Titre: Title:	Système embarqué et coopératif de localisation et de perception pour un robot mobile
Auteur: Author:	Sylvain Marleau
Date:	2006
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Marleau, S. (2006). Système embarqué et coopératif de localisation et de perception pour un robot mobile [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7727/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7727/
Directeurs de recherche: Advisors:	Richard Gourdeau, & Richard Hurteau
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

Système embarqué et coopératif de localisation et de perception pour un robot mobile

Sylvain Marleau
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
février 2006

Your file Votre référence
ISBN: 978-0-494-17958-1
Our file Notre référence
ISBN: 978-0-494-17958-1

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

Système embarqué et coopératif de localisation et de perception pour un robot mobile

présenté par: Sylvain Marleau

en vue de l'obtention du diplôme de: Maître ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

DeSantis Romano, président
Gourdeau Richard, membre et directeur de recherche
Hurteau Richard, membre et codirecteur de recherche
Baron Luc, membre

Remerciements

Un projet d'envergure tel qu'un projet de maîtrise n'est jamais réalisé sans l'aide de tierces personnes et je tiens à remercier toutes ces personnes qui, de près ou de loin, m'ont épaulé pendant ma maîtrise. Je désire toutefois remercier plus particulièrement mon directeur de recherche, M. Richard Gourdeau pour l'aide très appréciée qu'il m'a apporté, tant sur le plan technique que sur le plan financier. Je remercie aussi mon co-directeur, M. Richard Hurteau pour avoir mis à ma disposition les robots de son laboratoire de mécatronique et pour son soutient financier. Je désire remercier mon collègue Julien Beaudry avec qui j'ai travaillé pendant la réalisation du projet. Par sa détermination, Julien a su assurer le bon fonctionnement des robots et montrer leur efficacité en les menant à la compétition internationale de soccer robotisé, la *Robocup*. Je tiens à remercier aussi les autres collègues avec qui j'ai travaillé dont plus particulièrement, Vincent Zalzal, Alexandre Venne et Liko-Paul Pinsonnault. De plus, je désire remercier Pierre-Yves Mailhot et ma copine Marie-Hélène Murray pour leurs encouragements et pour avoir participé à la correction de ce mémoire. Finalement, je remercie mes inconditionnels supporteurs, mes parents, dont les encouragements ont toujours été grandement appréciés.

Résumé

Pour être automone dans l'exécution d'une tâche, un robot doit percevoir son environnement, reconnaître certains objets et se localiser. Ce mémoire présente un système de perception développé pour permettre à un robot de se localiser à l'aide de balises disposées dans un environnement contrôlé et connu, d'estimer la position et le déplacement d'un objet et de partager des informations avec les autres robots présents dans le même environnement afin que chacun puisse bénéficier de la perception des autres. Les capteurs utilisés sont des encodeurs optiques pour fournir des mesures d'odométrie et une caméra omnidirectionnelle afin de fournir des mesures de la position relative des objets présents dans l'environnement. Un système de vision a donc été développé afin de reconnaître en temps réel des objets dans une image. Une caméra omnidirectionnelle a été utilisée, car elle permet de produire en un seul cliché une image panoramique permettant de voir l'environnement situé tout autour du robot.

La reconnaissance en temps réel d'un objet dans une image doit être à la fois rapide et robuste. Les objets recherchés sont par hypothèse de formes simples (cylindriques, sphériques ou rectangulaires), de tailles connues et de couleurs uniformes, constrastantes et définies. La recherche d'un objet est basée sur la prédiction de sa position dans l'image. Ainsi seule la portion de l'image dans laquelle l'objet est attendu est traitée afin d'alléger le traitement et augmenter la robustesse. La taille de cette portion de l'image est déterminée en fonction de la précision de la prédiction.

L'image est représentée en HSI afin de permettre une segmentation couleur simple et efficace. Une segmentation est effectuée pour regrouper les pixels dont la couleur est contenue dans un intervalle attendu pour un objet donné. Les zones résultant

de la segmentation sont filtrées en comparant leur taille et leur aire avec celles qui avaient été prédites préalablement. Généralement, cette technique est suffisante pour identifier des objets dont la taille et la couleur diffèrent largement de celles des autres objets présents dans l'environnement.

Le système de localisation développé estime la position du robot en combinant de manière optimale les mesures asynchrones d'odométrie et de vision. Les mesures d'odométrie ont les avantages d'être précises pour un court déplacement et elles peuvent être acquisitionnées rapidement. Toutefois, pour déterminer la position du robot à partir de celles-ci, il faut additionner chaque mesure aux mesures précédentes ce qui cumule les imprécisions. La position estimée diverge alors rapidement de la position réelle. Bien que les mesures de vision soient moins précises, moins fréquentes et qu'elles demandent beaucoup plus de traitement, elles permettent de déterminer en une seule itération la position absolue du robot dans son environnement. Cette position estimée est peu précise, mais combinée par un filtre de *Kalman* avec les mesures d'odométrie et les mesures de vision effectuées précédemment, cette estimation devient très précise. Le système de localisation a aussi la capacité d'estimer et de corriger le biais entre l'orientation de la caméra et l'orientation du robot.

Le système de perception d'objets dynamiques a été développé pour estimer la position d'un objet qui peut se mouvoir et dont la position n'est pas nécessairement connue initialement. Le système estime la position et la vitesse de l'objet à partir des mesures de vision en relation avec la prévision de la dynamique de l'objet. Le bruit sur les mesures est filtré en fonction des accélérations attendues pour l'objet. Ainsi, la précision de la position estimée d'un objet subissant de faibles accélérations est grandement améliorée. Le filtre de *Kalman* utilisé fournit aussi une estimation de l'erreur sur la position de l'objet. Cette erreur est fonction de la dynamique prévue et de la position de l'objet par rapport au robot. Ainsi, l'erreur estimée est plus grande si l'objet est éloigné ou si sa dynamique est élevée. Une meilleure estimation de l'erreur permet une reconnaissance plus efficace de l'objet par le système de vision.

La vitesse estimée par le filtre est plus précise que la vitesse calculée par la varia-

tion de la position dans un intervalle de temps. Cette estimation permet de prédire la position de l'objet et la direction du déplacement. Un coefficient de frottement visqueux est utilisé pour modéliser une accélération négative en fonction de la vitesse de l'objet. Ce coefficient peut être estimé par le filtre et son utilisation permet une prédiction plus précise de la position et de la vitesse de l'objet.

Le partage d'informations entre les robots permet la mise en commun d'informations relatives à un même objet afin d'améliorer la précision de la perception des robots. La précision de la perception de chacun des robots après le partage et la mise en commun est similaire à celle du robot ayant la meilleure précision avant le partage. En plus d'améliorer la perception, le partage d'informations permet à un robot d'estimer l'état d'un objet à partir de l'information reçue même si aucune mesure n'est fournie par le système de vision. Il suffit qu'un seul robot mesure la position d'un objet pour que tous les autres robots en connaissent aussi la position. Lors d'une mise en commun, l'apport d'une information reçue est pondéré en fonction de la précision estimée par rapport à celles des autres informations. Ainsi, les informations plus précises sont privilégiées.

Pour mettre en commun plusieurs informations asynchrones, il est important de les synchroniser. Au préalable, les robots et le serveur de partage d'informations sont synchronisés entre eux afin d'utiliser le même référentiel temporel. Avant une mise en commun, les informations les moins récentes sont actualisées à l'aide d'une prédiction basée sur le modèle dynamique de l'objet concerné. Cette prédiction permet d'éliminer le retard engendré par le délai de traitement et de transmission des robots. Afin d'évaluer la précision de l'information prédite, les erreurs associées à l'information sont prédites aussi. Ceci permet ensuite de pondérer l'apport de chaque information lors de la mise en commun.

Abstract

To make a robot autonomous while executing a task, it is necessary to allow it to perceive its environment, to recognize certain objects and to localize itself. This work presents a perception system which allows a robot to localize itself from landmarks placed in a known and controlled environment, to estimate position and motion of an object and to share data with other robots in the same environment so that each one can take advantage of other's perception. The sensors used are optical encoders to measure robot's odometry and an omnidirectional camera to return measurements of relative position of objects around the robot. A vision system has also been developed to carry out real-time objects recognition. An omnidirectional camera has been used because it allows to take in one shot an panoramic picture of all environment around the robot.

The real-time object recognition should be fast and robust. Searched objects have by assumption simple shapes (cylindrical, spherical or rectangular), known size and uniform, contrasting and defined colours. Object search is based on the prediction of its position into the image. Then, only the image's subset in which the object should be is computed. This reduces the processing and increases the reliability. The size of an image's subset is function of the prediction's precision.

Images are encoded in HSI because it allows simple and efficient colour segmentation. Image segmentation is done to group together pixels which colour value is in the expected interval for a given object. Resulting regions are validated by comparing their size and their area with expected ones. Most of the time, this algorithm is able to identify objects which size and colour are very different of other objects in the robot's environment.

The localization system developed estimates robot's position by combining asynchronous odometry and vision measurements. Odometry measurements are very precise for short displacement and they can be taken very rapidly. However, to calculate the robot's position, each measurement should be added to the precedent. Then, imprecision are cumulated and the estimated robot's position diverges rapidly. Vision measurements are less precise, less frequent and they take a lot of processing time. But, they allow to calculate the absolute robot's position in one iteration. Combined by a *Kalman* filter to odometry measures and earlier vision measures, estimated position can be very precise. Furthermore, the localization system can estimate and correct the difference between camera and robot orientation.

The perception system of dynamic objects has been developed to estimate position of a moving object for which initial position is unknown. The system estimate object's position and speed with vision measurements in relation to the expected object's dynamic. Measure's noise is then filtered in function of object's acceleration. Position accuracy of an object which has low acceleration is greatly increased. The *Kalman* filter used estimates the object's position and its accuracy. Accurancy is function of vision's measurements accurancy, expected dynamic and object's distance from the robot. Then, estimated accuracy is less if object is far or if its expected dynamic is high. Better accuracy estimation makes objects recognition more efficient.

The object's speed estimated by the filter is more precise then the speed calculated by the position variation in a given time. Estimation of object's speed allows to predict its position and the motion direction. A viscous friction coefficient is used to model a negative acceleration proportional to the object's speed. This coefficient can be estimated by the filter and it allows to predict more precisely the object's position and speed.

Data sharing concerning a same object allows robots to improve their perception accuracy. Accuracy for each robot after the data sharing is similar to the one of the robot having the better accuracy before the sharing. Data sharing improves percep-

tion and allows robot to estimate the state of an object even if no measurement is available from the vision system. Only one robot has to measure an object's position so that all robots know the position. When data is put together, information received is weighed proportionally to their accuracy. Then, more accurate data are favoured.

To share efficiently asynchronous data, it is very important to synchronize them. Robots and server timers are synchronised first. Before date is put together, older data are brought up to date by a prediction based on the dynamic model of the concerned object. This prediction counterbalances the processing and transmission delays. To evaluate the data's precision, accuracies are also predicted too. This allows to weigh each data while they are put together.

Table des matières

Liste des figures

Liste des tableaux

Liste des sigles et abréviations

α	Angle horizontal entre le devant de la caméra et un objet.
α_{min}	Limite inférieure horizontale de la zone de recherche.
α_{max}	Limite supérieure horizontale de la zone de recherche.
$\Delta\alpha_{min}$	Largeur minimale de l'objet.
$\Delta\alpha_{max}$	Largeur maximale de l'objet.
β	Angle vertical entre l'horizon et un objet.
β_{min}	Limite inférieure verticale de la zone de recherche.
β_{max}	Limite supérieure verticale de la zone de recherche.
$\Delta\beta_{min}$	Hauteur minimale de l'objet.
$\Delta\beta_{max}$	Hauteur maximale de l'objet.
x	Position en x du robot ou d'un objet.
y	Position en y du robot ou d'un objet.
θ	Orientation du robot.
x_n	Position en x d'une balise.
y_n	Position en y d'une balise.
$\hat{x}$	Position en x estimée du robot ou d'un objet.
$\hat{y}$	Position en y estimée du robot ou d'un objet.
$\hat{\theta}$	Orientation estimée du robot.
$\hat{b}$	Biais estimé entre l'orientation de la caméra et celle du robot.
x_{camera}	Position en x de la caméra.
y_{camera}	Position en y de la caméra.
z_{camera}	Hauteur de la caméra.
θ_{camera}	Orientation de la caméra.

x_{objet}	Position en x de l'objet.
y_{objet}	Position en y de l'objet.
z_{objet}	Hauteur de la base de l'objet.
θ_{objet}	Orientation de la normale d'un objet rectangulaire seulement.
$Hauteur$	Hauteur de l'objet.
$Largeur$	Largeur de l'objet.
$\Delta Hauteur$	Tolérance sur la hauteur de l'objet.
$\Delta Largeur$	Tolérance sur la largeur de l'objet.
E_P	Estimation de l'erreur combinée sur la position du robot et de l'objet.
E_O	L'erreur liée à l'orientation du robot.
E_E	L'erreur liée à l'étalonnage vertical de la caméra.
$\Delta\phi$	Déplacement angulaire d'une roue.
Δv	Déplacement tangentiel mesuré par odométrie.
$\Delta\omega$	Déplacement angulaire mesuré par odométrie.
Δt	Temps écoulé depuis la dernière mise à jour.
B_v	Bruit sur le déplacement tangentiel en fonction du déplacement effectué.
B_w	Bruit sur le déplacement angulaire en fonction du déplacement effectué.
B_g	Bruit sur le glissement latéral en fonction du déplacement effectué.
B_{vs}	Bruit sur le déplacement tangentiel en fonction du temps écoulé.
B_{ws}	Bruit sur le déplacement angulaire en fonction du temps écoulé.
B_{gs}	Bruit sur le glissement latéral en fonction du temps écoulé.
B_b	Bruit de processus sur l'estimation du biais.
B_z	Bruit sur les mesures de vision.
B_{vx}	Bruit sur la vitesse en x temps écoulé.
B_{vy}	Bruit sur la vitesse en y temps écoulé.
B_ζ	Bruit sur le coefficient de frottement visqueux en fonction du temps écoulé.
B_α	Bruit sur une mesure α.
B_β	Bruit sur une mesure β.
B_x	Erreur sur la position de la caméra en x.
B_y	Erreur sur la position de la caméra en y.
B_θ	Erreur sur l'orientation de la caméra.

d_o Distance entre le point le plus rapporché et le centre de l'objet.

h_o Hauteur du centre de l'objet.

t_s Écart entre le minuteur du serveur et le minuteur local.

t_d Délai de transmission et de traitement d'une requête.

ζ Estimation du coefficient de frottement visqueux.

Introduction

L'automatisation et la robotique sont des domaines de plus en plus familiers. Chaque jour, nous utilisons plusieurs outils automatisés qui facilitent l'exécution de nos tâches. La plupart des tâches automatisées sont très spécifiques, mais peu à peu elles se complexifient et les robots deviennent plus polyvalents et plus autonomes. Les robots ainsi que les algorithmes de contrôle, de perception et d'intelligence artificielle deviennent de plus en plus raffinés et performants.

La robotique devient donc une discipline qui regroupe plusieurs domaines de compétences. Pour qu'un robot soit puissant, rapide, précis et autonome, il doit avoir une plateforme mécanique performante, des algorithmes de contrôle robuste, une perception rapide et précise ainsi qu'une intelligence artificielle développée. De plus, ce robot doit coopérer avec les autres robots présents pour exécuter sa tâche efficacement. Tout cela nécessite énormément de développement. Rendre autonome un simple robot mobile consiste en une tâche complexe. Développer les algorithmes permettant à ce robot d'avoir une connaissance minimale de son environnement et d'être en mesure d'intéragir avec celui-ci, représente un grand défi.

Ce projet porte essentiellement sur le développement et la mise en oeuvre d'un système embarqué de perception adapté à un robot mobile. Ce mémoire présente une solution permettant à un robot d'identifier des objets contrastants dans son environnement, d'utiliser la position relative de ces objets ainsi que des mesures d'odométrie pour trouver de façon optimale sa position, d'identifier et de localiser certains objets dans son environnement et de communiquer ces informations de manière optimale aux autres robots avec lesquels il interagit. En plus des algorithmes utilisés, ce mémoire présente les difficultés rencontrées, les limitations et leurs causes respectives.

La localisation d'un robot peut être déterminée de différentes façons. L'ouvrage [1] présente la localisation à partir de satellites GPS différentiels, de balises hyperfréquences, de balises infrarouges et de balises visuelles (voir aussi [2] et [3]). Plusieurs types de capteurs peuvent être utilisés tel que les caméras, les sonars, les capteurs infrarouge de proximité et les scanners laser (voir [4]). La localisation présentée dans ce mémoire est basée sur des balises visuelles identifiées dans l'image retournée par une caméra omnidirectionnelle. L'utilisation d'une caméra omnidirectionnelle pour localiser n'est pas une innovation. Plusieurs articles dont, [5], [6] et [7], présentent des solutions similaires. Toutefois, la solution utilisée permet de combiner les mesures des balises visuelles avec les mesures d'odométrie afin d'obtenir une localisation plus précise.

Chapitre 1

Contexte de mise en oeuvre

1.1 Soccer robotisé

Malgré son côté ludique, le soccer robotisé est une excellente plateforme de développement en robotique, car en plus de présenter plusieurs défis techniques complexes, il jouit de la très grande popularité de ce sport à travers le monde. Pour avoir un bon joueur de soccer robotisé, il faut une plateforme mécanique performante, une perception rapide et précise ainsi qu'une intelligence artificielle adaptée et efficace pour toutes les situations de jeux. De plus, pour que l'équipe dans son ensemble soit performante, tous les joueurs doivent être en mesure de s'entraider en se positionnant efficacement et en se communiquant de l'information sur l'environnement et leur comportement. Une équipe performante de robots joueurs de soccer pourra ensuite être utilisée dans n'importe quel autre environnement dynamique partiellement contrôlée.

