le premier objet mobile a été détecté tardivement.

Dans d'autres expériences réalisées, on a constaté quelques difficultés :

— Les fausses détections : malgré le test de cohérence temporelle réalisé dans le module de détection, il arrive que des objets qui n'existent pas soient considérés dans la boucle de commande. Le robot cherche alors à les éviter.

— Les détections intermittentes : les objets ne sont pas détectés à tous les pas de temps. La vitesse de déplacement des objets est faible par rapport à la fréquence d'acquisition des images (4 à 10 Hz), la différence entre l'image prédite et l'image obtenue est parfois trop petite pour détecter les objets mobiles. Le système d'association temporelle dans le module de filtrage n'est pas assez robuste pour pallier ce problème dans tous les cas. Par exemple, quand l'estimation de la vitesse est peu précise, la prédiction de la trajectoire ne correspond pas à la trajectoire réellement effectuée par l'objet mobile. On ne peut pas retrouver la correspondance entre l'objet prédit et l'objet mesuré. Pour avoir un fonctionnement plus robuste, on pourrait réaliser dans le module de détection une association temporelle sur plusieurs pas de temps consécutifs.

En conclusion, le système développé a montré sa capacité à détecter des objets en mouvement et à les éviter dans un objectif de ralliement de points en utilisant uniquement les informations visuelles. Cependant, le module de détection et la gestion des objets mobiles pourraient être améliorés afin d'éviter les problèmes de fausses détections et de détections intermittentes.

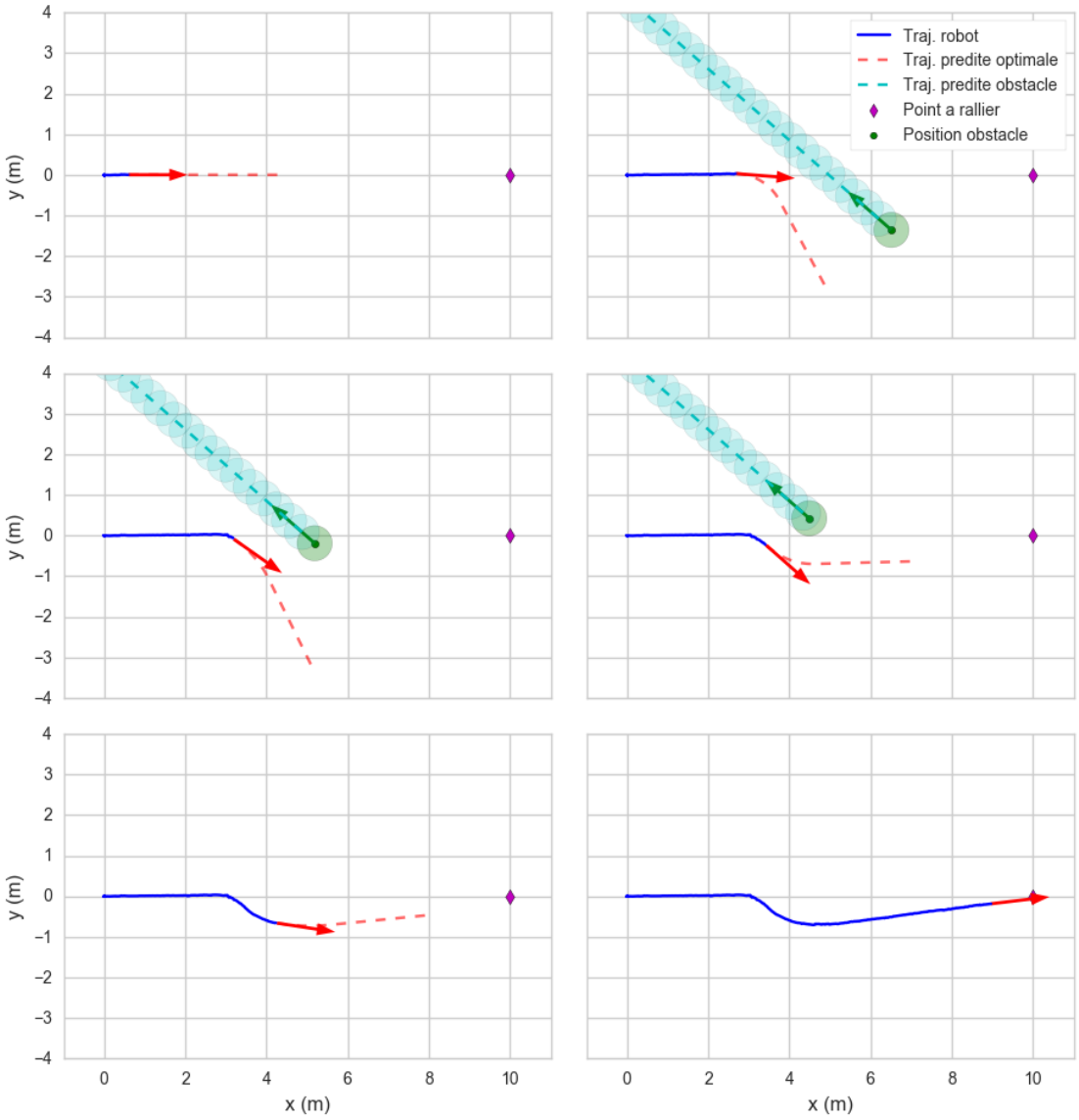


FIGURE VI.14 – Résultat d'une mission de ralliement de points de passage avec évitement d'un objet mobile. La courbe bleue est la trajectoire du robot, la courbe rouge en pointillé est sa trajectoire optimale prédite. Le cercle vert représente l'obstacle mobile, la courbe bleue en pointillé est sa trajectoire prédite. Le diamant violet est le point à rallier.