1.1.1 Compétition internationale Robocup

Il existe une compétition internationale de robots joueurs de soccer appelée *Robocup*. Les objectifs de cette compétition sont de favoriser la recherche et l'intégration des plus récentes technologies dans les domaines de la robotique, de l'intelligence et de la vision artificielle. De plus, cette compétition tente d'initier la population mondiale aux nouvelles percées technologiques et aux capacités toujours grandissantes de la robotique. L'objectif ultime de la compétition est qu'en 2050, une équipe de robots humanoïdes entièrement autonomes puisse rivaliser avec les champions du monde.

Présentement, il existe 4 catégories différentes de robots joueurs de soccer:

Small size league Des robots mobiles de petite taille (environ 20 cm de diamètre) jouant avec une balle de ping pong et dépourvus de vision embarquée. La perception est assurée par plusieurs caméras disposées au dessus de la surface de jeux.

Middle size league Des robots mobiles de taille moyenne (environ 50 cm de diamètre par 80 cm de haut) jouant avec un ballon Fifa taille 5 et dont toutes les capacités de perception sont embarquées. Cette catégorie présente un défi plus important au niveau du traitement temps réel et embarqué de l'intelligence artificielle et de la perception.

Four legged Les robots utilisés dans cette catégorie sont les chiens Sony *Aïbo* reprogrammés pour adopter des comportements de joueurs de soccer. La perception présente le plus grand défi de cette catégorie, car elle est entièrement basée sur une caméra unidirectionnelle, située dans la tête du robot, dont la position et l'orientation est fonction de sa posture.

Humanoid Des robots humanoïdes, généralement de petites tailles, dont la stabilisation et le contrôle sont pour l'instant les principaux enjeux. Ainsi, dans bien des cas, la perception et la rapidité de ces robots laisse à désirer.

Il existe d'autres catégories junior dont les objectifs sont de favoriser l'initiation et l'accessibilité des jeunes à la robotique par l'assemblage et la programmation de robots jouets.

1.1.2 Environnement de jeux

Les robots qui ont été utilisés pour réaliser toutes les expérimentations de ce projet sont conformes à la catégorie des robots de taille moyenne. Aussi, dans le but de permettre leur participation à la compétition, l'environnement de jeux de la *Robocup* a été utilisé pour caractériser l'environnement semi-contrôlé du système de perception décrit dans ce mémoire.

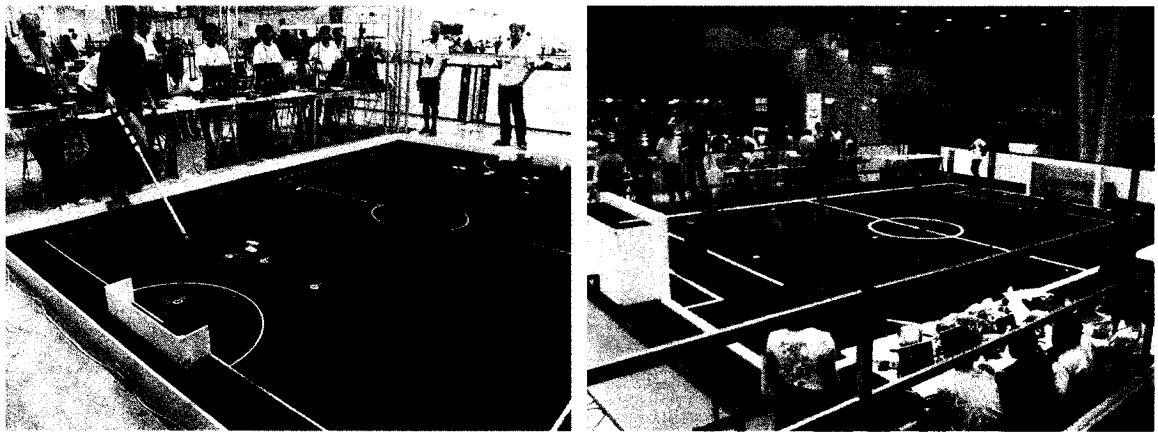

Figure 1.1: Catégories *Small Size* et *Middle Size*

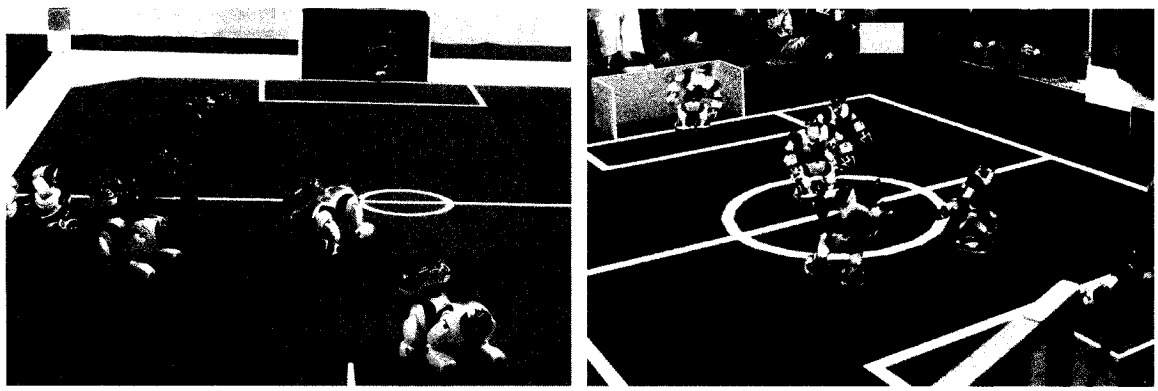

Figure 1.2: Catégories *Four Legged* et *Humanoid*

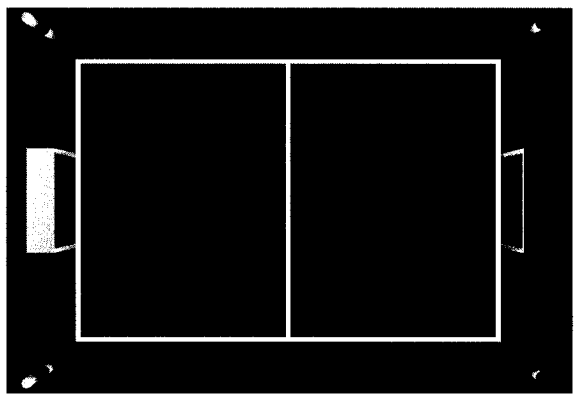

Figure 1.3: Environnement de la catégorie *Middle Size*

L'environnement des robots de taille moyenne est constitué d'un terrain vert avec des lignes blanches délimitant la surface de jeux, de 4 balises cylindriques à trois couleurs placées dans chacun des coins du terrain et de deux buts dont le fond opaque est aussi de couleur uniforme et constrastante jaune ou bleu. De plus, le ballon utilisé est le ballon orange de taille 5. À l'extérieur de la surface de jeu, l'environnement n'est pas défini et il peut être composé de participants, de spectateurs et d'affiches publicitaires. L'éclairage est constant et uniforme sur toute la surface de jeux. La figure 1.3 présente cet environnement.

1.2 Plateforme d'expérimentation

Le laboratoire de mécatronique de l'École Polytechnique de Montréal a développé les six robots mobiles utilisés pour réaliser la mise en oeuvre du projet. Ces robots ont été développés au départ dans le cadre du cours *Projet de génie électrique.* Par la suite, le groupe Robofoot a poursuivi le développement des robots et leur programmation dans le but de participer à la compétition *Robocup* décrite précédemment.

1.2.1 Description de la plateforme

La plateforme utilisée est un robot mobile à deux roues motrices. La figure 1.4 présente l'allure et les principales composantes du robot. L'unité de traitement, un ordinateur embarqué, permet d'effectuer tous les algorithmes de perception, d'intelligence artificielle et de contrôle de haut niveau directement sur le robot. Les deux moteurs sont commandés en vitesse via la carte de contrôle qui, en plus d'alimenter les moteurs, assure le respect des vitesses demandées.

1 Un miroir parabolique convexe est supporté dans le haut du montage de vision afin de permettre une vision omnidirectionnnelle.

2 Une webcam dirigée vers le miroir du dessus fait office de caméra pour la vision omnidirectionnelle. La caméra et le miroir sont encapsulés dans un cylindre d'acrylique transparent pour les protéger des impacts et éviter d'avoir des tiges qui obstruent certaines zones de la vision omnidirectionnelle.

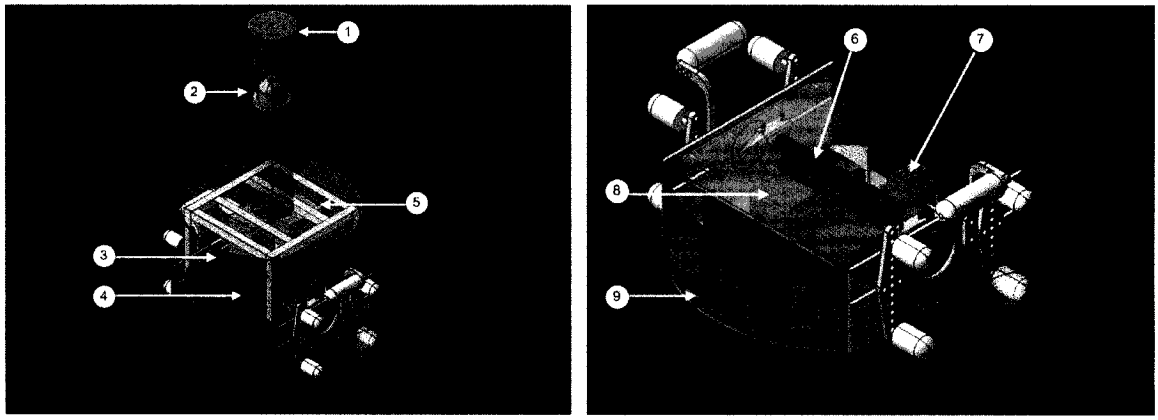

Figure 1.4: Composantes principales de la plateforme

3 L'ordinateur embarqué, un *Pentium Celeron 566 MHz ou Pentium III 800 MHz*, exécute sous *Linux* une application C++ comprenant tous les algorithmes de perception, d'intelligence artificielle et de contrôle de haut niveau. La carte de contrôle et la carte d'alimentation (*Convertisseur DCDC*) sont empilées sur le port *PC104* de l'ordinateur.

4 Une bonbonne d'air comprimé, un régulateur de pression et deux valves pneumatiques actionnent les pistons pneumatiques du botteur.

5 Un lien de communication *Ethernet* sans-fil permet de programmer et de contrôler le robot à distance en plus de permettre aux robots de communiquer entre eux.

6 Les pistons pneumatiques poussent une coupole pour botter le ballon. Un ressort permet de rentrer le piston lorsque la pression d'air est relâchée.

7 Les deux moteurs à courant continu 24 V sont alimentés et controlés par la carte de contrôle. De plus, un encodeur optique est intégré à chacun des moteurs pour permettre l'asservissement en position des roues. Les deux mesures de position fournies par les encodeurs permettent d'obtenir les déplacements tangentiel et angulaire du robot appelés aussi odométrie.

8 L'alimentation électrique du robot est réalisée par deux batteries acide-plomb de 12V placées en série.

9 Les deux roues sur les côtés permettent de mouvoir le robot alors que deux petites roues folles situées à l'avant et l'arrière assure sa stabilité.

1.2.2 Caractéristiques et performances mécaniques

Le tableau 1.1 fournit les vitesses et les accélérations maximales du robot. Ces valeurs sont approximatives étant donné qu'elles varient considérablement en fonction de la capacité des batteries. De plus, les accélérations maximales sont données pour une vitesse nulle, car l'accélération est nulle lorsque le robot roule à sa vitesse maximale. La vitesse maximale d'une roue correspond à la vitesse tangentielle maximale du robot ce qui fait en sorte que les vitesses tangentielles et angulaires maximales ne peuvent être atteintes en même temps. Ces caractéristiques n'influencent pas directement le système de perception décrit dans ce mémoire, elles sont considérées plutôt comme des objectifs de performances que le système de perception devrait atteindre.

Le tableau 1.2 présente quelques caractérisques pertinentes du robot afin de bien comprendre le contexte de mise en oeuvre.

Tableau 1.1: Performances du robot

Caractéristiques	***Valeurs***
Accélération tangentielle maximale	$2m/s^2$
Accélération angulaire maximale	$6rad/s^2$
Vitesse tangentielle maximale	$2m/s$
Vitesse angulaire maximale	$15rad/s$

1.2.3 Montage de la caméra omnidirectionnelle

Une caméra omnidirectionnelle offre un important avantage par rapport à une caméra régulière puisque son champ de vision est de 360°. Ceci permet de voir tout ce qui entoure le robot en une seule image. Ce type de caméra est maintenant utilisé dans plusieurs domaines dont la robotique, la surveillance, la vidéo conférence et plusieurs autres (voir [8]). Le montage de la figure 1.5 présente le montage de la caméra omnidirectionnelle des robots utilisés pour l'expérimentation du système. Il s'agit d'une

Tableau 1.2: Caractéristiques du robot

Caractéristiques	***Valeurs***
Poids	$20kg$
Hauteur	$0,79m$
Largeur	$0,48m$
Rayon d'une roue (R_{roue})	$0,062m$
Distance entre les deux roues (D_{roue})	$0,334m$
Résolution des encodeurs optiques	$23000impulsions/tour$

caméra orientée vers un miroir convexe disposé dans le haut du montage. L'image ainsi captée est la réflexion de tous les objets situés autour du robot (figure 1.6). Le montage est placé le plus haut possible au centre du robot afin de favoriser la vision des objets éloignés et faciliter la correspondance entre la caméra et la position du centre du robot. Le miroir est supporté par une vis au centre que l'on peut clairement distinguer au milieu de l'image. Le miroir et la caméra sont encapsulés dans un cylindre d'acrylique afin de les protéger des impacts et d'éviter que les composantes de la structure supportant le miroir n'obstruent certaines portions du champ de vision.

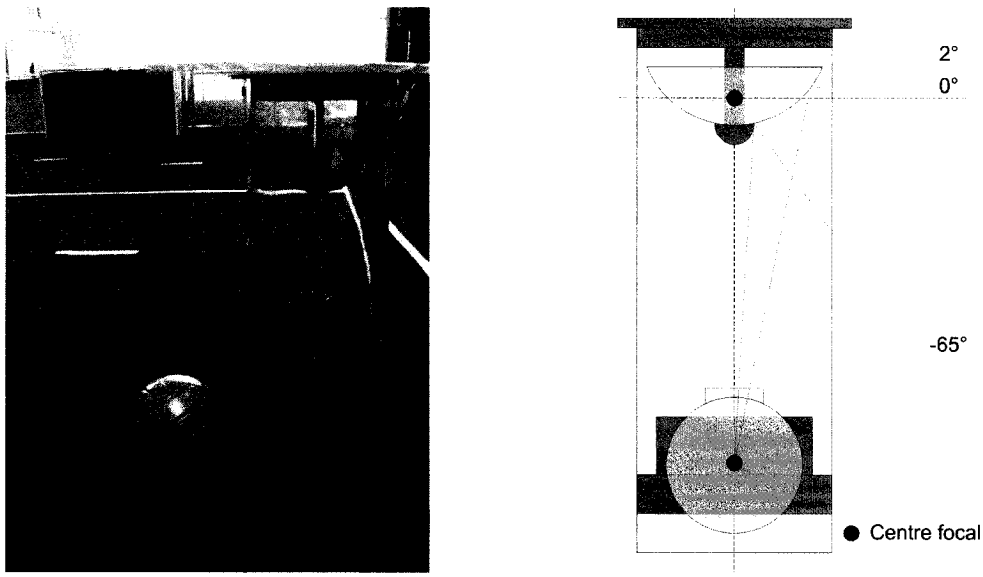

Figure 1.5: Montage de la caméra omnidirectionnelle

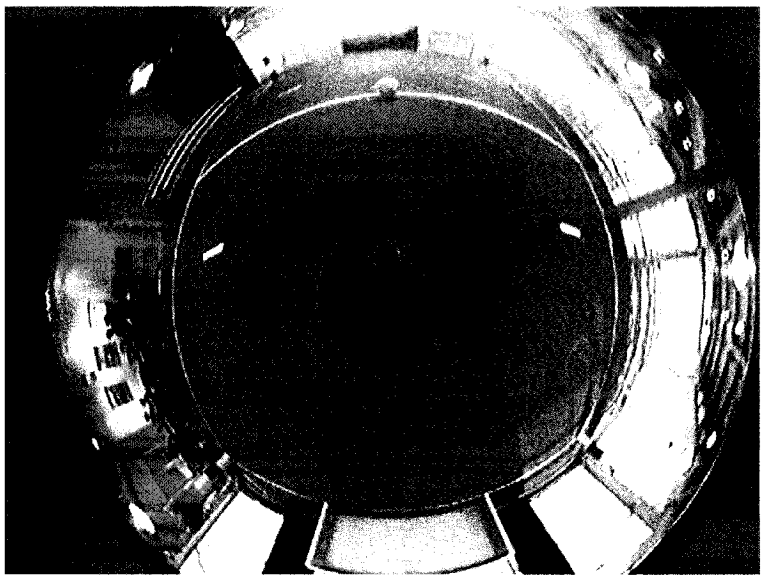

Figure 1.6: Exemple d'image omnidirectionnelle

À partir de l'image omnidirectionnelle, on peut mesurer en coordonnées polaires la position d'un objet par rapport au robot. On retrouve dans le haut de l'image les objets situés devant le robot. Ainsi, l'angle formé par le haut de l'image, le centre de l'image et l'objet correspond à l'angle horizontal entre l'avant du robot et l'objet. Lorsque le montage est aligné avec une grande précision, le centre de l'image correspond au centre du miroir et tous les objets présents dans l'environnement tournent autour de ce point lorsque le robot se déplace. La projection des objets rapprochés se retrouve près du centre de l'image alors que celle des objets éloignés est en périphérie de l'image. Ainsi, la distance en pixel entre un objet et le centre de l'image correspond à l'angle vertical formé par l'horizon, le centre focal du miroir et l'objet (voir figure 1.5). Cette relation est fonction de la courbure du miroir et de la distance entre le miroir et la caméra. Les courbures les plus populaires sont hyperbolique ou parabolique.