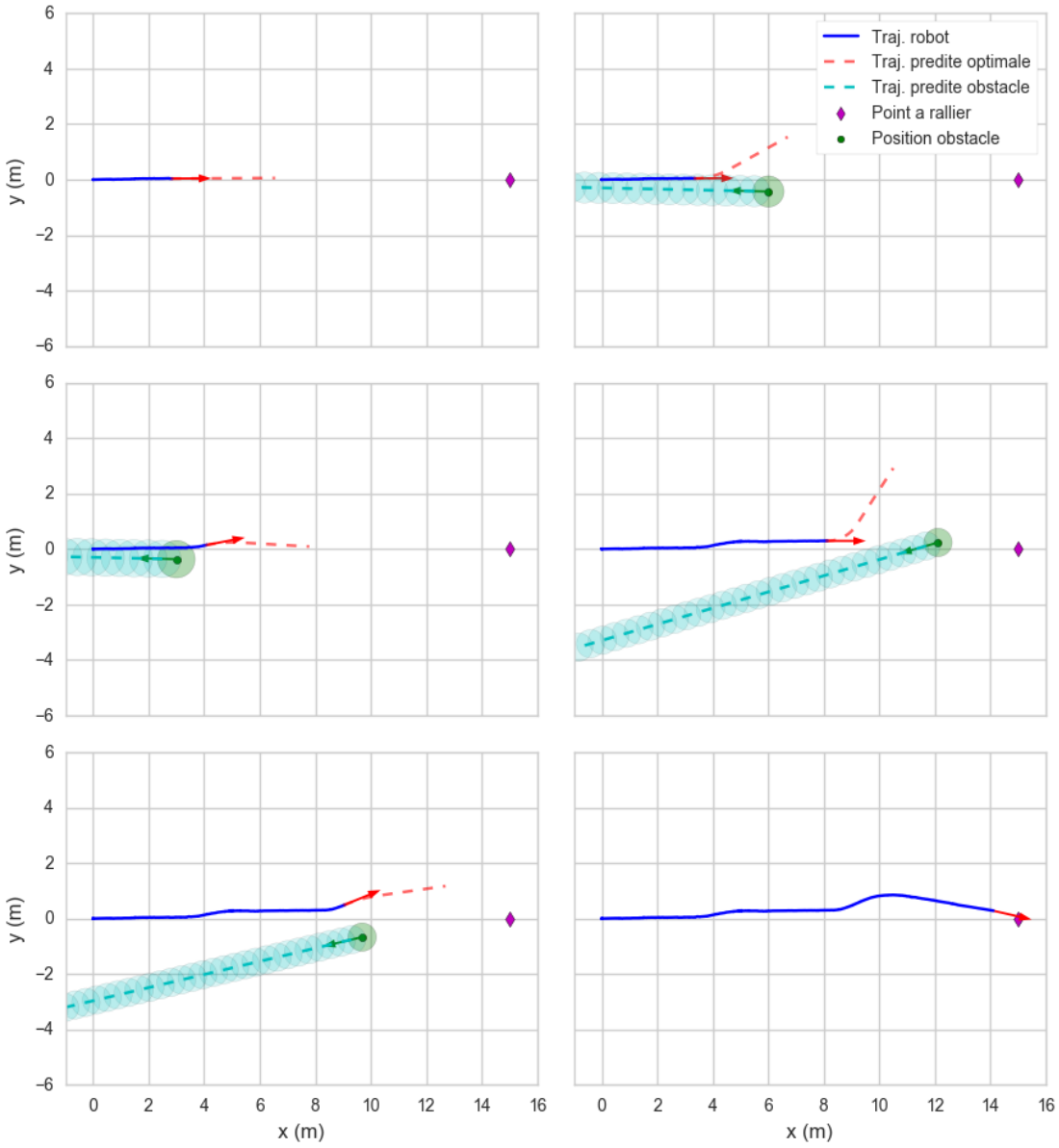


FIGURE VI.15 – Résultat d'une mission de ralliement de points de passage avec évitement de deux objets mobiles.

VI.3 Conclusion

On a montré dans ce chapitre la possibilité de réaliser des missions complexes sur des robots terrestres en utilisant uniquement les images provenant d'un banc stéréo. Deux problématiques ont été étudiées : la première est la réalisation de missions d'exploration avec évitement d'obstacles dans des environnements où la localisation basée vision peut être imprécise à cause du manque de texture dans les images perçues. Pour s'assurer que la localisation visuelle reste précise, un critère prédisant la qualité de la scène future a été intégré à la boucle de commande. De nombreuses expériences ont été réalisées et ont montré que l'ajout de ce critère permet d'éviter que le robot se retrouve face à des zones peu texturées. Ces expériences ont permis également de déterminer les paramètres permettant d'obtenir la meilleure couverture de zone et la meilleure précision dans la localisation. Deux critères ont été développés au chapitre précédent pour évaluer la qualité d'une scène future pour la localisation. Seul le critère par point a été étudié dans ces expériences. Une perspective envisagée est de réaliser de nouvelles expériences avec le critère par région. On pourrait ainsi comparer la performance entre les deux méthodes sur des expériences en boucle fermée.

La deuxième problématique étudiée est la réalisation de missions de ralliement de points de passage avec évitement d'obstacles mobiles. La détection des obstacles est réalisée uniquement à partir des images stéréo. Un filtre de Kalman permet d'obtenir une meilleure estimation de la position des obstacles détectés et de calculer leur vitesse. La boucle de commande prend en compte la trajectoire prédite de chaque objet mobile détecté afin de rallier le point de passage tout en anticipant les mouvements des objets mobiles. Les expériences menées ont permis de montrer que la boucle de perception-commande développée permet de réaliser la mission considérée. Le robot parvient à détecter et éviter les obstacles mobiles et à finalement rejoindre son objectif. Cependant, plusieurs limitations ont été observées. Le module de détection détecte tardivement les obstacles s'approchant droit sur les caméras. On a observé également des détections intermittentes et des fausses détections. Pour remédier à ces problèmes, une meilleure gestion temporelle des obstacles détectés pourrait être réalisée.

Chapitre VII

Conclusion et perspectives

Dans cette thèse, nous avons développé des solutions pour effectuer des missions de navigation diverses sur des robots terrestres équipés de capteurs de vision pour se localiser. Ces capteurs permettent d'obtenir une information riche de l'environnement. Nous avons choisi d'utiliser des bancs stéréo, qui apportent directement une information sur la profondeur des amers, et de travailler avec l'algorithme d'odométrie visuelle eVO. Cependant, toutes les solutions développées dans cette thèse sont compatibles avec n'importe quel algorithme d'odométrie visuelle stéréo basé amers. En termes de commande, tous les travaux réalisés se basent sur la commande prédictive. Cette approche permet de prendre en compte des objectifs de missions divers ainsi que des contraintes sur le robot et est compatible avec une utilisation temps-réel, contrairement aux méthodes de planification de trajectoire. Cependant, elle nécessite un réglage de paramètres qui n'est pas simple. De nombreuses expériences ont été réalisées sur des robots terrestres afin de valider les systèmes développés.

Pour s'assurer que la localisation visuelle reste précise pendant toute la durée de la mission, nous avons proposé deux solutions pour éviter les pertes de localisation dues au manque de texture dans la scène observée par les caméras.

La première solution proposée a consisté à mettre en place une fusion de données multi-capteurs entre les données de localisation visuelle et celles provenant des odomètres des roues et d'une centrale inertielle. Ce système de localisation a été intégré dans une boucle de commande afin de réaliser des missions d'exploration autonome avec évitement d'obstacles en environnement inconnu et encombré.

La deuxième solution proposée est d'adapter la trajectoire du robot afin de s'assurer que les images perçues par les caméras soient suffisamment texturées pour être exploitables par l'algorithme d'odométrie visuelle et ainsi d'obtenir une localisation toujours précise du robot. Pour cela, nous avons développé deux critères pour prédire la qualité de la scène future. Le premier critère cherche à prédire le nombre d'amers 3D qui seront visibles dans les images futures à partir de la connaissance de la position future de la caméra et des amers fournis par l'algorithme d'odométrie visuelle. Ce critère a ensuite été intégré dans une boucle de commande afin de réaliser des missions de navigation par points de passage. Les résultats d'expériences ont montré que l'ajout du critère dans la stratégie de commande permet au robot d'adapter sa trajectoire face aux zones peu texturées et de rallier son objectif tout en gardant une localisation précise. Le deuxième critère développé cherche à améliorer le temps de calcul trop élevé du premier critère et à le rendre moins dépendant du fonctionnement de l'algorithme d'odométrie visuelle. Deux stratégies ont été proposées : le découpage en régions de l'image par des rectangles ou par des superpixels. En comparant les différentes méthodes sur des données d'expé-

riences, nous avons montré que la méthode basée sur le découpage en rectangles présente les mêmes performances que le critère par point tout en réduisant significativement le temps de calcul.

Après avoir réalisé des premières expériences simples de couplage commande/vision, nous avons effectué des expériences de navigation autonome prenant en compte des objectifs de mission plus complexes sur des robots équipés uniquement de caméras. Un premier type d'expérience considéré est l'exécution de missions d'exploration autonome avec évitement d'obstacles. La stratégie de commande mise en place intègre le critère par point afin que le robot évite les zones où le calcul de la localisation visuelle risque d'échouer. De nombreuses expériences ont été réalisées et ont montré que l'ajout du critère sur la qualité de localisation permet au robot d'éviter les zones dans lesquelles sa localisation sera imprécise et permet d'améliorer la couverture de la zone ainsi que la précision dans le calcul de la localisation visuelle.

Le deuxième type d'expériences étudié est le ralliement de points de passage avec évitement des obstacles mobiles, en utilisant de nouveau uniquement les informations visuelles pour détecter les obstacles mobiles et calculer la position du robot. Après avoir été détectés, les obstacles sont modélisés et leur trajectoire future est prédite afin qu'ils soient pris en compte dans la commande du robot. Des expériences en situation réelle ont montré que la chaine de perception-commande embarquée permet de détecter correctement et d'éviter les obstacles mobiles qui traversent la trajectoire future du robot.

Perspectives

A la suite de ces travaux, de nombreuses perspectives sont envisagées.

Pour les missions d'exploration autonome avec prise en compte de la qualité de localisation, les expériences effectuées, décrites en partie VI.1, ont été réalisées en utilisant uniquement le premier critère développé pour évaluer la qualité d'une scène future pour la localisation, qui cherche à prédire la visibilité future des amers issus de l'algorithme d'odométrie visuelle, décrit dans la partie V.1.

Une des perspectives envisagée après la réalisation de ces expériences est de réaliser une deuxième série d'expériences en utilisant le deuxième critère développé, basé sur la prédiction de la visibilité des régions de l'image, décrit dans la partie V.3. Le gain en temps de calcul de cette deuxième méthode, démontré dans la partie V.3.3, permettrait de tester plus de trajectoires dans la procédure de commande et donc d'affiner la commande à envoyer au robot. On pourrait également obtenir une comparaison des deux méthodes de prédiction de la scène future sur des expériences en boucle fermée.

Sur les expériences de ralliement de points de passage avec évitement des obstacles mobiles, nous avons observé plusieurs difficultés sur la détection des objets mobiles à l'issue des expériences réalisées. Le système pourrait être perfectionné en améliorant la gestion de la détection et de l'association d'un même objet sur plusieurs images, particulièrement quand l'objet n'a pas été détecté sur une ou plusieurs images et réapparait. Une autre limitation provient de la méthode de détection elle-même qui n'exploite que des notions géométriques (vitesse des obstacles par rapport à la scène), qui peuvent être peu

observables dans certaines configurations. Le recours à des méthodes de détection basées sur l'apparence des objets pourrait apporter une meilleure robustesse à ces situations.