Utilisées avec la lentille correspondante, ces courbures font en sorte que tous les rayons qui convergent sur le plan image de la caméra convergent aussi sur le centre focal du miroir. Cela permet d'avoir au focus l'ensemble de l'image, donc à la fois les objets rapprochés et éloignés. De plus, cela permet une correspondance plus simple entre les pixels de l'image et l'environnement du robot. Toutefois, le miroir

utilisé est de type parabolique alors que le type de lentille de la webcam conviendrait mieux à un miroir hyperbolique. Toutefois, cela a peu d'importance étant donné la faible résolution de la caméra. De plus, lorsque le montage est aligné avec une grande précision, la relation entre la distance en pixel par rapport au centre de l'image et l'angle vertical correspondant est indépendante de l'angle horizontal et constante pour un même rayon autour du centre de l'image. Toutefois, l'alignement des montages des différents robots laisse à désirer et les angles mesurés peuvent être faussés de manière différente pour chacun des robots. Les problèmes engendrés par ce mauvais alignement seront discutés plus amplement dans les prochaines sections du mémoire.

Le tableau 1.3 décrit les principales caractéristiques de la caméra utilisée. L'encodage *YUV420P* est un encodage couleur de type *YUV* ayant 4 fois moins d'information couleur que d'intensité. Cela a l'avantage de produire une image relativement belle pour l'oeil humain qui est plus sensible aux contrastes qu'aux couleurs tout en limitant la quantité d'informations nécessaires pour produire l'image. De plus, les canaux Y, U et V sont regroupés séparément pour faciliter la compression de l'image. Ce type d'encodage n'est toutefois pas le mieux adapté au traitement à effectuer. Cela sera discuté plus amplement dans la prochaine section.

Les avantages et inconvénients de ce montage sont présentés dans les tableaux 1.4.

Tableau 1.3: Caractéristiques de la caméra

Caractéristiques	***Descriptions***
Caméra	Webcam Logitech Quickcam Pro 4000
Résolutions et fréquences	640x480, 15 images/sec
	320x240, 30 images/sec
	160x120, 30 images/sec
Type d'encodage	YUV420P
Interface	USB 1.0

Tableau 1.4: Avantages et inconvénients du montage de la caméra omnidirectionnelle

Avantages	***Inconvénients***
Peu coûteux	Résolution de l'image limitée. (particulièrement au niveau de la couleur)
Robuste	Fréquence d'images limitée à 15 images/sec.
Caméra facile à interfacer avec l'ordinateur.	Présence de reflets dans l'image à cause du cylindre d'acrylique.
	Image déformée par l'alignement peu précis de la caméra par rapport au miroir.

1.2.4 Unités de traitement, de communication et de contrôle

Les robots sont tous munis d'un ordinateur embarqué identique à l'exception d'un robot qui possède un processeur plus performant. Chaque ordinateur effectue le traitement de la vision, de la perception, de l'intelligence artificielle et du contrôle de haut niveau. Pour mouvoir le robot ou actionner les botteurs, l'ordinateur communique la vitesse désirée de chacune des roues motrices et les deux bits correspondant à l'état de chacun des botteurs à la carte de contrôle. Chaque robot possède aussi un lien de communication sans-fil pour partager de l'information, communiquer avec un interface de contrôle ou pour être reprogrammé. L'ordinateur embarqué présente, en plus d'être à la fois puissant et compact, plusieurs possibilités de programmation et d'utilisation ce qui facilite grandement le développement logiciel. Toutefois, une application temps-réel avec traitement d'images nécessite beaucoup de calculs. Cela fait en sorte que les algorithmes utilisés se doivent d'être simples et optimisés pour éviter de saturer le processeur et le réseau sans-fil. Le tableau 1.5 résume les caractéristiques de ces composantes.

Tableau 1.5: Caractérisques des unités de traitement, de communication et de contrôle

Composantes	*Descriptions*
Ordinateur embarquée	Celeron 566 MHz ou Pentium III 800 MHz 256 Mo de mémoire Mini-disque de 1 Go Système d'exploitation Linux Debian
Liens de communication	Ethernet sans-fil compatible IEEE 802.11b environ 3 Mbits/s
Carte de contrôle	Carte PC104 avec 2 PID en position ou vitesse 2 entrées pour encodeurs optiques 2 ponts en H de puissance 24 entrées ou sorties numériques

1.3 Objectifs du projet

Ce projet a grandement été inspiré par les robots autonomes développés par le laboratoire de mécatronique et Robofoot dont l'objectif principal est de participer à la compétition internationale de soccer robotisé, la Robocup (voir section 1.1.1). Ainsi, le système de perception doit être en mesure de fonctionner parfaitement dans l'environnement prévu pour cette compétition (voir section 1.1.2) tout en étant capable de fonctionner facilement dans n'importe quel autre environnement partiellement contrôlé.

Les éléments reconnus et utilisés par le système sont définis comme suit:

- Un **objet** est de forme simple (cylindrique, sphérique ou rectangulaire), de taille connue et de couleur uniforme, constrastante et définie.
- Une **balise** est un assemblage d'objets.
- Un **objet dynamique** est un objet qui peut se déplacer et dont la position initiale n'est pas nécessairement connue à l'avance.

Les principaux objectifs du projet sont:

- Repérer les objets dans l'image omnidirectionnelle. Cette reconnaissance doit être rapide et robuste.
- Déterminer la position du robot en combinant de manière optimale les mesures d'odométrie et les mesures de la vision obtenues en repérant plusieurs balises disposées dans l'environnement à des emplacements connus par le robot.
- Localiser des objets dynamiques se déplaçant dans l'environnement (ex: ballon de soccer) et estimer leur déplacement en fonction d'une dynamique donnée.
- Partager de manière optimale l'information concernant la position des robots et des objets dynamiques avec ces autres robots présents dans l'environnement.

Les prochaines sections traitent respectivement de chacun de ses objectifs, des solutions retenues, des problèmes rencontrés et des limitations du système.

Chapitre 2

Système de vision

Pour localiser le robot et trouver la position du ballon, le système de perception a besoin d'obtenir de l'information sur son environnement. Plus particulièrement, il doit identifier et positionner des objets par rapport à lui-même. Le système de vision est le système de mesure le plus important du système de perception. Les deux principales étapes décrites dans cette section sont l'acquisition de l'image et la reconnaissance des objets. Tel que décrit dans la section 1.3, les principaux objectifs du système de vision sont:

- Repérer dans l'image omnidirectionnelle des objets de forme simple (cylindrique, sphérique ou rectangulaire), de taille connue et de couleur uniforme, constrastante et définie.
- Effectuer une reconnaissance rapide et robuste.

Malheureusement, rapidité et robustesse sont des éléments contradictoires, mais essentiels pour le bon fonctionnement du système. En effet, dans une application temps-réel où le robot évolue rapidement dans son environnement, le traitement de la vision doit être exécuté dans un très court laps de temps sans quoi l'information recueillie n'est plus d'aucun intérêt. De plus, la reconnaissance doit être suffisamment robuste pour éviter de confondre les objets recherchés avec d'autres objets présents dans l'environnement. Ainsi, la solution retenue présente un compromis entre la rapidité du traitement et la robustesse.

La reconnaissance d'objets est un problème complexe qui impose des techniques avancées de traitement de l'image. Plusieurs ouvrages en discutent (voir [9], [10], [11], [12], [13], [14] et [15]). Les hypothèses établies concernant les caractéristiques des objets recherchés simplifient la reconnaissance des objets. Lorsque l'environnement est inconnu, d'autres techniques, tel que celles discutées dans [16] doivent être envisagées pour déterminer les objets qui peuvent être utilisés comme balises.

2.1 Acquisition

2.1.1 Modélisation d'une caméra

Le modèle de la caméra est simple d'utilisation et polyvalent. Il peut représenter à la fois une caméra unidirectionnelle et omnidirectionnelle. La figure 2.1 présente le modèle utilisé. Les principales hypothèses sont:

- La caméra est alignée par rapport à un axe vertical perpendiculaire au sol. Cela a pour conséquence que la ligne d'horizon est constante dans l'image indépendamment de la coordonnée en x d'un pixel.
- L'angle α est inversement proportionnel à la coordonnée en x d'un pixel de l'image.
- L'angle β est fonction de la coordonnée en y d'un pixel de l'image.

L'angle α est l'angle horizontal formé par le devant de la caméra, le centre focal et un objet. L'angle α a été défini de cette façon pour être conforme à la convention suivante: si le robot a une orientation de 0° c'est-à-dire parallèle au sens croissant de l'axe des x, alors un objet situé dans le sens croissant de l'axe des y est situé à 90° par rapport au robot et son image est dans la portion gauche de l'image panoramique. L'angle β est l'angle vertical décrit entre l'horizon, le centre focal de la caméra et l'objet. Cet angle est défini nul à l'horizon et il croit vers le haut. Une table est utilisée pour faire la correspondance entre la coordonnée en y d'un pixel et l'angle β. Cela a l'avantage de permettre une plus grande flexibilité principalement par rapport au type de miroir utilisé dans le cas d'une caméra omnidirectionnelle. Toutefois, la création de la table nécessite un bon étalonnage du sytème. La correspondance entre

les angles α et β est simple et rapide. Toutefois, dépendamment de la lentille, des aberrations en bordure de l'image font en sorte que les hypothèses ne sont pas parfaitement respectées et la qualité des mesures effectuées est affectée dans ce cas.

La caméra omnidirectionnelle est modélisée comme une caméra unidirectionnelle ayant un très grand angle de vue. Les routines de traitement doivent considérer dans ce cas particulier que les pixels à l'extrême gauche sont liés à ceux de l'extrême droite. Pour satisfaire ce modèle, l'image du miroir de la caméra omnidirectionnelle devra être transformée (voir section 2.1.2).

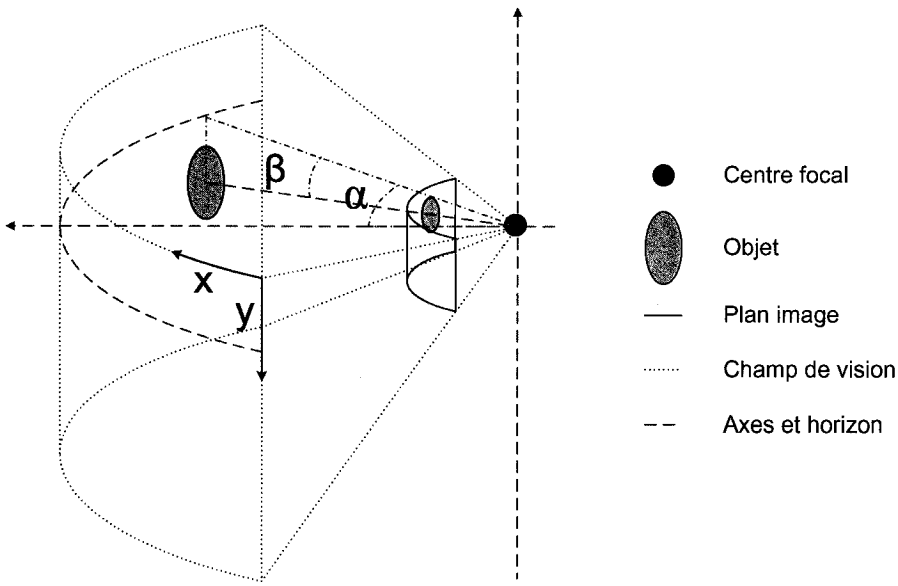

Figure 2.1: Modèle d'une caméra unidirectionnelle

2.1.2 Image panoramique

2.1.2.1 Avantages de l'image panoramique

L'image originale captée par la caméra (voir figure 2.3) est la réflexion de l'environnement du robot sur le miroir convexe. Idéalement, le miroir est perpendiculaire à la caméra et son centre est parfaitement au centre de l'image. Malheureusement, tel

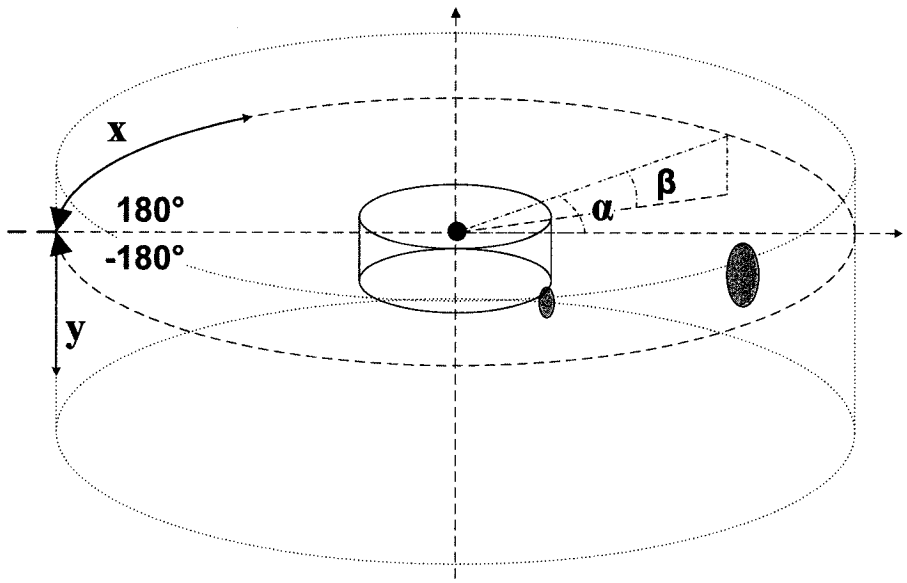

Figure 2.2: Modèle d'une caméra omnidirectionnelle

que discuté à la section 1.2.3, cet alignement est imprécis dans la réalité. Le centre du miroir autour duquel toute la scène tourne n'est pas au centre de l'image et sa position doit être identifiée avec précision pour produire une image panoramique juste et non biaisée. Sur l'image, on peut remarquer que les bordures verticales des objets situés dans l'environnement forment des lignes qui convergent vers le centre du miroir. Une procédure d'étalonnage basée sur cette caractéristique a été élaborée pour trouver le centre (voir figure 2.7).

Les angles α et β définis dans la section 2.1.1 peuvent être déduits par la position de l'objet dans l'image captée lorsque la position est représentée en coordonnés polaire (angle et rayon) à partir du centre du miroir. Cette représentation, en plus de ne pas concorder avec le modèle de la caméra, est moins intuitive. L'utilisation d'une image panoramique permet une interprétation plus simple et conforme au modèle établi. Dans l'image panoramique, les bordures verticales des objets forment des lignes verticales et non des rayons. Ainsi, un cylindre qui prend une forme conique dans l'image du miroir, conserve une forme rectangulaire dans l'image panoramique. Ceci facilite la recherche et la reconnaissance des objets. De plus, l'image panoramique a l'avantage d'éliminer les portions inutilisées de l'image du miroir et d'éviter le traitement de

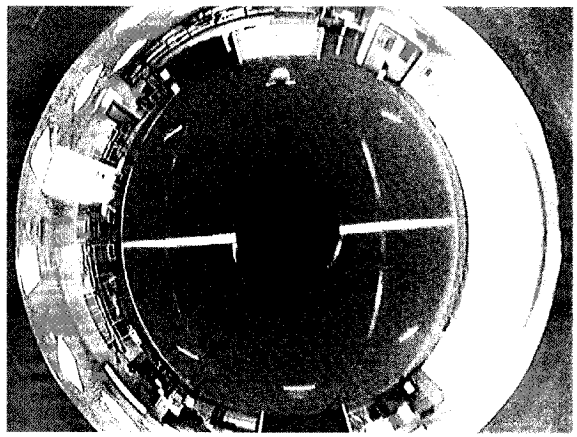

Figure 2.3: Image originale de la réflexion sur le miroir convexe

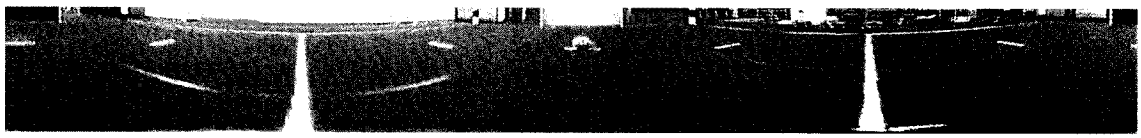

Figure 2.4: Image panoramique de l'environnement

ces zones. Par contre, à cause de l'algorithme de conversion décrit dans la section suivante, un pixel près du centre du miroir est répété plusieurs fois dans le bas de l'image panoramique ce qui entraîne dans certains cas un traitement supplémentaire.

2.1.2.2 Algorithme de transformation

La figure 2.5 présente le principe de l'algorithme de tranformation de l'image du miroir à l'image panoramique. La ligne supérieure de l'image panoramique correspond au cercle plein tracé dans l'image du miroir alors que la ligne inférieure correspond au cercle pointillé. Le beigne décrit par l'aire entre ces deux cercles, la zone utile, contient toute l'information permettant de construire l'image panoramique. Ces deux cercles sont centrés autour du centre du miroir, pixel autour duquel toute la scène tourne lorsque le robot se déplace. L'aire du petite cercle pointillé, la zone morte, est une zone qui ne renferme pas d'information utile puisqu'il s'agit de l'image de la vis supportant le miroir ou bien d'une partie du robot.

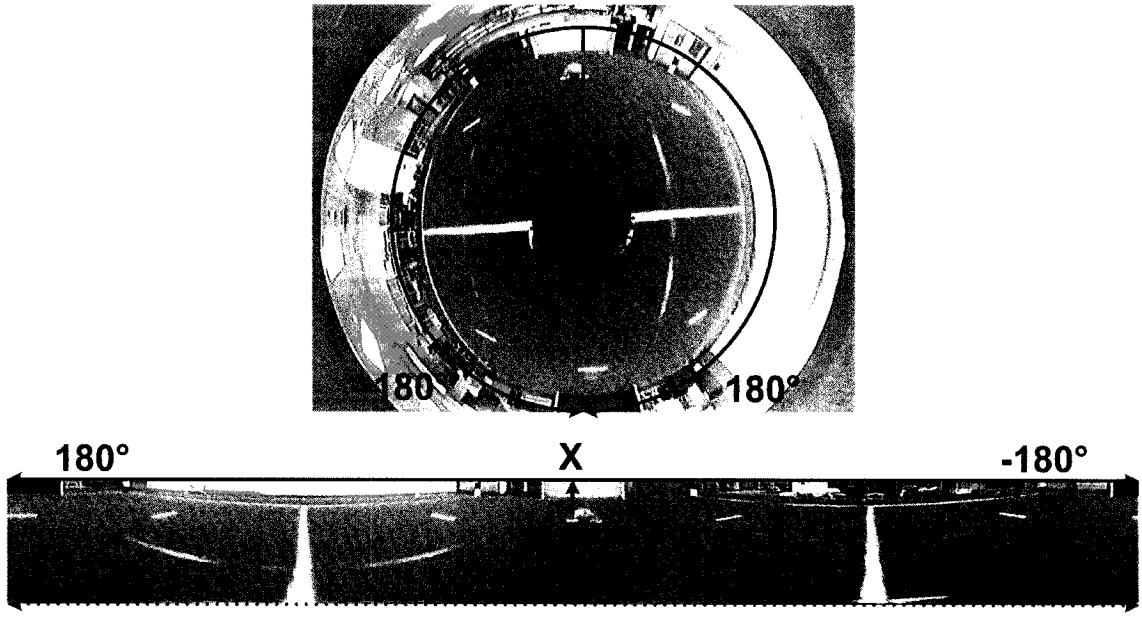

Figure 2.5: Transformation de l'image du miroir en image panoramique

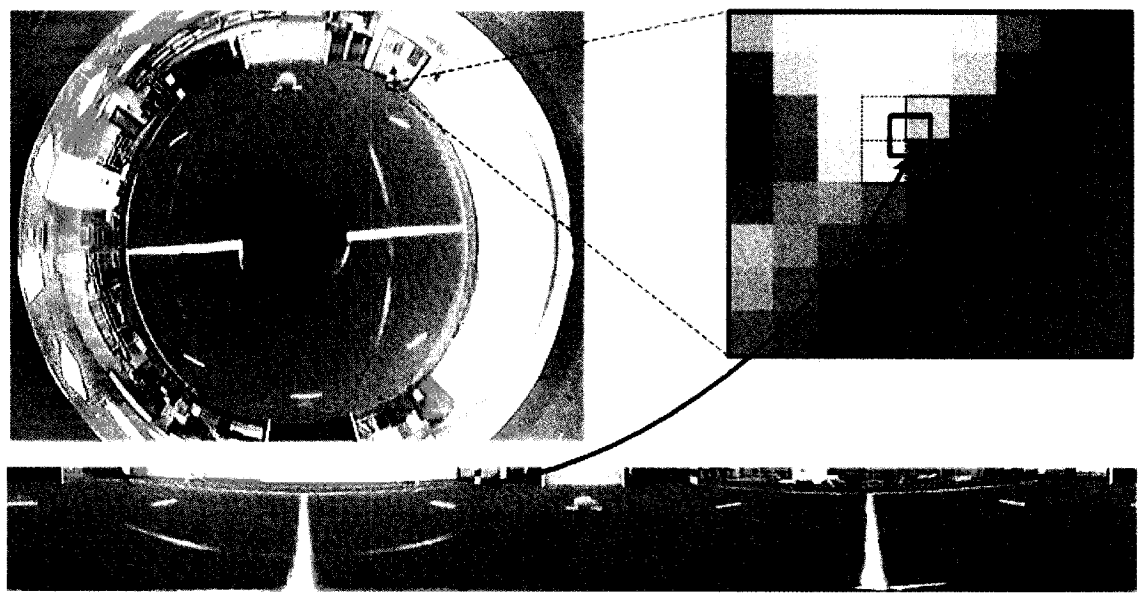

Figure 2.6: Correspondance entre les pixels de l'image panoramique et ceux du miroir

Pour chacun des pixels de l'image panoramique, il faut déterminer l'angle et le rayon correspondant aux mêmes pixels dans l'image du miroir. La position en x d'un pixel de l'image panoramique représente l'angle de rotation tandis que la position en y représente la longueur du rayon à partir du centre du miroir dans l'image du miroir. L'information des pixels d'une ligne horizontale de l'image panoramique suit un cercle qui débute au milieu de la partie inférieure de l'image du miroir pour se poursuivre dans le sens anti-horaire. Tel que montré à la figure 2.6, un pixel de l'image panoramique ne correspond pas exactement à un seul pixel, mais à une pondération de quatre pixels. Toutefois, pour limiter le temps de calcul nécessaire à la transformation, seul le pixel le plus significatif a été utilisé dans ce projet. De plus, pour accélérer le traitement et éviter les opérations trigonométriques, une table a été utilisée pour stocker la correspondance entre les pixels de l'image panoramique et ceux de l'image du miroir. Ainsi, pour chacun des pixels de l'image panoramique, on lit la position inscrite dans la table de correspondance et on récupère l'information dans l'image du miroir à la position indiquée.

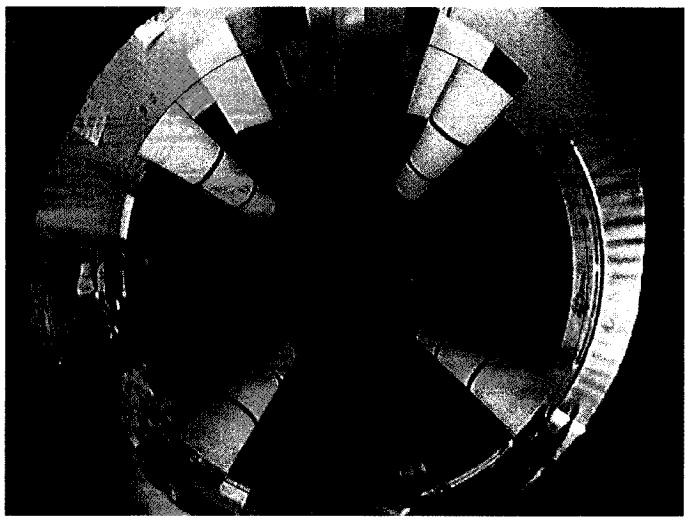

Figure 2.7: Étalonnage du centre du miroir

Cet algorithme est simple et fonctionnel, mais il est conçu pour fonctionner sur un montage dont l'alignement entre le miroir et la caméra est parfait. Pour éliminer les déformations engendrées par un mauvais alignement, un autre algorithme doit être utilisé pour définir la table de correspondance. Une fois la table redéfinie,

tous les autres algorithmes de vision peuvent s'appliquer normalement sur l'image panoramique qui n'a plus de déformations indésirables. Toutefois, cet algorithme de correction n'est pas discuté dans ce mémoire bien qu'il ait été étudié dans le cadre d'un projet de fin d'étude.

2.1.3 Représentation des couleurs

2.1.3.1 Représentation RGB

La représentation *RGB* est probablement la représentation la plus connue. Basée sur le principe d'addition des trois couleurs primaires (voir figure 2.8), cette représentation est couramment utilisée dans les téléviseurs et les écrans d'ordinateurs. Ceux-ci sont constitués d'une série de pixels rouge, vert et bleu très rapprochés dont la luminosité varie pour produire les couleurs désirées. Numériquement, la couleur d'un pixel est généralement représentée sur 24 bits, soit respectivement 8 bits pour le rouge, le vert et le bleu. Bien que cette représentation découple les trois couleurs primaires, elle est moins intuitive et elle rend l'information de luminance couplée avec la chrominance.

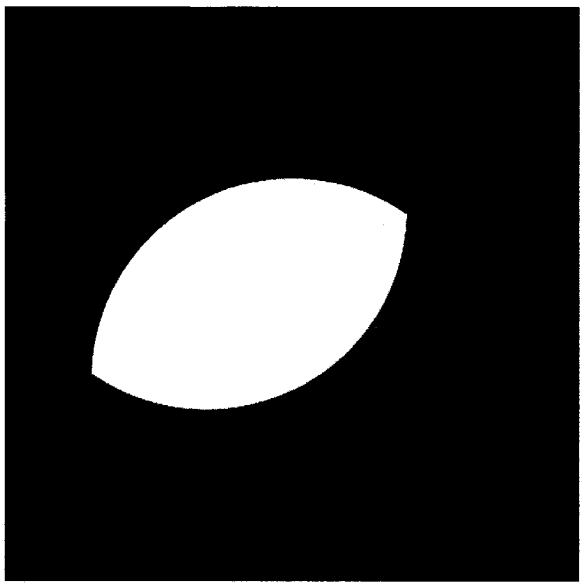

Figure 2.8: Addition des couleurs

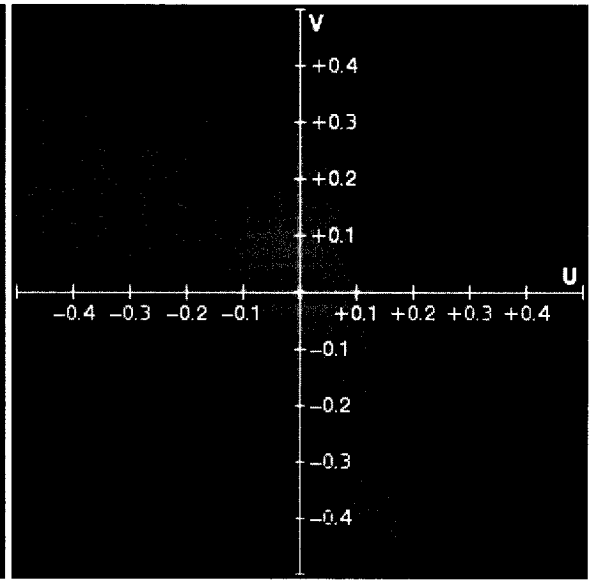

Figure 2.9: Les canaux U et V

2.1.3.2 Représentation YUV

La représentation YUV est couramment utilisée pour l'acquisition et la transmission de signaux vidéo. Le canal Y correspond à la luminosité en noir et blanc d'un pixel. Les canaux U et V, quant à eux, correspondent à la chrominance (la couleur). Le canal U est proportionnel à la différence entre l'intensité de la couleur bleu et la luminosité du pixel. De la même façon, le canal V est proportionnel à la différence entre l'intensité de la couleur rouge et la luminosité du pixel (voir figure 2.9). Il est à noter que les canaux U et V sont signés.

La représentation YUV a l'avantage de découpler l'information relative à la luminosité et la chrominance des pixels d'une image. Ceci a premis, notamment, la compatibilité entre les téléviseurs couleurs et ceux noir et blanc, ces derniers ignorant tout simplement les canaux U et V. De plus, étant donné que l'oeil humain est plus sensible aux constrastes qu'à la couleur, le découplage de la luminance et de la chrominance favorise grandement la compression. Ainsi, les signaux YUV contiennent généralement beaucoup plus d'information relative à l'intensité des pixels d'une image qu'à ses couleurs. C'est le cas notamment de la caméra utilisée par la plateforme expérimentale qui utilise un encodage $YUV420P$ (voir section 1.2.3).

2.1.3.3 Représentation HSI

La représentation HSI (*Hue, Saturation and Intensity ou en francais TSL*) est plus intuitive que les représentations RGB et YUV, car elle permet de définir une couleur d'une manière plus naturelle en termes de teinte, de saturation et de luminance. Par exemple, la couleur rose est représentée par une teinte rouge de saturation modérée avec une luminance élevée (voir figure 2.10). Généralement, la composante H est définie de manière circulaire, c'est-à-dire qu'une valeur faible représente le rouge et qu'en augmentant cette valeur, la teinte passe au jaune, au vert, au bleu, au violet et revient finalement au rouge. Ainsi, la composante H est généralement représentée en degrés. En informatique, une couleur est représentée sur 24 bits, soit 8 bits pour chacune des composantes. La teinte varie alors de la valeur 0 à 252 (voir table 2.1).

Il existe plusieurs variantes de la représentation HSI tel que le HSV et le HSL. Le principe général demeure le même, mais les définitions des composantes S et I sont légèrement différentes. La composante de saturation S utilisée dans ce travail est définie comme étant la différence entre la couleur primaire la plus utilisée $(max(R, G, B))$ et la couleur primaire la moins utilisée $(min(R, G, B))$. De plus, la composante de luminance I est définie comme étant l'intensité moyenne des trois couleurs primaires. Cette composante représente l'image noir et blanc tout comme la composante Y de la représentation YUV à quelques nuances près.

En plus de découpler l'information relative à la luminance et la chrominance des pixels, la représentation HSI a l'avantage de faciliter l'identification des couleurs. Par exemple, pour identifier les pixels d'une couleur vive, il suffit de déterminer l'intervalle de la composante H correspondant à la teinte désirée et de fixer la limite inférieure de la composante S à la saturation minimale de la couleur désirée (à quel point la couleur doit être prononcée). De la même façon, pour identifier les tons de gris, il suffit de déterminer la limite supérieure de la composante S correspondant à la saturation maximale désirée (tolérance à avoir une légère teinte) et l'intervalle de la composante I, la luminance désirée. La composante I permet aussi d'effectuer un traitement sur l'image sans tenir compte de la couleur comme s'il s'agissait d'une image en noir et blanc. Étant donné ces avantages, la représentation HSI a été utilisée.

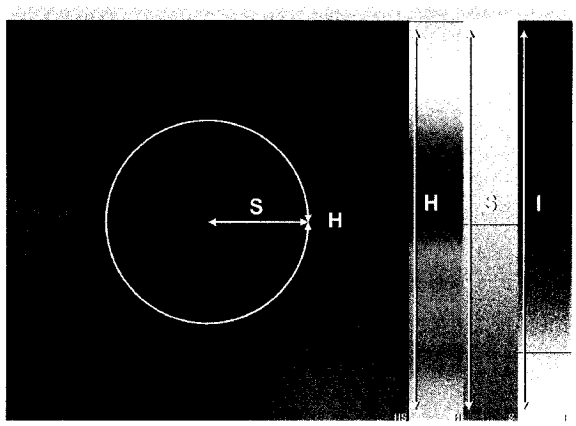

Figure 2.10: Représentation HSI

Tableau 2.1: Valeurs de la composante H pour les principales teintes

H (unités)	***H (degrés)***	***Teintes***
0	0°	rouge
42	60°	jaune
84	120°	vert
126	180°	cyan
168	240°	bleu
210	300°	magenta
252	360°	rouge

2.1.3.4 Conversion YUV, HSI, RGB

La conversion entre les représentations YUV et RGB est exprimée par les équations algébriques du tableau 2.2. Cette conversion est parfaitement réversible dans la mesure où les valeurs peuvent être représentées sur 8 bits (de 0 à 255). Des composantes YUV peuvent produire des composantes RGB inférieures à 0 ou supérieures à 255 ce qui oblige l'algorithme de conversion à saturer les valeurs, rendant ainsi la conversion irréversible. De plus, pour accélérer la conversion, il est préférable d'utiliser des valeurs entières, tel que décrit dans le tableau 2.3. La division entière de 65536 peut être effectuée par un décalage de 16 bits vers la droite, ce qui accélère davantage l'opération.

La conversion entre les représentations RGB et HSI est plus complexe et elle ne peut être représentée par des équations algébriques. Le tableau 2.4 présente la méthode utilisée pour convertir la représentation RGB en HSI avec H de 0 à 252 et non de 0 à 360°. La composante I est la valeur moyenne des composantes R, G et B alors que la saturation est l'écart entre la valeur minimale et maximale. La composante H, la teinte, peut être retrouvée en observant la proportion du mélange entre la composante dominante et la seconde. De plus, la composante H n'est pas définie si la saturation est égale à 0 ou si elle est inférieure à une tolérance fixée. Dans ce cas, la valeur 255 est assignée à la composante H pour montrer que la teinte n'est pas déterminée.

Tableau 2.2: Conversion *YUV* et *RGB*

Y	=	0.299R + 0.587G + 0.114B
U	=	0.492 (B-Y)
	=	-0.147R - 0.289G + 0.436B
V	=	0.877 (R-Y)
	=	0.615R - 0.515G - 0.100B
R	=	Y + 1.140V
G	=	Y - 0.395U - 0.581V
B	=	Y + 2.032U

Tableau 2.3: Conversion *YUV* et *RGB* optimisée

Y	=	(19595R + 38470G + 7471B) / 65536
U	=	32244(B-Y) / 65536
	=	(-9634R - 18940G + 28574B) / 65536
V	=	57475(R-Y) / 65536
	=	40305R - 33751G - 6554B) / 65536
R	=	(65536Y + 74711V) / 65536
G	=	(65536Y - 25887U - 38076V) / 65536
B	=	(65536Y + 133169U) / 65536

La conversion *HSI* à *RGB* est présentée dans le tableau 2.5. Cette conversion est réversible dans la mesure où les composantes S et I ne présentent pas d'aberrations. Par exemple, il est impossible d'avoir à la fois une saturation et une intensité de la valeur maximale 255. Pour avoir une saturation de 255, il faut minimalement avoir une des composantes R, G et B nulle, ce qui entraîne une intensité maximale de 170. De plus, la seule façon d'obtenir une intensité de 255 est d'avoir toutes les composantes R, G et B à 255, ce qui correspond à une saturation nulle. Toutefois, lorsqu'une image est produite par une caméra, elle ne contient pas ces aberrations.