Nos choix algorithmiques ont été faits pour limiter autant que possible la charge de calcul de manière à obtenir des solutions embarquables sur de nombreux types de plateformes robotiques. Dans les expériences présentées dans ce manuscrit, les algorithmes ont été embarqués sur des robots terrestres mais ils pourraient, pour la plupart, être embarqués sur des petits drones. Une exception est l'algorithme d'évitement d'objets mobiles qui utilise une fonction de détection nécessitant un GPU. Cependant les autres modules de cette solution (filtrage temporel, MPC) peuvent fonctionner sur des CPU embarquables sur drones. La référence [Roggeman et al., 2017c] décrit une solution d'évitement d'obstacle mobile par un drone qui est fondée sur les solutions développées dans cette thèse, avec un algorithme de détection d'objets moins coûteux que celui que nous avons utilisé dans cette thèse. De plus, les drones possèdent un degré de liberté supplémentaire par rapport aux robots utilisés dans les expériences menées dans cette thèse, ils peuvent fixer l'angle de lacet indépendamment de leur direction de déplacement. Il serait alors intéressant d'optimiser cet angle lors de missions de suivi de trajectoire en intégrant le critère développé pour prédire la qualité de la scène future. Cela pourrait permettre d'obtenir un calcul de localisation plus précis sans dévier la trajectoire.

Il serait également intéressant de combiner les modules de détection des objets mobiles et de prédiction de leur trajectoire future avec le module de prédiction de la qualité de scène. En effet, les objets mobiles peuvent générer des occultations, ce qui cache une partie de la scène et donc potentiellement des zones texturées. Combiner ces deux modules permettrait de réaliser une prédiction plus fine de la qualité de scène future, en anticipant les zones qui ne seront pas visibles pour les caméras.

Le problème multi-objectifs a été formulé comme une somme pondérée des différents objectifs mais il existe d'autres formulations qui mériteraient d'être approfondies [Marler and Arora, 2004, Andersson, 2000]. Dans ce cadre, le réglage des pondérations a été effectué de manière empirique, en fixant des valeurs arbitraires aux pondérations et en les modifiant petit à petit selon le résultat observé sur le robot jusqu'à obtenir le comportement désiré. Cette méthode de réglage est assez laborieuse et n'assure pas l'obtention des valeurs de pondérations optimales. Des méthodes basées sur le calcul du front de Pareto pourrait permettre d'obtenir les valeurs optimales des pondérations.

Comme indiqué précédemment, dans les expériences réalisées, le coût de calcul a régulièrement été une contrainte fortement limitante. L'optimisation de la fonction de coût de la commande prédictive n'a pu être réalisée par un algorithme d'optimisation que dans le chapitre IV car la fonction de coût utilisée dans ce chapitre nécessite un temps de calcul faible à chaque appel. Dans les expériences réalisées dans les chapitres V et VI, la fonction de coût intègre en plus le coût de qualité de localisation. Son temps de calcul est trop important pour pouvoir utiliser l'algorithme d'optimisation. Il a donc fallu définir un ensemble fini de séquences de commandes et chercher la valeur minimale sur cet ensemble pour garantir que la solution soit obtenue dans un temps limité. La taille de l'ensemble est très restreinte, la solution calculée est donc sous-optimale par rapport à celle qui serait obtenue en cherchant sur l'ensemble défini par les commandes

extrémales, ensemble de recherche utilisé dans l'algorithme d'optimisation.

Si les robots embarquent des processeurs plus puissants, il serait possible d'utiliser un tel algorithme pour optimiser les fonctions de coût intégrant le coût de qualité de localisation. Cela permettrait d'optimiser sur un ensemble plus important, et donc autoriserait un contrôle plus fin du robot. De plus, le coût de qualité de localisation est évalué une seule fois pour chaque trajectoire prédite, au pas de temps défini par l'horizon de prédiction. Un processeur avec une puissance de calcul plus importante permettrait d'évaluer ce coût sur plusieurs positions futures de la trajectoire prédite. On pourrait ainsi imaginer résoudre un objectif double, par exemple tester la qualité de la localisation à court terme et à long terme. Tester à court terme permet d'éviter que le robot tourne directement vers un mur peu texturé comme il a été montré dans les expériences présentées dans la partie VI.1. Effectuer le test à long terme permet de guider le robot vers des zones bien texturées.

De façon plus large, ces travaux amènent d'autres perspectives concernant la commande, le coût de calcul, la vision et la prédiction de la qualité de localisation à long terme.

Au niveau de la vision, un aspect qui devrait être considéré est la portée du système de stéréovision qui est limitée par la distance entre les deux caméras. Les deux critères développés pour estimer la qualité d'une scène future, par point ou par région, exploitent chacun l'information 3D issue du système stéréo. Même si la méthode par région tente de dépasser cette limite, en prédisant la profondeur des amers non triangulés, cette prédiction est toujours réalisée à partir de données de profondeur obtenues par la stéréovision. Une perspective intéressante serait de chercher à exploiter dans les images l'information qui est située au-delà de la portée du système stéréo mais qui pourrait tout de même être utile pour la prédiction de la qualité de la scène future. On pourrait s'inspirer des travaux de [Saxena et al., 2009], dans lesquels les auteurs cherchent à prédire la profondeur de la scène à partir d'une seule image, en utilisant des techniques d'apprentissage, cf. figure VII.1. D'autres travaux, tels que ceux de [Hadsell et al., 2009] cherchent à classifier des zones lointaines de l'image à partir de leur apparence par apprentissage. La figure VII.2 présente une application à la classification du terrain en termes de traversabilité pour un robot terrestre, dans laquelle l'extension de la zone classifiée est augmentée par le recours à l'apprentissage. On pourrait s'inspirer de ces méthodes pour prédire la qualité pour la localisation de zones plus éloignées et ainsi permettre une prédiction à plus long terme.

Pour étendre la qualité de localisation à plus long terme encore, on peut s'appuyer sur un modèle de l'environnement. Dans les travaux que nous avons réalisés, l'environnement est supposé inconnu a priori. De plus, pendant son déplacement, le robot ne mémorise pas les lieux déjà visités. Par conséquent, si le robot repasse à un endroit qu'il a déjà exploré, il ne peut pas utiliser l'information acquise au premier passage. Il serait donc intéressant de mettre en place un système de cartographie afin de pouvoir utiliser les informations déjà acquises ou connues et permettre l'utilisation de ces informations, par exemple pour effectuer des prédictions à long terme.

Une première solution pourrait consister à généraliser l'approche développée dans

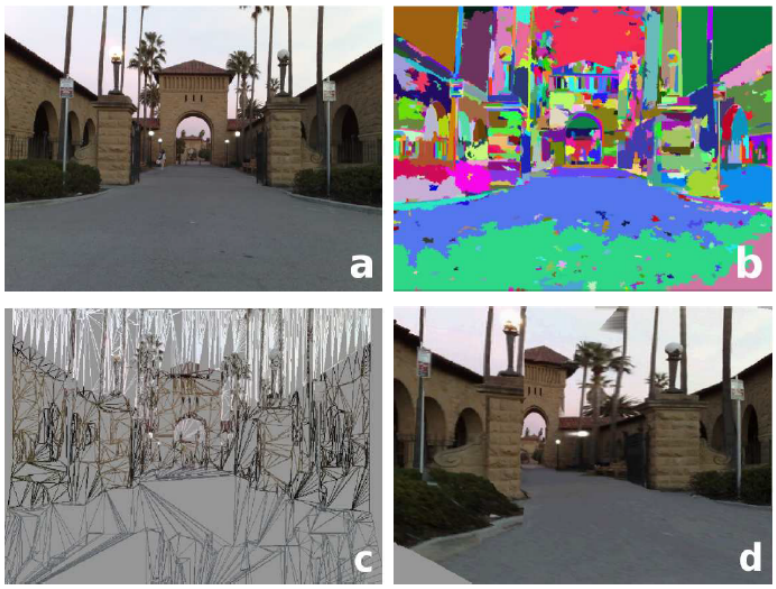


FIGURE VII.1 – Image originale (a), calcul des superpixels (b), modèle 3D prédit par l'algorithme (c), image extraite du modèle 3D texturé (d). Illustration issue de [Saxena et al., 2009].

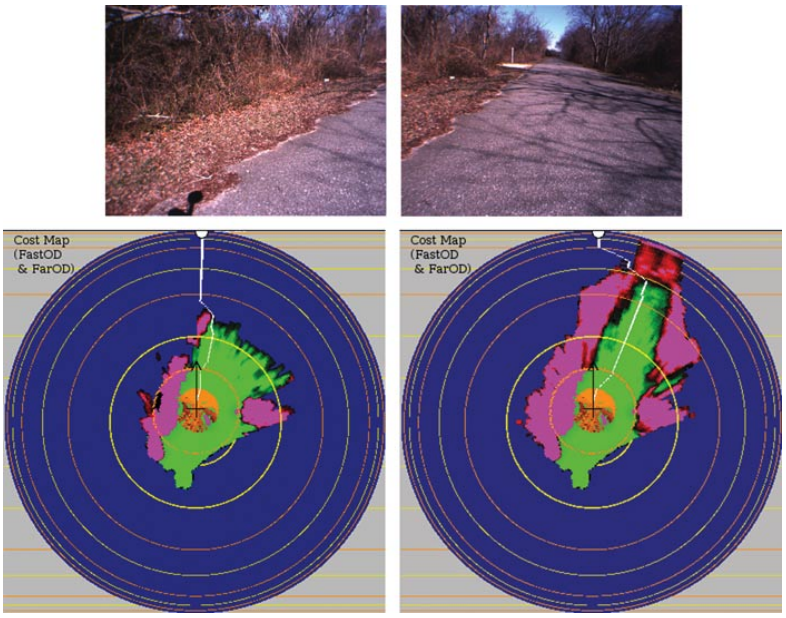


FIGURE VII.2 – Images de l'environnement et résultat de la prédiction à long terme du type de terrain. Illustration issue de [Hadsell et al., 2009].

cette thèse et de calculer un critère de qualité à partir des amers 3D mémorisés. Mais le

temps de calcul du coût de qualité de localisation est proportionnel au nombre d'amers 3D testés, ce qui risque de rapidement donner un coût de calcul beaucoup trop important. Une solution serait de faire un pré-traitement de la liste des amers connus en sélectionnant uniquement ceux qui sont à portée des capteurs. Pour économiser le temps de calcul, il serait également intéressant de regrouper les amers 3D proches en un seul élément afin de limiter le nombre de calculs. Pour aller plus loin, il sera nécessaire d'introduire une étape de modélisation de l'environnement permettant de réduire le stockage nécessaire et le coût d'un calcul de critère de qualité de localisation. Dans cette direction, on pourrait s'inspirer des travaux de [Sadat et al., 2014], dans lesquels l'environnement est reconstruit sous la forme d'un maillage 3D triangulaire et le critère de qualité est lié à la densité de triangles du maillage dans le champ de vision futur. Lorsque la carte de l'environnement est connue au préalable, on peut aussi qualifier les régions de cet environnement par des critères liés à la localisation comme les "Sensory Uncertainty Fields" (une carte de l'incertitude sur la mesure de la localisation) utilisée pour la planification de trajectoire dans [Takeda et al., 1994].