Finalement, il est important de restreindre les valeurs des composantes R, G et B à l'intervalle 0 à 255 après la conversion si elles sont représentées sur 8 bits.

La conversion entre les représentations YUV et HSI n'est pas triviale et la méthode la plus simple d'y parvenir est de passer par la représentation RGB. Par contre, cette double conversion nécessite beaucoup plus de temps de calcul ce qui convient peu à une application temps-réel. Toutefois, il existe une correspondance directe entre les composantes de chrominance UV et HS. Ainsi, l'utilisation d'une table de correspondance entre ces deux composantes est préférable et très efficace. Pour construire cette table, il suffit de parcourir toutes les possiblités des composantes UV (65536 combinaisons pour des composantes de 8 bits) et de les convertir en HS en passant par la représentation RGB. Il est préférable d'utiliser une représentation RGB de plus de 8 bits pour cette phase de la convertion afin d'éviter de saturer les valeurs dans certains cas. La composante I peut être déduite directement des composantes Y, U et V (voir tableau 2.6). Pour accélérer la conversion, il est possible d'utiliser des valeurs entières et même d'ajouter le résultat $12209V + 35756U$ à la table de conversion.

Tableau 2.4: Algorithme de conversion RGB à HSI

max	=	max(R,G,B)
min	=	min(R,G,B)
I	=	(R+G+B)/3
S	=	max - min
H	=	42-(R-G)*42/S (si R = max et B = min)
	=	210+(R-B)*42/S (si R = max et G = min)
	=	42+(G-R)*42/S (si G = max et B = min)
	=	126-(G-B)*42/S (si G = max et R = min)
	=	126+(B-G)*42/S (si B = max et R = min)
	=	210-(B-R)*42/S (si B = max et G = min)
	=	255 (si S = 0)

Tableau 2.5: Algorithme de conversion HSI à RGB

	reste	=	H modulo 42
si $0 < H < 42$	min	=	I - (S+reste*S/42)/3
	R	=	S + min
	G	=	reste*S/42 + min
	B	=	min
si $42 \leq H < 84$	min	=	I - (S+(42-reste)*S/42)/3
	R	=	(42-reste)*S/42 + min
	G	=	S + min
	B	=	min
si $84 \leq H < 126$	min	=	I - (S+reste*S/42)/3
	R	=	min
	G	=	S + min
	B	=	reste*S/42 + min
si $126 \leq H < 168$	min	=	I - (S+(42-reste)*S/42)/3
	R	=	min
	G	=	(42-reste)*S/42 + min
	B	=	S + min
si $168 \leq H < 210$	min	=	I - (S+reste*S/42)/3
	R	=	reste*S/42 + min
	G	=	min
	B	=	S + min
si $210 \leq H < 252$	min	=	I - (S+(42-reste)*S/42)/3
	R	=	S + min
	G	=	min
	B	=	(42-reste)*S/42 + min
si $H = 252$	min	=	I - (S+reste*S/42)/3
	R	=	S + min
	G	=	min
	B	=	min

Tableau 2.6: Correspondance entre la composante I et le YUV

$$I = Y + 0.1863V + 0.5456U$$

ou

$$I = (65536Y + 12209V + 35756U)/65536$$

2.2 Reconnaissance des objets

Une fois l'acquisition, la transformation et la conversion terminées, l'image panoramique est utilisée pour identifier la position relative de différents objets par rapport au robot. Tel que mentionné plus tôt, un objet a une forme cylindrique, sphérique ou rectangulaire, sa taille est connue ainsi que sa couleur. La couleur d'un objet doit être uniforme et contrastante pour qu'il soit repéré correctement. On peut vouloir mesurer la position relative d'un objet pour deux raisons différentes soit, mesurer la position du robot ou mesurer la position d'un objet.

Dans le premier cas, la position du robot est connue approximativement et la position des objets est connue précisément. De plus, pour rendre la reconnaissance plus robuste, on recherche dans ce cas des balises dans l'image. Une balise est simplement un assemblage d'objet. Ainsi, pour valider une balise, on vérifie la présence des objets qui la composent.

Dans le second cas, la position du robot est connue précisément alors que la position de l'objet recherché, l'objet dynamique, est connue très approximativement. De plus, un objet dynamique est par hypothèse constitué d'un seul objet.

Il est donc possible, dans les deux cas, de prédire la position des objets dans l'image. L'algorithme de reconnaissance est basé sur cette hypothèse et son fonctionnement se résume alors à trois grandes étapes. La première, la prédiction, a pour objectif de prédire la position de l'objet et sa taille dans l'image afin de définir la zone dans laquelle il doit être recherché. La seconde, la segmentation, analyse la zone prédite en fonction de la couleur recherchée. La dernière étape, la validation, vérifie les résultats obtenus par la segmentation en fonction de la prédiction et des tolérances

établies. Par la suite, des algorithmes de plus haut niveau traitent les objets perçus afin d'estimer la position du robot ou celle de l'objet.

2.2.1 Prédiction

La prédiction est la première étape de la reconnaissance d'un objet. Elle détermine la zone de recherche utilisée lors de la segmentation ainsi que les paramètres permettant le filtrage et la validation des zones trouvées. Les paramètres recherchés sont présentés dans le tableau 2.7.

Paramètres	***Descriptions***
α_{min}	Limite inférieure correspondant à l'extrémité gauche de la zone de recherche une fois convertie en pixel dans l'image.
α_{max}	Limite supérieure correspondant à l'extrémité droite de la zone de recherche une fois convertie en pixel dans l'image.
$\Delta\alpha_{min}$	Largeur minimale en pixel de l'objet.
$\Delta\alpha_{max}$	Largeur maximale en pixel de l'objet.
β_{min}	Limite inférieure correspondant à l'extrémité inférieure de la zone de recherche une fois convertie en pixel dans l'image.
β_{max}	Limite supérieure correspondant à l'extrémité supérieure de la zone de recherche une fois convertie en pixel dans l'image.
$\Delta\beta_{min}$	Hauteur minimale en pixel de l'objet.
$\Delta\beta_{max}$	Hauteur maximale en pixel de l'objet.

Tableau 2.7: Paramètres recherchés lors de la prédiction

Tous ces paramètres sont exprimés sous forme d'angles avant d'être convertis en pixel pour permettre la segmentation. Les figures 2.13 et 2.14 présentent shématique-

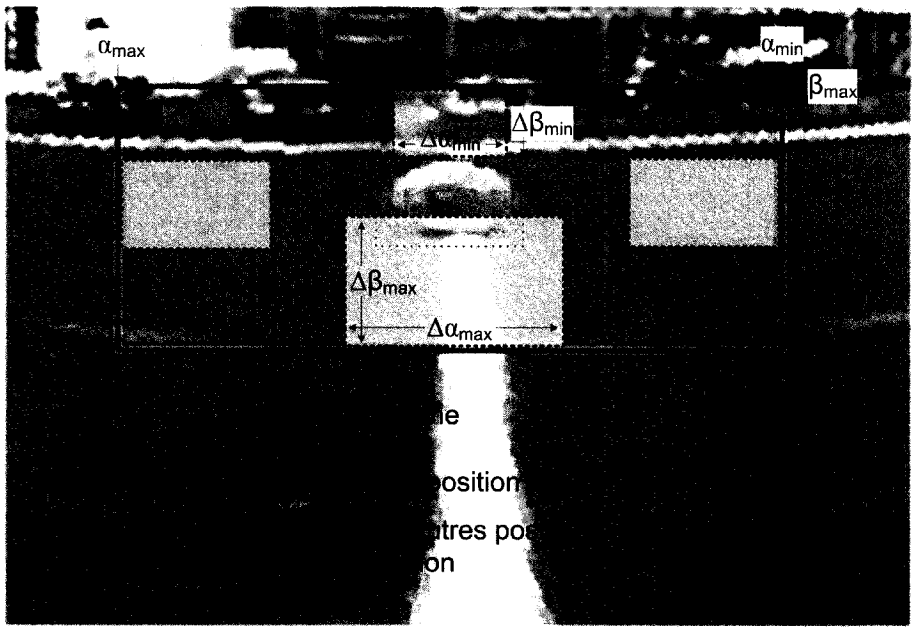

Figure 2.11: Prédiction de la zone de recherche et de la taille dans l'image

ment ces paramètres et la figure 2.11 présente le résultat escompté une fois les paramètres convertis sous forme de pixel. Ces figures illustrent des exemples typiques, mais les paramètres peuvent varier lorsque d'autres situations sont illustrées.

Les algorithmes utilisés ont pour objectif de fournir une approximation de ces paramètres et non leurs valeurs exactes étant donné que ceux-ci dépendent de variables dont nous ignorons les valeurs réelles telles que la position du robot ou de l'objet. De plus, une approximation est largement suffisante pour permettre une recherche rapide qui restreint l'algorithme de segmentation à une zone précise et qui élimine facilement les solutions aberrantes.

Les variables utilisées par les algorithmes de prédictions sont décrites dans le tableau 2.8. Les tolérances sur la hauteur et la largeur d'un objet sont utilisées conjointement à l'erreur de position pour définir les variations tolérées sur la taille d'un objet. Cela a pour effet s'assouplir les critères de validation discutés dans la section 2.2.3.

Variables	***Descriptions***
x_{camera}	Position en x de la caméra.
y_{camera}	Position en y de la caméra.
z_{camera}	Hauteur de la caméra.
θ_{camera}	Orientation de la caméra.
x_{objet}	Position en x de l'objet.
y_{objet}	Position en y de l'objet.
z_{objet}	Hauteur de la base de l'objet.
θ_{objet}	Orientation de la normale du rectangle dans le cas d'un objet rectangulaire seulement.
$Hauteur$	Hauteur de l'objet.
$Largeur$	Largeur de l'objet.
$\Delta Hauteur$	Tolérance sur la hauteur de l'objet.
$\Delta Largeur$	Tolérance sur la largeur de l'objet.
$ErreurPosition$	Estimation de l'erreur combinée sur la position du robot et de l'objet.
$ErreurOrientation$	L'erreur liée à l'orientation du robot. Celle-ci a pour effet d'agrandir la largeur la zone de recherche.
$ErreurEtalonnage$	L'erreur liée à l'étalonnage vertical de la caméra. Celle-ci a pour effet d'augmenter la hauteur de la zone de recherche et permet de chercher correctement des objets même si l'étalonnage vertical de la caméra est de piètre qualité.

Tableau 2.8: Variables utilisées pour effectuer la prédiction

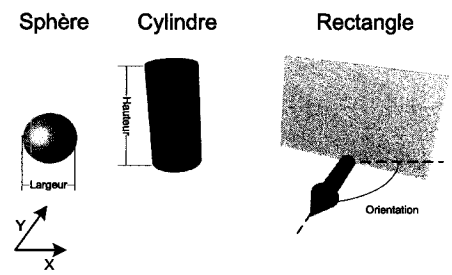

Figure 2.12: Formes et caractéristiques des objets

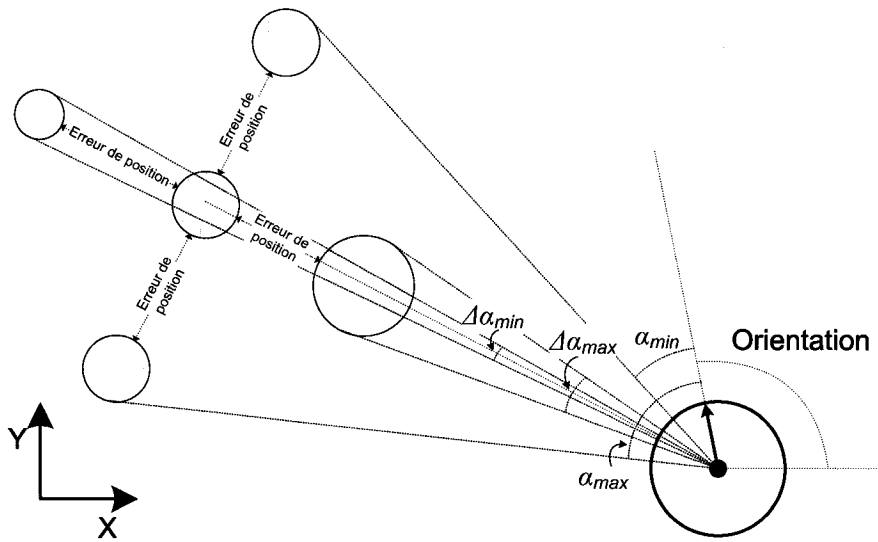

Figure 2.13: Prédiction de la zone de recherche et de la taille à l'horizontal

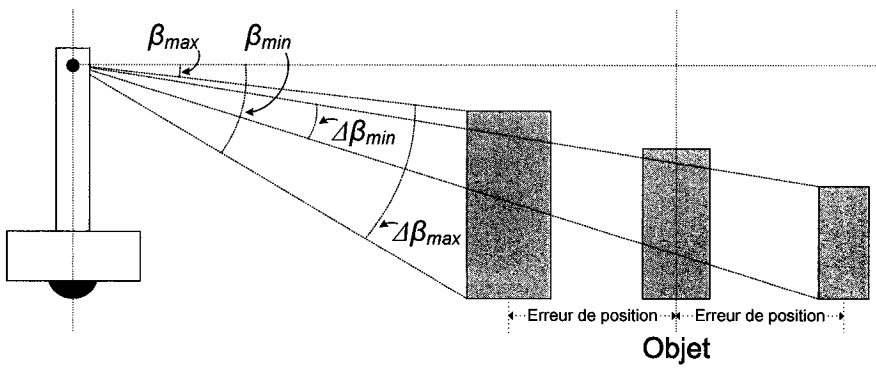

Figure 2.14: Prédiction de la zone de recherche et de la taille à la verticale

Tel qu'illustré sur les figures 2.13 et 2.14, la prédiction s'effectue en deux étapes. La première consiste à estimer les paramètes α, α_{min}, α_{max}, β, β_{min} et β_{max} pour une position donnée et une taille donnée. Les tailles minimale, normale et maximale sont définies par rapport aux dimensions de l'objet plus ou moins les tolérances sur la taille.

Tableau 2.9: Prédiction des paramètres de base

$$\begin{aligned} \alpha &= \arctan\left(y_{objet} - y_{camera}, x_{objet} - x_{camera}\right) - \theta_{camera} \\ distance &= \sqrt{(x_{objet} - x_{camera})^2 + (y_{objet} - y_{camera})^2} \\ \beta &= \arctan\left(z_{objet} - z_{camera}, distance\right) \end{aligned}$$

Tableau 2.10: Prédiction des paramètres d'un cylindre

$$\begin{aligned} \beta_{min} &= \arctan\left(z_{objet} - z_{camera}, distance - \tfrac{Largeur}{2}\right) \\ \beta_{max} &= \arctan\left(z_{objet} + Hauteur - z_{camera}, distance - \tfrac{Largeur}{2}\right) \\ \Delta\beta &= \beta_{max} - \beta_{min} \\ \Delta\alpha &= 2\arctan\left(\tfrac{Largeur}{2}, distance\right) \\ \alpha_{min} &= \alpha - \tfrac{\Delta\alpha}{2} \\ \alpha_{max} &= \alpha + \tfrac{\Delta\alpha}{2} \end{aligned}$$

Tableau 2.11: Prédiction des paramètres d'une sphère

$$\begin{aligned} distance3D &= \sqrt{(x_{objet} - x_{camera})^2 + (y_{objet} - y_{camera})^2 + (z_{objet} - z_{camera})^2} \\ \Delta\beta &= 2\arctan\left(\tfrac{Hauteur}{2}, distance3D\right) \\ \beta_{CM} &= \arctan\left(z_{objet} + Hauteur - z_{camera}, distance\right) \\ \beta_{min} &= \beta_{CM} - \tfrac{\Delta\beta}{2} \\ \beta_{max} &= \beta_{CM} + \tfrac{\Delta\beta}{2} \\ \Delta\alpha &= 2\arctan\left(\tfrac{Largeur}{2}, distance\right) \\ \alpha_{min} &= \alpha - \tfrac{\Delta\alpha}{2} \\ \alpha_{max} &= \alpha + \tfrac{\Delta\alpha}{2} \end{aligned}$$

Tableau 2.12: Prédiction des paramètres d'un rectangle

$$
\begin{aligned}
\Delta\theta &= \theta_{objet} - \alpha - \theta_{camera} \\
\beta_{min} &= \arctan\left(z_{objet} - z_{camera}, distance - \frac{Largeur|\sin\Delta\theta|}{2}\right) \\
\beta_{max} &= \arctan\left(z_{objet} - z_{camera}, distance + \frac{Largeur|\sin\Delta\theta|}{2}\right) \\
\Delta\beta &= \beta_{max} - \beta_{min} \\
\Delta\alpha &= 2\arctan\left(\frac{Largeur|\cos\Delta\theta|}{2}, distance\right) \\
\alpha_{min} &= \alpha - \frac{\Delta\alpha}{2} \\
\alpha_{max} &= \alpha + \frac{\Delta\alpha}{2}
\end{aligned}
$$

La deuxième étape consiste à répéter plusieurs fois la première étape pour différentes positions et tailles de l'objet afin de déterminer les α_{min}, α_{max}, $\Delta\alpha_{min}$, $\Delta\alpha_{max}$, β_{min}, β_{max}, $\Delta\beta_{min}$ et $\Delta\beta_{max}$ absolus. Ainsi, la première étape est répétée pour la position rapprochée et la taille maximale, la position éloignée et la taille minimale, la position la plus à gauche et la taille normale ainsi que la position la plus à droite et la taille normale de l'objet. De plus, la distance de l'objet par rapport au robot doit être minimalement égale au rayon du robot pour éviter la prédiction d'un objet situé à l'intérieur du robot. Les paramètres α_{min}, α_{max}, β_{min} et β_{max} doivent être limités à l'image lorsque l'objet est partiellement à l'extérieur du champ de vision du robot et les paramètres $\Delta\alpha_{min}$, $\Delta\alpha_{max}$, $\Delta\beta_{min}$ et $\Delta\beta_{max}$ doivent être corrigés en conséquences. Après la conversion en pixel, si les paramètres $\Delta\alpha_{max}$, $\Delta\beta_{max}$ correspondant à la taille maximale de l'objet sont inférieurs à la limite donnée (par exemple: 2 pixels), l'objet est considéré introuvable et les étapes subséquentes ne sont pas exécutées. Cela évite d'effectuer la recherche d'objets dont la taille est similaire à la résolution de l'image et aux perturbations engendrées par le bruit numérique. De la même façon, plus un objet est loin du robot, plus les Δ_{min} et Δ_{max} sont similaires. Il est alors important de limiter la différence à une limite inférieure définie en fonction des perturbations engendrées par le bruit numérique, de l'imprécision du montage et des erreurs d'étalonnage.