Liste des contributions

[1] H. Roggeman, J. Marzat, M. Sanfourche, and A. Plyer. Embedded vision-based localization and model predictive control for autonomous exploration. In *IROS Workshop on Visual Control of Mobile Robots (ViCoMoR)*, (Chicago, United States), pp. 13–20, 2014.

[2] H. Roggeman, J. Marzat, A. Bernard-Brunel, and G. Le Besnerais. Prediction of the scene quality for stereo vision-based autonomous navigation. In *IAV 2016, 9th IFAC Symposium on Intelligent Autonomous Vehicles*, (Leipzig, Germany), pp. 94–99, 2016.

[3] H. Roggeman, J. Marzat, A. Bernard-Brunel, and G. Le Besnerais. Autonomous exploration with prediction of the quality of vision-based localization. In *IFAC WC 2017*, (Toulouse, France), pp. 10763-10768, 2017.

[4] H. Roggeman, J. Marzat, M. Derome, M. Sanfourche, A. Eudes, and G. Le Besnerais. Detection, estimation and avoidance of mobile objects using stereo-vision and model predictive control. In *ICCV Workshop UAVision2017*, (Venice, Italy), 2017.

[5] H. Roggeman, J. Marzat, A. Bernard-Brunel, and G. Le Besnerais. Predicting stereo-vision based localization quality for safe guidance of an autonomous exploration robot. *Control engineering practice*, 2017 (soumis).

Bibliographie

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11) :2274–2282.

[Ameho et al., 2013] Ameho, Y., Niel, F., Defaÿ, F., Biannic, J.-M., and Bérard, C. (2013). Adaptive control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 5396–5401. IEEE.

[Andersson, 2000] Andersson, J. (2000). A survey of multiobjective optimization in engineering design. *Department of Mechanical Engineering, Linktjping University. Sweden.*

[Bachrach et al., 2012] Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A. S., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2012). Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. *The International Journal of Robotics Research*, 31(11) :1320–1343.

[Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM) : Part II. *IEEE Robotics & Automation Magazine*, 13(3) :108–117.

[Bar-Shalom and Tse, 1976] Bar-Shalom, Y. and Tse, E. (1976). Concepts and methods in stochastic control. *Control and Dynamic Systems : Advances in Theory and Applications*, 12.

[Bascetta et al., 2012] Bascetta, L., Cucci, D., Magnani, G., Matteucci, M., Osmankovic, D., and Tahirovic, A. (2012). Towards the implementation of a MPC-based planner on an autonomous all-terrain vehicle. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012-Workshop on Robot Motion Planning : Online, Reactive, and in Real-time.*

[Bayard and Eslami, 1985] Bayard, D. S. and Eslami, M. (1985). Implicit dual control for general stochastic systems. *Optimal Control Applications and Methods*, 6(3) :265–279.

[Bouabdallah et al., 2004] Bouabdallah, S., Noth, A., and Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2451–2456.

[Bourgault et al., 2002] Bourgault, F., Makarenko, A. A., Williams, S. B., Grocholsky, B., and Durrant-Whyte, H. F. (2002). Information based adaptive robotic exploration. In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems, Lausanne, Switzerland*, volume 1, pages 540–545.

[Bresenham, 1965] Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1) :25–30.

[Bryson and Sukkarieh, 2008] Bryson, M. and Sukkarieh, S. (2008). Observability analysis and active control for airborne SLAM. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1) :261–280.

[Cadena et al., 2016] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6) :1309–1332.

[Chen and Patton, 2012] Chen, J. and Patton, R. J. (2012). *Robust model-based fault diagnosis for dynamic systems*, volume 3. Springer Science & Business Media.

[Crowley, 1989] Crowley, J. L. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE International Conference on Robotics and Automation*, pages 674–680.

[Cvišić and Petrović, 2015] Cvišić, I. and Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. In *European Conference on Mobile Robots (ECMR)*, pages 1–6.

[Davison, 2003] Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, volume 3, pages 1403–1410.

[Davison and Murray, 1998] Davison, A. J. and Murray, D. W. (1998). Mobile robot localisation using active vision. In *European Conference on Computer Vision*, pages 809–825.

[Derome, 2017] Derome, M. (2017). *Vision stéréoscopique temps-réel pour la navigation autonome d'un robot en environnement dynamique.* PhD thesis, Universite Paris-Saclay.

[Derome et al., 2016] Derome, M., Plyer, A., Sanfourche, M., and Le Besnerais, G. (2016). A prediction-correction approach for real-time optical flow computation using stereo. In *German Conference on Pattern Recognition*, pages 365–376.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1) :269–271.

[Dunn et al., 2009] Dunn, E., Van Den Berg, J., and Frahm, J.-M. (2009). Developing visual sensing strategies through next best view planning. In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems, St Louis, MO, USA*, pages 4001–4008.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping : part I. *IEEE robotics & automation magazine*, 13(2) :99–110.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM : Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849.

[Engel et al., 2015] Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct SLAM with stereo cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942.

[Feder et al., 1999] Feder, H. J. S., Leonard, J. J., and Smith, C. M. (1999). Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7) :650–668.

[Feldbaum, 1961] Feldbaum, A. (1960-1961). Dual control theory. I-IV. *Automation and Remote Control*.

[Filatov and Unbehauen, 2004] Filatov, N. M. and Unbehauen, H. (2004). *Adaptive dual control : Theory and applications*, volume 302.

[Findeisen et al., 2003] Findeisen, R., Imsland, L., Allgower, F., and Foss, B. A. (2003). State and output feedback nonlinear model predictive control : An overview. *European journal of control*, 9(2-3) :190–206.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395.

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Appearance-based active, monocular, dense reconstruction for micro aerial vehicle. In *Robotics : Science and Systems (RSS)*.

[Fraundorfer and Scaramuzza, 2012] Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry : Part ii : Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2) :78–90.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11) :1231–1237.

[Goerzen et al., 2010] Goerzen, C., Kong, Z., and Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4) :65.

[Gorecki et al., 2013] Gorecki, T., Piet-Lahanier, H., Marzat, J., and Balesdent, M. (2013). Cooperative guidance of UAVs for area exploration with final target allocation. *IFAC Proceedings Volumes*, 46(19) :260–265.

[Hadsell et al., 2009] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2) :120–144.

[Haner and Heyden, 2011] Haner, S. and Heyden, A. (2011). Optimal view path planning for visual slam. In *Scandinavian Conference on Image Analysis*, pages 370–380.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2) :100–107.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[He et al., 2008] He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a gps-denied environment. In *IEEE International Conference on Robotics and Automation*, pages 1814–1820. IEEE.

[Hemami et al., 1992] Hemami, A., Mehrabi, M., and Cheng, R. (1992). Synthesis of an optimal control law for path tracking in mobile robots. *Automatica*, 28(2) :383–387.

[Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap : an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3) :189—-206.

[Howard, 2008] Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952.

[Howard et al., 2010] Howard, T. M., Green, C. J., and Kelly, A. (2010). Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Field and Service Robotics*, pages 69–78.

[Huang et al., 2005] Huang, S., Kwok, N. M., Dissanayake, G., Ha, Q. P., and Fang, G. (2005). Multi-step look-ahead trajectory planning in SLAM : Possibility and necessity. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1091–1096.

[Hull et al., 1990] Hull, D., Speyer, J., and Burris, D. (1990). Linear-quadratic guidance law for dual control of homing missiles. *Journal of Guidance, Control, and Dynamics*, 13(1) :137–144.

[Jadbabaie et al., 2001] Jadbabaie, A., Yu, J., and Hauser, J. (2001). Unconstrained receding-horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5) :776–783.

[Jones et al., 1993] Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1) :157–181.

[Kanjanawanishkul and Zell, 2009] Kanjanawanishkul, K. and Zell, A. (2009). Path following for an omnidirectional mobile robot based on model predictive control. In *IEEE International Conference on Robotics and Automation*, pages 3341–3346.

[Karaman and Frazzoli, 2011] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7) :846–894.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4) :566–580.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1) :90–98.

[Kim and Eustice, 2013] Kim, A. and Eustice, R. M. (2013). Perception-driven navigation : Active visual SLAM for robotic area coverage. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3196–3203.

[Klančar and Škrjanc, 2007] Klančar, G. and Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and autonomous systems*, 55(6) :460–469.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan*, pages 225–234.

[Krombach et al., 2016] Krombach, N., Droeschel, D., and Behnke, S. (2016). Combining feature-based and direct methods for semi-dense real-time stereo visual odometry. In *International Conference on Intelligent Autonomous Systems*, pages 855–868.

[Künhe et al., 2005] Künhe, F., Gomes, J., and Fetter, W. (2005). Mobile robot trajectory tracking using model predictive control. In *II IEEE latin-american robotics symposium*.

[LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees : A new tool for path planning.

[LaValle, 2006] LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

[Leonard and Durrant-Whyte, 1991] Leonard, J. J. and Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, IROS*, pages 1442–1447.

[Leung et al., 2006] Leung, C., Huang, S., and Dissanayake, G. (2006). Active SLAM using model predictive control and attractor based exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5031.

[Leutenegger et al., 2015] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3) :314–334.

[Li et al., 2013] Li, G., Tamura, Y., Yamashita, A., and Asama, H. (2013). Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning. *IJMA*, 3 :141–170.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110.

[Maimone et al., 2007] Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3) :169–186.

[Makarenko et al., 2002] Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). An experiment in integrated exploration. In *Proceedings of the IEEE/RSJ International Conference on Robots and Systems, Lausanne, Switzerland*, volume 1, pages 534–539.

[Mallet et al., 2000] Mallet, A., Lacroix, S., and Gallo, L. (2000). Position estimation in outdoor environments using pixel tracking and stereovision. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3519–3524.

[Marler and Arora, 2004] Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6) :369–395.

[Marzat et al., 2017] Marzat, J., Bertrand, S., Eudes, A., Sanfourche, M., and Moras, J. (2017). Reactive MPC for autonomous MAV navigation in indoor cluttered environments : Flight experiments. In *Proceedings of the 20th IFAC World Congress, Toulouse, France*, pages 16566–16572.

[Mostegel et al., 2014] Mostegel, C., Wendel, A., and Bischof, H. (2014). Active monocular localization : Towards autonomous monocular exploration for multirotor MAVs. In *Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China*, pages 3848–3855.

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE international conference on Robotics and automation*, pages 3565–3572.

[Murray-Smith et al., 2003] Murray-Smith, R., Sbarbaro, D., Rasmussen, C. E., and Girard, A. (2003). Adaptive, cautious, predictive control with gaussian process priors.

[Nistér et al., 2006] Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1) :3–20.

[Orsag et al., 2015] Orsag, M., Haus, T., Palunko, I., and Bogdan, S. (2015). State estimation, robust control and obstacle avoidance for multicopter in cluttered environments : Euroc experience and results. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 455–461.

[Pharpatara, 2015] Pharpatara, P. (2015). *Trajectory planning for aerial vehicles with constraints*. PhD thesis, Universite Paris-Saclay.

[Plyer et al., 2016] Plyer, A., Le Besnerais, G., and Champagnat, F. (2016). Massively parallel lucas kanade optical flow for real-time video processing applications. *Journal of Real-Time Image Processing*, 11(4) :713–730.

[Rathouskỳ and Havlena, 2013] Rathouskỳ, J. and Havlena, V. (2013). Mpc-based approximate dual controller by information matrix maximization. *International Journal of Adaptive Control and Signal Processing*, 27(11) :974–999.