2.2.2 Segmentation

La segmentation est une procédure classique en vision qui consiste à regrouper des pixels adjacents selon un critère de sélection et de faire resortir ainsi certains paramètres de la surface trouvée. Tel que mentionné dans la section précédente, la segmentation de l'image s'effectue sur une portion précise de l'image. Cette portion appelée zone de recherche est déterminée lors de la prédiction. L'image panoramique est une image rectangulaire dont l'information est disposée linéairement en mémoire, une ligne à la suite de l'autre. Toutefois, les pixels à l'extrême gauche et à l'extrême droite d'une même ligne doivent être traités comme s'ils étaient adjacents même si ce n'est pas le cas en mémoire puisqu'ils représentent l'image de ce qui est disposé exactement à 180° derrière le robot. Ainsi, une zone de recherche peut commencer dans la portion droite de l'image et se terminer dans la portion gauche ce qui rend l'adressage de l'image plus complexe.

Trois types de segmentation, ou encore trois critères de segmentation ont été utilisés (voir la section 2.1.3.3 portant sur la représentation HSI). Le type HS est utilisé pour effectuer une segmentation couleur. Cette segmentation regroupe tous les pixels dont la composante H est comprise dans l'intervalle H_{min} et H_{max} et dont la composante S est comprise dans l'intervalle S_{min} et S_{max}. Ainsi, on regroupe les pixels d'une teinte précise et d'une saturation précise. Par exemple, pour regrouper des pixels oranges vifs, la composante H doit être comprise dans l'intervalle $[10, 20]$ et la saturation doit être forte ($[80, 255]$). La figure 2.15 présente, dans un premier temps, les canaux H, S et I, et dans un second temps, le canal H seulement. Cela fait ressortir l'information relative aux teintes. La figure 2.16 présente quant à elle le résultat de la sélection après un critère sur la composante H et après un critère sur H et S. On remarque que le critère sur H élimine les teintes non désirées alors que le critère sur S assure une teinte prononcée et élimine donc les ballons plus clairs.

De la même façon, le type SI permet de regrouper les pixels selon une teinte de gris en définissant l'intervalle S_{min} et S_{max} et l'intervalle I_{min} et I_{max}. Par exemple, pour regrouper les pixels de couleur blanche, la saturation doit être faible ($[0, 15]$, absence de teinte) et l'intensité doit être élevée. Finalement, le type I permet une seg-

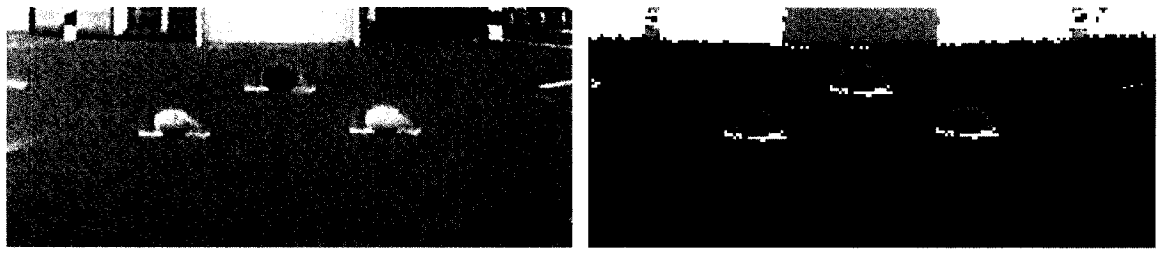

Figure 2.15: Image originale HSI et H seulement

Figure 2.16: Image après segmentation avec critère sur H puis sur H et S

mentation noir et blanc en définissant seulement un intervalle d'intensité I_{min} et I_{max}.

Les paramètres retournés par la segmentation pour chacune des zones trouvées sont présentés dans le tableau 2.13. Ces paramètres sont aussi représentés dans la figure 2.17 où une segmentation des pixels oranges a été effectuée à titre d'exemple. Pour représenter les résultats d'une segmentation, on utilise une croix comprise dans les limites décrites par X_{min}, X_{max}, Y_{min} et Y_{max} dont l'intersection est située au centre de masse de la zone (voir figure 2.17). X_{max} est inférieur à X_{min} lorsque la zone trouvée commence à droite de l'image pour se terminer à gauche.

L'algorithme consiste à parcourir tous les pixels de la zone de recherche à partir du coin supérieur gauche au coin inférieur droit pour vérifier, un par un, s'il correspond aux critères établis et pour l'ajouter à la zone adjacente, s'il y a lieu. L'algorithme préserve, dans une variable temporaire, la zone à laquelle le pixel précédent est associé (aucune si le pixel ne répond pas au critère). Il préserve aussi, dans un vecteur temporaire, la zone associée à chacun des pixels de la ligne au-dessus, soit la ligne traitée précédamment. Ainsi, lorsqu'un pixel est conforme aux critères, on vérifie la zone des pixels adjacents, soit le pixel de gauche et le pixel du dessus.

Paramètres	***Descriptions***
X_{min}	La position, dans l'image, du pixel situé à la limite gauche de la zone trouvée.
X_{max}	La position, dans l'image, du pixel situé à la limite droite de la zone trouvée.
CX	La position, dans l'image, du centre de masse en x de tous les pixels de la zone trouvée.
Y_{min}	La position, dans l'image, du pixel situé à la limite supérieure de la zone trouvée.
Y_{max}	La position, dans l'image, du pixel situé à la limite inférieure de la zone trouvée.
CY	La position, dans l'image, du centre de masse en y de tous les pixels de la zone trouvée.
$NbPixel$	Le nombre de pixels de la zone.

Tableau 2.13: Paramètres retournés par la segmentation

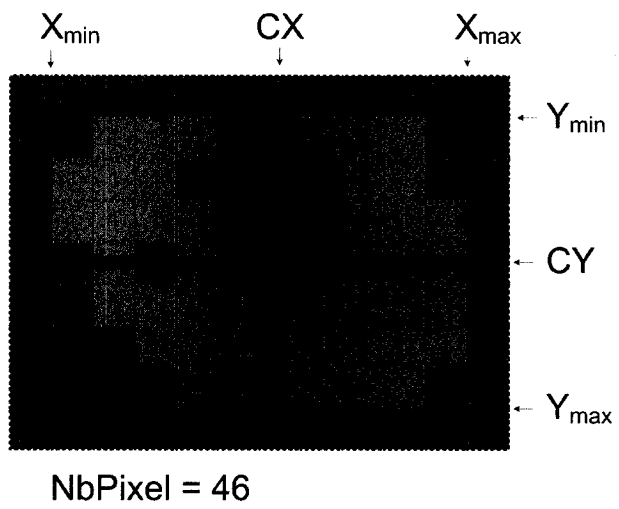

Figure 2.17: Segmentation et paramètres recherchés

Trois scénarios sont alors possibles:

- Si les deux pixels adjacents n'appartiennent à aucune zone, une nouvelle zone est créée.

- Si un des deux pixels voisins appartient à une zone ou si les deux pixels voisins appartiennent à la même zone, le pixel est ajouté à celle-ci. Les limites de la zone sont corrigées si le pixel est à l'extérieur de celle-ci, les positions en x et en y du pixel sont respectivement ajoutées aux variables CX et CY et le nombre de pixels de la zone est incrémenté. Les centres de masse sont déterminés à la fin de la segmentation en divisant les variables CX et CY par le nombre de pixels de la zone.

- Si les deux pixels adjacents appartiennent à deux zones différentes, les zones sont fusionnées puisqu'il s'agit en réalité d'une même zone. Les paramètres de la plus petite zone sont combinés à ceux de la plus grosse et la petite zone est détruite.

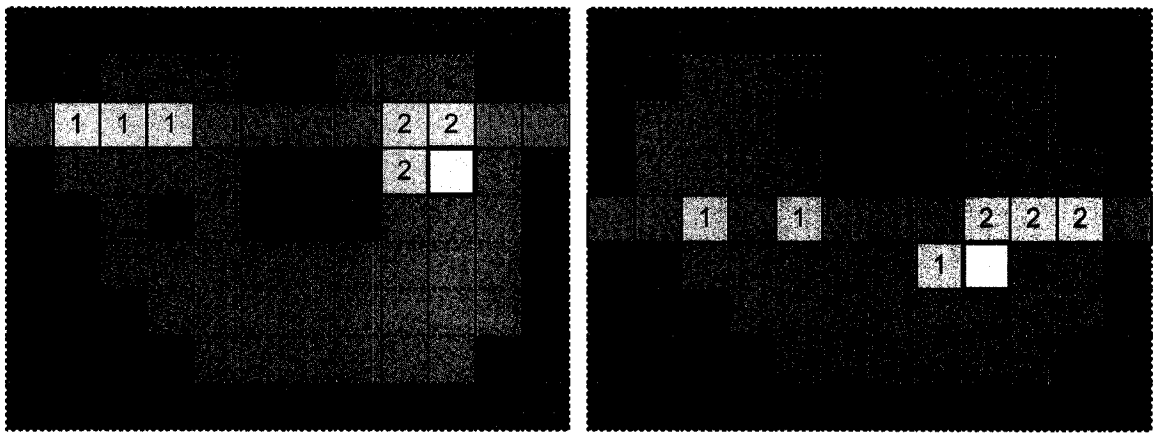

Figure 2.18: Algorithme de segmentation et zones à fusionner

La figure 2.18 présente un exemple de fusion de deux zones. Le pixel en traitement est le pixel clair encadré par un trait gras. Les pixels représentant la variable et le vecteur temporaires sont plus clairs et ils sont encadrés par un trait plein. Dans un premier temps, on remarque que l'algorithme a détecté deux zones. Dans

le second, l'algorithme vient de remarquer que les pixels voisins au pixel en traitement n'appartiennent pas à la même zone, donc les deux zones seront fusionnées. L'algorithme de segmentation fournit donc toutes les zones trouvées qui ne se touchent pas.

2.2.3 Validation

Une validation des zones trouvées lors de la segmentation s'impose pour éliminer les zones aberrantes. Il suffit d'un seul pixel répondant aux critères de sélection pour produire une zone. Ainsi, la segmentation retrouve habituellement, en plus de la solution espérée, plusieurs zones de petites tailles correspondant à du bruit numérique ou à de petites portions d'objets présents dans la zone de recherche dont les pixels sont similaires aux critères de sélection. La validation utilisée a pour but d'éliminer ces zones de manière simple et rapide. Toutefois, elle ne permet pas de différencier des zones similaires pour identifier celle qui correspond à l'objet recherché. Ce traitement doit être effectué à un niveau supérieur après la validation, lorsque plusieurs solutions sont encore possibles.

Les critères de validation sont la largeur, la hauteur et le nombre de pixel de la zone trouvée. La prédiction détermine la taille minimale et maximale ($\Delta\alpha_{min}$, $\Delta\beta_{min}$, $\Delta\alpha_{max}$ et $\Delta\beta_{max}$) que la solution doit avoir. De plus, en fonction de la forme de l'objet et de sa taille, on est en mesure d'estimer en nombre de pixels la surface que la solution doit avoir. Ainsi, la validation implique simplement de parcourir toutes les zones trouvées et de vérifier si leur largeur, leur hauteur et leur surface sont comprises dans l'intervalle prédit.

Tel que discuté dans la section 2.2.1, $\Delta\alpha_{min}$, $\Delta\beta_{min}$, $\Delta\alpha_{max}$ et $\Delta\beta_{max}$ sont déterminés, pour une position et une taille données de l'objet, en fonction de l'erreur de position et des tolérances sur la taille ($\Delta Hauteur$ et $\Delta Largeur$). Ainsi, plus l'erreur de position et les tolérances sont faibles, plus les critères de validation sont contraignants. De plus, une faible erreur de position produit une zone de recherche étroite ce qui a pour effet de réduire les risques de confondre l'objet recherché avec d'autres

objets similaires présents dans l'environnement. Ainsi, étant donné qu'un objet est cherché à un endroit précis dans l'image, la présence d'un autre objet identique a aucune influence tant qu'il n'est pas dans la zone de recherche. Dans le cas contraire, des critères de sélection de plus haut niveau doivent être utilisés pour différencier les objets. Ces critères peuvent être, par exemple, l'objet le plus près de la position prédite ou l'objet le plus près du robot.

La validation offre des résultats concluants lorsque ces hypothèses de base sont respectées:

- L'erreur de position est supérieure à l'erreur réelle entre la position prédite et la position réelle de l'objet.
- La taille et la forme de l'objet sont exactes.
- Le montage de la caméra est bien aligné et la caméra est correctement étalonnée (la relation entre la hauteur d'un pixel et l'angle β est exacte).

Dans le cas contraire, la validation sera trop contraignante et l'objet ne sera pas perçu correctement. Il faudra alors augmenter l'erreur de position, les tolérances ou les variables *ErreurOrientation* et *ErreurEtalonnage* si la problématique ne peut être corrigée. Ces problèmes sont discutés plus en détail dans la section 2.2.4.

2.2.4 Problèmes rencontrés et limitations

2.2.4.1 Mauvaise estimation de la position d'un objet

La position d'un objet dans l'image dépend de la position absolue du robot et de celle de l'objet. Toutefois, ces positions ne sont pas connues avec exactitude et une erreur doit être estimée pour chacune d'elle. L'erreur de position utilisée lors de la prédiction est en quelque sorte la somme des deux erreurs et elle doit être en tout temps supérieure à l'erreur réelle. Dans le cas contraire, l'objet peut être tronqué en partie ou en totalité par la zone de recherche. De plus, la taille prédite est alors incorrecte et la zone de l'objet trouvée dans l'image peut être exclue si elle est trop

petite ou trop grande. De plus, un mauvais étalonnage du montage peut dégrader considérablement la qualité de la prédiction. Par exemple, un mauvais étalonnage de la relation entre la hauteur d'un pixel et l'angle β fausse l'estimation de la distance, ce qui fausse par conséquent la prédiction de la position de l'objet dans l'image et sa taille. Augmenter l'erreur de position et les tolérances sur la taille augmente la zone de recherche et l'intervalle des tailles valides pour l'objet. De plus, les variables *ErreurOrientation* et *ErreurEtalonnage* permettent d'agrandir respectivement la largeur et la hauteur de la zone de recherche. Cela peut permettre de contourner le problème, mais l'environnement doit être mieux contrôlé pour pallier à la diminution de la robustesse qui sera alors engendrée.

2.2.4.2 Reconnaissance d'objets partiellement obstrués

La reconnaissance d'un objet est problématique lorsque celui-ci est partiellement caché derrière un autre objet présent dans l'environnement. La zone trouvée dans l'image sera invalidée si sa taille est inférieure à la taille minimale prédite. Si la zone est valide, alors les mesures effectuées ne seront pas exactes, puisque l'objet ne sera pas perçu en entier. Ainsi, un objet partiellement obstrué sur la gauche donnera l'impression qu'il est situé plus à droite que ce qu'il est en réalité. Les mesures seront alors biaisées.

L'environnement du robot doit être éclairé par une lumière ambiante constante (dans le temps et dans l'environnement) afin que la reconnaissance des objets s'effectue correctement. Puisque la caméra utilisée est omnidirectionnelle, il suffit d'une seule image pour observer tout l'environnement. Un éclairage inégale fait en sorte que certaines portions de l'image panoramique sont surexposées ou sousexposées. De plus, la saturation d'une couleur varie généralement en fonction de l'intensité lumineuse. Ainsi, les pixels des zones ombragées d'un objet ont une saturation plus faible puisque leur couleur est sombre et qu'il y a peu d'information permettant de distinguer la teinte. De la même façon, les reflets de lumière sur un objet diminuent la saturation des pixels affectés en rendant leur couleur très près du blanc. De plus, certaines caméras vont varier aussi la teinte d'un objet en fonction de l'intensité lumineuse. Les portions surexposées et sousexposées de l'image amplifient ces phénomènes et rendent les intervalles de teinte et de saturation d'un objet impossible à identifier précisément.

Les intervalles de segmentation doivent donc être plus larges ce qui augmente le nombre de solutions trouvées. Certaines portions d'objets présents dans l'environnement peuvent alors avoir des pixels d'une couleur comprise dans les intervalles utilisés par la segmentation et produire ainsi des zones qui peuvent être confondues avec l'objet recherché. D'un autre côté, des intervalles trop restreints font en sorte que certaines portions ombragées ou présentant un reflet de lumière ne sont pas retenues lors de la segmentation. Ceci engendre les mêmes conséquences que si l'objet avait été partiellement obstrué par un autre objet puisqu'il n'est pas perçu en entier.

2.2.4.3 Qualité de l'image

L'erreur de mesure est liée à la résolution de l'image. Une résolution élevée fait en sorte que les objets sont représentés par un plus grand nombre de pixel. Une mesure basée sur le centre de masse d'une zone de pixel est d'autant précise que le nombre de pixels formant la zone est élevé. De plus, pour une même surface dans l'image, une résolution élevée rend possible la reconnaissance d'objets plus petits et plus éloignés. La caméra utilisée dans ce projet produit une image dont la résolution de l'information de luminance (image noir et blanc) est quatre fois plus grande que la résolution de l'information de chrominance. Cela dégrade alors la qualité de la segmentation couleur même si la résolution de l'image semble plus élevée.