[Ratliff et al., 2009] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). CHOMP : Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494.

[Rochefort et al., 2014] Rochefort, Y., Piet-Lahanier, H., Bertrand, S., Beauvois, D., and Dumur, D. (2014). Model predictive control of cooperative vehicles using systematic search approach. *Control Engineering Practice*, 32 :204–217.

[Roggeman et al., 2016] Roggeman, H., Marzat, J., Bernard-Brunel, A., and Le Besnerais, G. (2016). Prediction of the scene quality for stereo vision-based autonomous navigation. In *9th IFAC Symposium on Intelligent Autonomous Vehicles IAV*, pages 94–99, Leipzig, Germany.

[Roggeman et al., 2017a] Roggeman, H., Marzat, J., Bernard-Brunel, A., and Le Besnerais, G. (2017a). Autonomous exploration with prediction of the quality of vision-based localization. In *IFAC WC 2017*, Toulouse, France.

[Roggeman et al., 2017b] Roggeman, H., Marzat, J., Bernard-Brunel, A., and Le Besnerais, G. (2017b). Predicting stereo-vision based localization quality for safe guidance of an autonomous exploration robot. *Control engineering practice*.

[Roggeman et al., 2017c] Roggeman, H., Marzat, J., Derome, M., Sanfourche, M., Eudes, A., and Le Besnerais, G. (2017c). Detection, estimation and avoidance of mobile objects using stereo-vision and model predictive control. In *ICCV Workshop UAVision2017*.

[Roggeman et al., 2014] Roggeman, H., Marzat, J., Sanfourche, M., and Plyer, A. (2014). Embedded vision-based localization and model predictive control for autonomous exploration. In *IROS Workshop on Visual Control of Mobile Robots (ViCoMoR)*, pages 13–20, Chicago, United States.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision, Graz, Austria*, volume 1, pages 430–443.

[Roy et al., 1999] Roy, N., Burgard, W., Fox, D., and Sebastian, T. (1999). Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 35–40.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB : An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3D is here : Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–4.

[Ryoo et al., 2005] Ryoo, C.-K., Cho, H., and Tahk, M.-J. (2005). Optimal guidance laws with terminal impact angle constraint. *Journal of Guidance Control and Dynamics*, 28(4) :724–732.

[Sadat et al., 2014] Sadat, S. A., Chutskoff, K., Jungic, D., Wawerla, J., and Vaughan, R. (2014). Feature-rich path planning for robust navigation of MAVs with mono-SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China*, pages 3870–3875.

[Sanfourche et al., 2013] Sanfourche, M., Vittori, V., and Le Besnerais, G. (2013). eVO : A realtime embedded stereo odometry for MAV applications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan*, pages 2107–2114.

[Saxena et al., 2009] Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3d : Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5) :824–840.

[Scaramuzza et al., 2014] Scaramuzza, D., Achtelik, M. C., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., Achtelik, M. W., Chli, M., Chatzichristofis, S., Kneip, L., et al. (2014). Vision-controlled micro flying robots : from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics & Automation Magazine*, 21(3) :26–40.

[Scaramuzza and Fraundorfer, 2011] Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4) :80–92.

[Shaferman and Shima, 2008] Shaferman, V. and Shima, T. (2008). Linear quadratic guidance laws for imposing a terminal intercept angle. *Journal of Guidance, Control, and Dynamics*, 31(5) :1400.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Jerusalem, Israel*, pages 593–600.

[Sim and Roy, 2005] Sim, R. and Roy, N. (2005). Global a-optimal robot exploration in SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain*, pages 661–666.

[Singh and Fuller, 2001] Singh, L. and Fuller, J. (2001). Trajectory generation for a UAV in urban terrain, using nonlinear MPC. In *Proceedings of the American Control Conference*, volume 3, pages 2301–2308.

[Sizintsev and Wildes, 2010] Sizintsev, M. and Wildes, R. P. (2010). Coarse-to-fine stereo vision with accurate 3D boundaries. *Image and Vision Computing*, 28(3) :352–366.

[Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer.

[Song and Um, 1996] Song, T. L. and Um, T. Y. (1996). Practical guidance for homing missiles with bearings-only measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 32(1) :434–443.

[Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. In *Robotics : Science and Systems*, volume 2, pages 65–72.

[Takeda et al., 1994] Takeda, H., Facchinetti, C., and Latombe, J.-C. (1994). Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10) :1002–1017.

[Tokekar et al., 2014] Tokekar, P., Karnad, N., and Isler, V. (2014). Energy-optimal trajectory planning for car-like robots. *Autonomous Robots*, 37(3) :279–300.

[Ucinski, 2000] Ucinski, D. (2000). Optimal sensor location for parameter estimation of distributed processes. *International Journal of Control*, 73(13) :1235–1248.

[Umeyama, 1991] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4) :376–380.

[Unbehauen, 2000] Unbehauen, H. (2000). Adaptive dual control systems : a survey. In *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 171–180.

[Valencia et al., 2011] Valencia, R., Andrade-Cetto, J., and Porta, J. M. (2011). Path planning in belief space with pose slam. In *2011 IEEE International Conference on Robotics and Automation*, pages 78–83. IEEE.

[Vidal-Calleja et al., 2006] Vidal-Calleja, T., Davison, A. J., Andrade-Cetto, J., and Murray, D. W. (2006). Active control for single camera SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, FL, USA*, pages 1930–1936.

[Walter and Pronzato, 1997] Walter, E. and Pronzato, L. (1997). *Identification of parametric models from experimental data.* Springer Verlag.

[Weiss et al., 2012] Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 957–964. IEEE.

[Wittenmark, 1995] Wittenmark, B. (1995). Adaptive dual control methods : An overview. In *IFAC Symposium on Adaptive Syst. in Control and Signal Proc*, pages 67–72.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151.