Une image peu bruitée facilite la reconnaissance des objets en permettant de restreindre les critères de segmentation. De plus, un temps d'exposition très court est nécessaire si le robot ou les objets observés se déplacent rapidement. Ainsi, une caméra avec une grande ouverture est préférable. Un ajustement de la luminosité est essentiel pour avoir des crontrastes prononcés et un maximum d'information. De plus, l'ajustement de la teinte du blanc doit être constant et adapté aux couleurs définies pour les objets. Dans le cas contraire, la couleur des objets change en fonction du type d'éclairage (naturel, fluorescent ou incandescent) rendant les intervalles de segmentation incorrects. Un ajustement est nécessaire à chaque fois que l'éclairage de l'environnement change rendant ainsi l'utilisation de la luminère naturelle pratiquement impossible.

La figure 2.19 présente un exemple d'image utilisée pour effectuer l'ajustement de la luminosité et des couleurs. Quatre cartons de couleurs sont placés sur le sol à des endroits précis autour du robot. Dans un premier temps, l'algorithme permettant l'ajustement de la lumisonité de l'image détermine l'intensité moyenne des pixels disposés sur les cartons. Cette intensité est calculée en effectuant la moyenne des composantes Y de la représentation YUV des pixels utilisés. Le paramètre de la caméra correspondant au temps d'exposition de l'image est alors corrigé en fonction de la différence entre la luminosité calculée et la luminosité désirée. Par la suite, une autre image est capturée et la même séquence est répétée jusqu'à ce que l'image aie la luminosité désirée. Les cartons sont utilisés afin de s'assurer que l'ajustement est constant d'un robot à l'autre peu importe les conditions d'éclairage.

Figure 2.19: Exemple d'image utilisée pour effectuer l'ajustement des couleurs

Les couleurs sont ajustées à partir d'un algorithme basé sur l'ajustement du blanc. Deux paramètres de la webcam permettent d'ajuster les gains des canaux U et V de l'image. Ces gains sont respectivement proportionnels à la quantité de bleu et de rouge dans l'image. Dans un premier temps, l'algorithme calcule les valeurs moyennes des composantes R, G et B des pixels du carton blanc. Les gains des canaux U et V sont ensuite modifiés en fonction de l'écart entre les composantes B et G ainsi que l'écart entre les composantes R et G. Par la suite, une autre image est capturée et la même séquence est répétée jusqu'à ce que la valeur des composantes R, G et B soit la même. Cet algorithme utilise comme référence seulement le carton blanc. Toutefois, les autres cartons permettent à l'utilisateur de vérifier rapidement l'efficacité de l'algorithme d'ajustement des couleurs en regardant l'image retournée.

Chapitre 3

Système de localisation

3.1 Problématique

Pour interagir avec son environnement, le robot doit connaître sa position. La position du robot est généralement utilisée pour prendre une décision et pour planifier les déplacements. De plus, cette position doit être partagée avec les autres robots présents dans l'environnement afin que chacun tienne compte de la position des autres dans son processus décisionnel et dans ses déplacements. La position du robot est donc une information essentielle qui doit être connue avec précision. Les documents [17], [18] et [6] présentent des solutions similaires permettant de localiser un robot.

Deux types de mesures sont disponibles: l'odométrie (mesure du déplacement angulaire des roues) et la vision (mesure de position relative d'objets par rapport au robot). Le système de localisation a donc pour objectif d'utiliser de manière optimale ces deux types de mesures afin de fournir, le plus précisément possible, la position x, y, l'orientation, la vitesse tangentielle et la vitesse angulaire du robot.

3.1.1 Types de mesures

3.1.2 Vision et reconnaissance des objets

Tel que discuté dans la section 1.1.2, le système de perception a été conçu pour permettre au robot de fonctionner dans un environnement similaire à celui prévu pour

la compétition *Robocup*. Dans cet environnement, le robot dispose de 6 points de repère: le but jaune, le but bleu, deux cylindres de couleurs jaune, bleu et jaune ainsi que deux autres de couleurs bleu, jaune et bleu. Ainsi, pour généraliser, le système de localisation recherche des balises pour effectuer ses mesures. Une **balise** est un assemblage d'un ou de plusieurs objets de forme rectangulaire, cylindrique ou sphérique, de taille, de position et de couleur connues (voir section 2.2). Pour valider une balise, on s'assure qu'un certain nombre des objets qui la composent sont présents. Ainsi, les balises situées dans les coins sont constituées de trois objets: un cylindre jaune, un cylindre bleu et un autre cylindre jaune empilés les uns par-dessus les autres. Étant donné que l'extérieur des buts est blanc, un but peut être représenté par un rectangle de la même couleur disposé à l'ouverture du but. Les figures 3.1 et 3.2 présentent le robot dans un environnement similaire à la compétition *Robocup* à deux positions différentes et les images panoramiques résultantes.

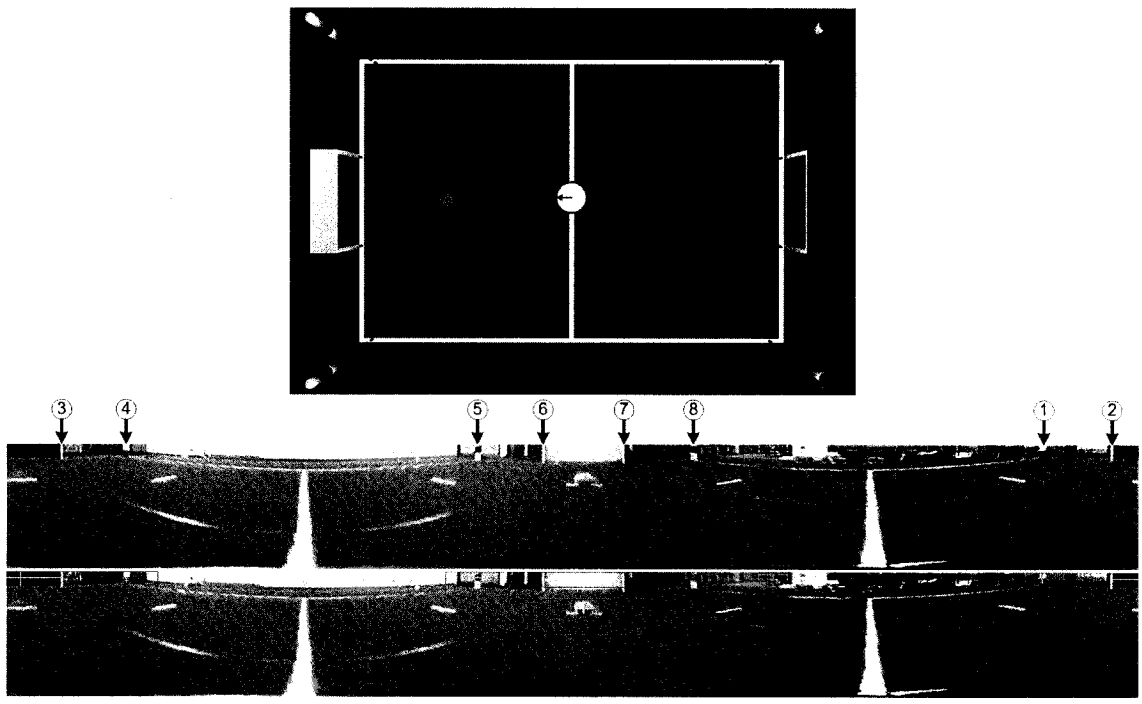

Figure 3.1: Situation A: Environnement, image panoramique et zones de recherche

On remarque sur les images que la plupart des objets ne sont pas vus en totalité, car la hauteur du champ de vision du robot n'est pas suffisamment grande. Ceci

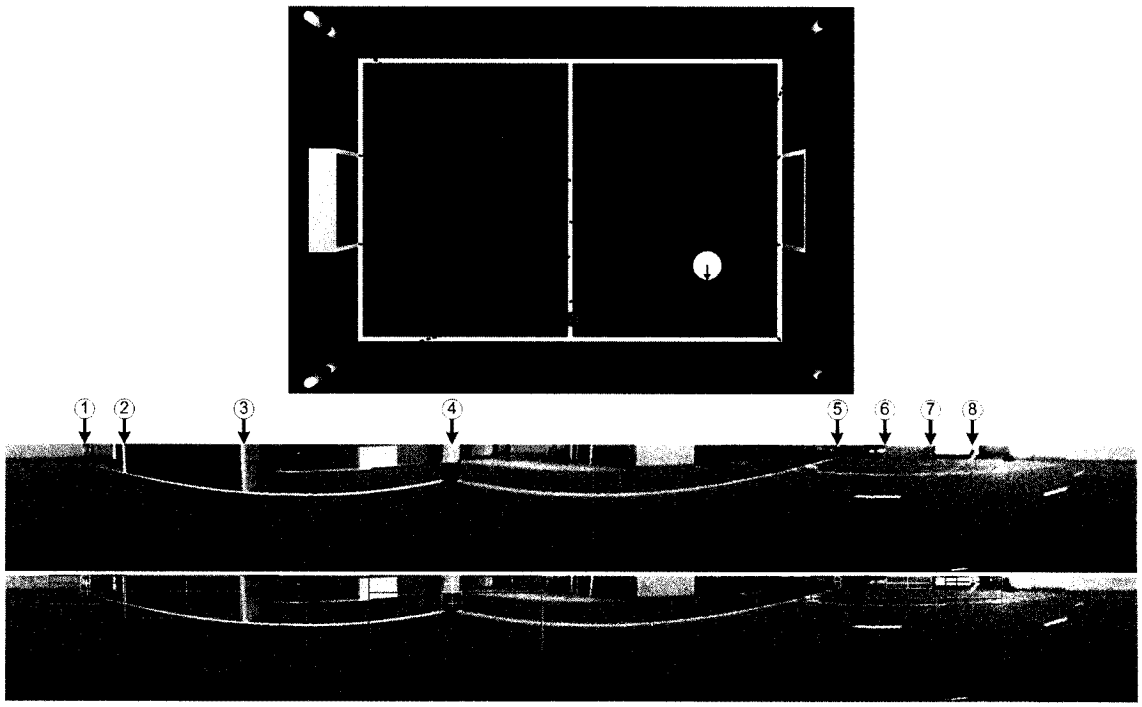

Figure 3.2: Situation B: Environnement, image panoramique et zones de recherche

fait en sorte que la hauteur du centre de masse de la zone de pixels correspondant à l'objet dans l'image n'est pas représentative de la position réelle du centre de masse. Toutefois, puisque la base des objets est visible, la limite y_{max} de la zone de pixels peut être utilisée pour évaluer la distance entre la base de l'objet et le robot. On peut remarquer sur les images que l'étalonnage du montage fait en sorte que l'angle β est faussé, car les prédictions de la position des objets dans l'image (les carrés jaunes) sont situées au-dessus des positions réelles. De plus, bien que les balises cylindriques des coins du terrain sont toutes disposées à une distance égale du robot dans la situation A, leur hauteur dans l'image panoramique varie légèrement. Cette variation est causée principalement par le mauvais alignement du miroir avec la caméra. Ces deux observations font en sorte qu'une mesure β déterminée à partir de l'image ne fournit pas une estimation exacte de la distance mais plutôt une estimation biaisée. Étant donné que le système de localisation doit être précis, cette mesure ne peut pas être utilisée.

La position en x du centre de masse d'une zone correspondant à un objet cylin-

drique ou sphérique dans l'image est une mesure de l'angle α entre le devant de la caméra et l'objet. Cet angle est très précis lorsque la position du centre du miroir dans l'image a été identifiée correctement. Cet étalonnage est simple et il a été effectué avec succès (voir 2.1.2). La position du centre de masse ne peut cependant pas être utilisée pour les objets de forme rectangulaire, car la perspective est plus importante et elle fait en sorte que la position du centre de masse ne coincide pas avec le centre réel de l'objet (voir but bleu sur image 3.2). Dans ce cas, les limites x_{min} et x_{max} qui correspondent parfaitement à la position des extrémités du rectangle sont utilisées comme mesures au lieu du centre de masse. Toutefois, par expérimentation, on a remarqué que le devant de la caméra n'est pas parfaitement aligné avec le devant du robot. Bien que l'écart soit faible, il s'ajoute à chacun des angles α mesurés et il les rend ainsi légèrement biaisés. Les conséquences de ce biais sont discutées dans la section 3.5.1.1. Le filtre de *Kalman* développé observe ce biais et corrige les mesures.

Le système de vision ne fournit pas directement une mesure absolue de la position du robot. Il fournit plutôt un nombre variable de mesures angulaires α entre le devant de la caméra et l'objet. Un **repère** est défini comme étant l'angle α mesuré et la position x et y de la balise correspondante. Lorsqu'une balise est vue, un repère est ajouté à une liste pour le traitement mathématique qui suivra. Ainsi, bien que le nombre de balises soit constant, le nombre de repère peut varier en fonction des balises qui ont été identifiées. Le traitement du système de localisation est donc divisé en deux grandes étapes: la reconnaissance des balises et le calcul de la position à partir des répères. Cela permet d'effectuer différents traitements mathématiques indépendamment de la reconnaissance des balises.

3.1.3 Odométrie

L'odométrie est la mesure du déplacement du robot. Les déplacements angulaires $\Delta\phi_1$ et $\Delta\phi_2$ obtenus par les encodeurs optiques de chacun des moteurs peuvent aussi être représentés par un déplacement tangentiel Δv et un déplacement angulaire $\Delta\omega$ du robot (voir figure 3.3). Les mesures des déplacements angulaires $\Delta\phi_1$ et $\Delta\phi_2$ sont précises à $\pm 2,7318^{-4}$ (voir tableau 1.2). Toutefois, l'imprécision des paramètres D_{roue} et R_{roue} s'ajoute en plus des imprécisions sur l'alignement et la forme des roues. Dans

la situation B, la roue de droite est bloquée et seule la roue de gauche tourne. Le robot tourne alors autour du point de frottement entre la roue de droite et le sol. Ce point, estimé à la position du centre de la surface de contact entre la roue et le sol, s'ajoute aussi aux imprécisions. De plus, lorsqu'il y a un glissement tangentiel ou un glissement latéral, les mesures d'odométrie sont faussées, car le robot ne s'est pas déplacé de la distance mesurée. Ces glissements dépendent des accélérations du robot, de sa vitesse tangentielle dans les virages et du revêtement de l'environnement. Par exemple, un tapis entraîne beaucoup plus de glissement qu'une surface rigide en caoutchou.

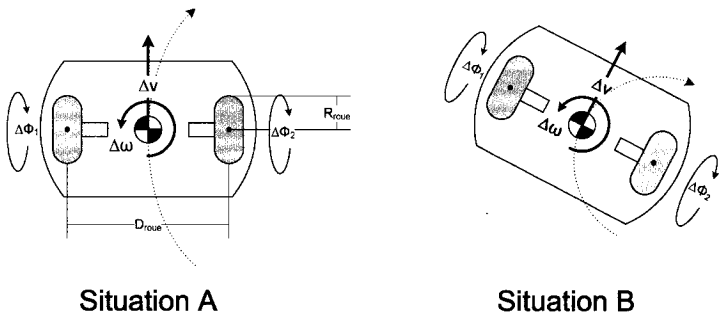

Figure 3.3: Odométrie Situations A et B

Malgré ces imprécisions, les mesures d'odométrie demeurent précises pour un petit déplacement. Lors de grands déplacements, les erreurs de mesures s'accumulent rapidement et la position du robot devient vite erronée. De plus, la sommation des Δv et $\Delta\omega$ est conforme au modèle non-linéaire du déplacement du robot seulement pour de petites valeurs de Δv et de $\Delta\omega$. Les mesures d'odométrie doivent donc être échantillonnées fréquemment.

$$\Delta v = R_{roue}\frac{\Delta\phi_1 + \Delta\phi_2}{2} \tag{3.1}$$

$$\Delta\omega = R_{roue}\frac{\Delta\phi_1 - \Delta\phi_2}{2D_{roue}} \tag{3.2}$$

$$x_n = x_{n-1} + \cos\theta_{n-1}\Delta v \tag{3.3}$$

$$y_n = y_{n-1} + \sin\theta_{n-1}\Delta v \tag{3.4}$$

$$\theta_n = \theta_{n-1} + \Delta\omega \tag{3.5}$$

3.2 Triangulation de la position

Cette section propose deux types de solution pour trouver la position du robot à partir des repères transmis par le système de vision après la reconnaissance des balises. Ces méthodes ont été développées principalement pour évaluer la précision des mesures de vision. Les mesures d'odométrie et de vision sont utilisées directement par un filtre de *Kalman* qui a l'avantage de les combiner de manière optimale. Les deux solutions présentées dans cette section utilisent les mesures de vision uniquement.

La figure 3.4 illustre la problématique pour quatre repères. Les inconnus sont la position x et y ainsi que l'orientation θ du robot. La position x_n et y_n de chacun des repères ainsi que la mesure de l'angle α_n correspondant à l'angle entre le devant du robot et un repère sont connus. Le nombre de repères varie d'une itération à l'autre en fonction du nombre de balises identifiées. Toutefois, un minimum de trois repères est requis pour trouver une solution.

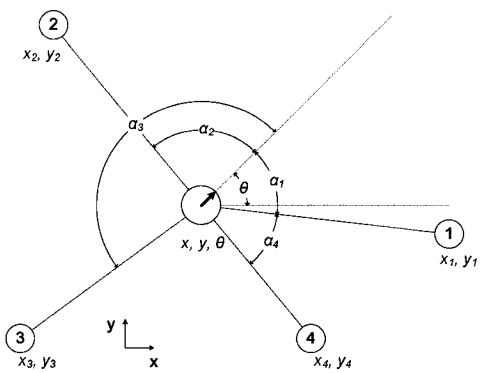

Figure 3.4: Triangulation

3.2.1 Solution analytique

Malgré les apparences, la solution analytique du problème illustré à la figure 3.5, avec trois inconnus (x, y et θ) et trois repères (x_n, y_n et α_n) n'est pas triviale. Il existe toutefois plusieurs types de problèmes similaires (voir [19]). Il est donc plus simple de

modifier la forme du problème pour le ramener sous la forme d'un problème typique à savoir l'intersection de deux cercles de centre cx_n, cy_n et de rayon R_n (voir figure 3.6). Dans un premier temps, il faut déterminer l'équation des cercles passant par deux repères et la position du robot.

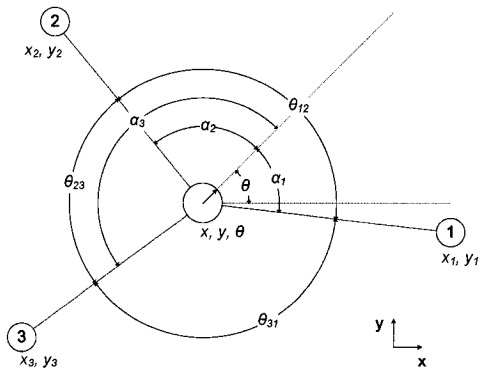

Figure 3.5: Triangulation analytique

Soit n et k deux repères différents et $\vec{on}$ et $\vec{ok}$ les vecteurs formés par la différence entre la position du robot et celle de chacun des repères. Sachant que l'angle $0 \leq \theta_{nk} < \pi$, et à l'aide de la figure 3.5, on peut déduire:

$$\begin{aligned}
\cos\theta_{nk} &= \frac{\vec{on}\cdot\vec{ok}}{|\vec{on}||\vec{ok}|} \\
\sin\theta_{nk} &= \frac{|\vec{on}\times\vec{ok}|}{|\vec{on}||\vec{ok}|} \\
\tan\theta_{nk} &= \frac{|\vec{on}\times\vec{ok}|}{\vec{on}\cdot\vec{ok}} \\
&= \frac{x_n y_k - x y_k - y x_n + xy - x_k y_n + y x_k + x y_n - xy}{x_n x_k - x x_k + x x_n + x^2 + y_n y_k - y y_k - y y_n + y^2} \qquad (3.6) \\
\tan\theta_{nk} &= \gamma \qquad (3.7)
\end{aligned}$$

$$\begin{aligned}
\gamma x_n x_k - \gamma x(x_k + x_n) + \gamma x^2 + \gamma y_n y_k - \gamma y(y_n + y_k) + \gamma y^2 &= \\
x_n y_k + x(y_n - y_k) + y(x_k - x_n) - x_k y_n & \\
\gamma y^2 - (\gamma(y_n + y_k) + (x_k - x_n))\, y + \gamma x^2 - (\gamma(x_n + x_k) + (y_n - y_k))\, x & \\
+\gamma x_n x_k + \gamma y_n y_k - x_n y_k + x_k y_n &= 0 \qquad (3.8)
\end{aligned}$$

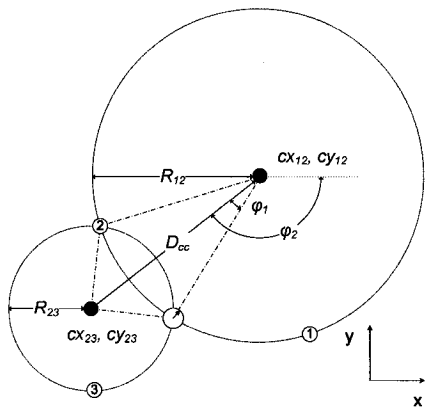

Figure 3.6: Triangulation analytique, intersection de deux cercles

$$\begin{aligned} a &= \gamma \\ b &= \gamma x_n + \gamma x_k + y_n - y_k \\ c &= \gamma y_n + \gamma y_k + x_k - x_n \\ d &= \gamma x_n x_k + \gamma y_n y_k - x_n y_k + x_k y_n \end{aligned}$$

L'équation suivante peut être arrangée sous la forme de l'équation d'un cercle.

$$\begin{aligned} ay^2 + by + ax^2 + cx + d &= 0 \\ y^2 + \frac{b}{a}y + ax^2 + \frac{c}{a}x + d &= 0 \\ \left(y + \frac{b}{2a}\right)^2 + \left(x + \frac{c}{2a}\right)^2 &= \frac{b^2 + c^2 - 4ad}{4a^2} \end{aligned} \tag{3.9}$$

Ainsi les paramètres du cercle sont:

$$\begin{aligned} cx_{nk} &= \frac{-b}{2a} \\ &= \frac{\gamma x_n + \gamma x_k + y_n - y_k}{2\gamma} \end{aligned} \tag{3.10}$$

$$\begin{aligned} cy_{nk} &= \frac{-c}{2a} \\ &= \frac{\gamma y_n + \gamma y_k + x_k - x_n}{2\gamma} \end{aligned} \tag{3.11}$$

$$\begin{aligned} R_{nk} &= \frac{\sqrt{b^2 + c^2 - 4ad}}{2a} \\ &= \frac{\sqrt{(1+\gamma^2)\left(y_k^2 - 2y_k y_n + y_n^2 + x_k^2 - 2x_n x_k + x_n^2\right)}}{2\gamma} \\ &= \frac{\sqrt{(1+\gamma^2)\left((y_k - y_n)^2 + (x_k - x_n)^2\right)}}{2\gamma} \\ &= \frac{\sqrt{(y_k - y_n)^2 + (x_k - x_n)^2}}{2\sin\theta_{nk}} \end{aligned} \tag{3.12}$$

À partir de ces équations, on peut déterminer les paramètres cx, cy et R des deux cercles. La position du robot est alors déterminée par:

$$D_{cc} = \sqrt{(cx_{23} - cx_{12})^2 + (cy_{23} - cy_{12})^2} \tag{3.13}$$

$$\varphi_1 = \arccos\left(\frac{R_{12}^2 - R_{23}^2 + D_{cc}^2}{2R_{12}D_{cc}}\right) \tag{3.14}$$

$$\varphi_2 = \arctan 2\left(cy_{23} - cy_{12}, cx_{23} - cx_{12}\right) \tag{3.15}$$

$$x = R_{12}\cos\left(\varphi_2 \pm \varphi_1\right) + cx_{12} \tag{3.16}$$

$$y = R_{12}\sin\left(\varphi_2 \pm \varphi_1\right) + cy_{12} \tag{3.17}$$

Une des deux solutions est à éliminer puisqu'elle correspond à la position d'un repère. Une fois que la position x et y du robot a été déterminée, l'orientation peut être déduite à partir de l'équation suivante:

$$\theta = \arctan 2\left(y_n - y, x_n - x\right) - \alpha_n \tag{3.18}$$

3.2.2 Solution numérique par moindre carré

À partir d'une liste de repères, on peut aussi déterminer la position et l'orientation du robot en utilisant une méthode numérique par moindre carré. Cette méthode a l'avantage de trouver la position qui correspond le mieux à une liste de 3 repères ou plus. Toutefois, cette méthode itérative nécessite un estimé initial ($\hat{x}$, $\hat{y}$ et $\hat{\theta}$) de la position et de l'orientation du robot. Elle ne peut pas être utilisée pour effectuer un recouvrement de la position du robot si la position est inconnue.

L'estimé de la position est utilisé pour estimer la mesure $\hat{\alpha}_n$ de chacun des repères. La figure 3.7 illustre la position réelle du robot ainsi que la position estimée. Afin d'alléger la figure, seul l'angle α_2 et son estimé $\hat{\alpha}_2$ sont illustrés. Référez-vous à la figure 3.4 pour voir les mesures disponibles dans cet exemple.

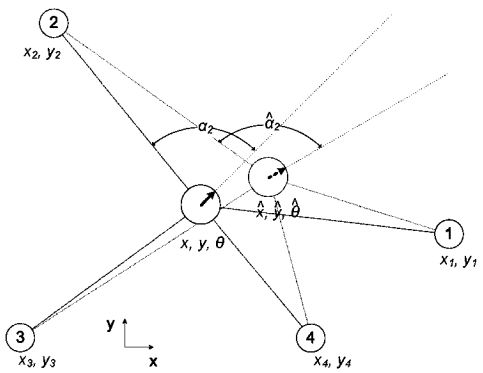

Figure 3.7: Triangulation numérique

L'estimé de la mesure $\hat{\alpha}_n$ est donné par l'équation suivante où x_n et y_n correspondent à la position du repère tandis que $\hat{x}$, $\hat{y}$ et $\hat{\theta}$ correspondent à l'estimé de la position et de l'orientation du robot.

$$\hat{\alpha}_n = \arctan 2(y_n - \hat{y}, x_n - \hat{x}) - \hat{\theta} \tag{3.19}$$

Chaque mesure recueillie est donnée par l'équation suivante:

$$\begin{aligned} \alpha_n &= \arctan 2(y_n - y, x_n - x)) - \theta \\ &= \arctan 2(y_n - (\hat{y} + \delta y), x_n - (\hat{x} + \delta x)) - (\hat{\theta} + \delta\theta) \end{aligned} \tag{3.20}$$

Chaque mesure peut donc être approximée par:

$$\begin{aligned}\alpha_n \approx & \arctan 2(y_n - \hat{y}, x_n - \hat{x}) \\ & -\frac{\partial}{\partial x}\arctan 2(y_n - \hat{y}, x_n - \hat{x})\delta x \\ & -\frac{\partial}{\partial y}\arctan 2(y_n - \hat{y}, x_n - \hat{x})\delta y + \hat{\theta} - \delta\theta \end{aligned} \tag{3.21}$$

Soit:

$$\hat{q} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix} \quad \delta q = \begin{bmatrix} \delta x \\ \delta y \\ \delta\theta \end{bmatrix}$$

$$h(\hat{q}) = \hat{\alpha} = \begin{bmatrix} \arctan 2(y_1 - \hat{y}, x_1 - \hat{x}) - \hat{\theta} \\ \vdots \\ \arctan 2(y_n - \hat{y}, x_n - \hat{x}) - \hat{\theta} \end{bmatrix}$$

$$\frac{\partial}{\partial q}h(\hat{q}) = \begin{bmatrix} \frac{y_1 - \hat{y}}{x_1^2 - 2x_1\hat{x} + \hat{x}^2 + y_1^2 - 2y_1\hat{y} + \hat{y}^2} & \frac{-x_1 + \hat{x}}{x_1^2 - 2x_1\hat{x} + \hat{x}^2 + y_1^2 - 2y_1\hat{y} + \hat{y}^2} & -1 \\ \vdots & \vdots & \vdots \\ \frac{y_n - \hat{y}}{x_n^2 - 2x_n\hat{x} + \hat{x}^2 + y_n^2 - 2y_n\hat{y} + \hat{y}^2} & \frac{-x_n + \hat{x}}{x_n^2 - 2x_n\hat{x} + \hat{x}^2 + y_n^2 - 2y_n\hat{y} + \hat{y}^2} & -1 \end{bmatrix}$$

Ainsi,

$$[\vec{\alpha} - h(\hat{q})] = \frac{\partial}{\partial q}h(\hat{q})\delta q \tag{3.22}$$

$$\delta q = \frac{\partial}{\partial q}h(q)^{-1}[\alpha - h(q)] \tag{3.23}$$

Il faut au moins trois repères pour être en mesure de déterminer la position du robot. Si on a mesuré plus de 3 repères, un moindre carré est effectué comme suit:

$$\delta q = \left(\frac{\partial}{\partial q}h(\hat{q})^T \frac{\partial}{\partial q}h(\hat{q})\right)^{-1} \frac{\partial}{\partial q}h(\hat{q})^T[\alpha - h(\hat{q})] \tag{3.24}$$

La nouvelle position estimée est donnée par:

$$\hat{q}_{nouveau} = \hat{q}_{ancien} + \delta q \tag{3.25}$$

La position estimée converge sur la position réelle après quelques itérations seulement.

3.3 Solution par filtrage de Kalman

3.3.1 Forme générale d'un filtre de Kalman étendu

Le filtre de *Kalman* est un outil puissant permettant d'estimer de manière optimale l'état d'un système en fonction des entrées, d'une modélisation mathématique de la dynamique et des mesures des sorties. En plus d'estimer l'état du système, le filtre estime aussi l'erreur associée à la précision de l'état estimé par rapport à l'état réel du système. Un filtre de *Kalman* discret évolue dans le temps par une étape de prédiction qui évalue l'état courant du système à partir de l'état précédent, des entrées et du modèle dynamique. Une erreur est associée à la prédiction en fonction du bruit sur les entrées et de l'imprécision du modèle. Cette erreur se propage et s'ajoute à l'estimation de l'erreur de l'état après une prédiction. Pour conserver une bonne précision à la suite d'une ou de plusieurs prédictions successives, il est essentiel de corriger l'état du système à l'aide des mesures.

Les mesures, à elles seules, permettent de déterminer l'état du système avec une précision limitée par les bruits de mesures et par la relation entre les mesures et l'état du système. En utilisant conjointement la prédiction et les mesures, le filtre de *Kalman* estime l'état du système avec une précision supérieure à celle resultant des mesures uniquement. La correction apportée par les mesures est pondérée en fonction de l'erreur associée à la prédiction et de l'erreur associée aux mesures. Ainsi, si la prédiction est beaucoup plus précise que les mesures, les corrections apportées par les mesures seront faibles. De la même façon, si les mesures sont beaucoup plus précises que la prédiction, les corrections apportées seront significatives.

Pour bien comprendre les fondements de l'étape de correction d'un filtre de *Kalman*, considérons l'exemple simple de l'estimation d'un scalaire (voir [20]).

Soit m la valeur réelle mais inconnue du nombre estimé, z la mesure effectuée et x_0 l'estimé initial. Les valeurs z et x_0 sont égales à la valeur m augmentée respectivement des variables aléatoires v et w, l'erreur sur les mesures et l'erreur sur l'estimé initial. Ces variables aléatoires indépendantes suivent une distribution normale de moyenne

nulle dont les variances sont respectivement σ_v^2 et σ_w^2.

$$z = m + v \tag{3.26}$$

$$x_0 = m + w \tag{3.27}$$

L'estimateur utilisé pour trouver la correction qui minimise la variance de x_1 a la forme suivante:

$$x_1 = x_0 + k(z - x_0) \tag{3.28}$$

En remplaçant z et x_0, on obtient:

$$\begin{aligned} x_1 &= m + w + k(m + v - m - w) \\ &= m + w + k(v - w) \end{aligned} \tag{3.29}$$

La variance de x_1

$$\begin{aligned} \sigma_1^2 &= E(w + k(v - w))^2 \\ &= (1 - k)^2\sigma_w^2 + k^2\sigma_v^2 \\ &= (\sigma_w^2 + \sigma_v^2)k^2 - 2k\sigma_w^2 + \sigma_w^2 \end{aligned} \tag{3.30}$$

Le k qui minimise l'expression est donné par:

$$\begin{aligned} (\sigma_w^2 + \sigma_v^2)2k - 2\sigma_w^2 &= 0 \\ (\sigma_w^2 + \sigma_v^2)k &= \sigma_w^2 \\ k &= \frac{\sigma_w^2}{\sigma_w^2 + \sigma_v^2} \end{aligned} \tag{3.31}$$

Ainsi, la valeur minimale de σ_1^2 est:

$$\begin{aligned} \sigma_1^2 &= (\sigma_w^2 + \sigma_v^2)\left(\frac{\sigma_w^2}{\sigma_w^2 + \sigma_v^2}\right)^2 - 2\frac{\sigma_w^2}{\sigma_w^2 + \sigma_v^2}\sigma_w^2 + \sigma_w^2 \\ &= \sigma_w^2 - \frac{\sigma_w^4}{\sigma_w^2 + \sigma_v^2} \\ &= \frac{\sigma_w^2\sigma_v^2}{\sigma_w^2 + \sigma_v^2} \end{aligned} \tag{3.32}$$

$$\frac{1}{\sigma_1^2} = \frac{1}{\sigma_w^2} + \frac{1}{\sigma_v^2} \tag{3.33}$$

Dans un filtre de *Kalman*, σ_w^2 représente l'erreur estimée sur la prédiction et σ_v^2 représente l'erreur résultant des mesures. Lorsque l'erreur sur la prédiction est faible par rapport celle sur la mesure ($\sigma_w^2 << \sigma_v^2$), le gain k tend vers 0. À l'inverse, lorsque l'erreur sur la prédiction est élevée par rapport sur la mesure ($\sigma_w^2 >> \sigma_v^2$), le gain k tend vers 1 et la correction s'applique.

3.3.2 Filtre de Kalman étendu

Soit un système discret dont l'état courant du système est donné par le vecteur x. La dynamique du système est décrite par la fonction non-linéaire f et l'état courant est déterminé à partir de l'état précédent et des entrées de la façon suivante:

$$x_k = f\left(x_{k-1}, u_{k-1}, w_{k-1}\right) \tag{3.34}$$

où le vecteur u contient les entrées du système et le vecteur w contient les variables aléatoires résultant du bruit de processus. Le bruit de processus est attribuable aux incertitudes et aux erreurs du modèle mathématique et des entrées.

De plus, la relation entre l'état du système et les mesures est définie par la fonction non-linéaire h où x_k est le vecteur de l'état courant du système et v le vecteur des variables aléatoires résultant du bruit de mesure.

$$z_k = h\left(x_k, v_k\right) \tag{3.35}$$

Toutefois, l'état réel x_k et les vecteurs v et w sont inconnus. Le filtre de *Kalman* estime l'état $\hat{x}_k$ et la covariance de l'erreur P tel que:

$$P_{[n,n]} = E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] \tag{3.36}$$

où n est le nombre d'état du système. Le filtre de *Kalman* non-linéaire est basé sur la linéarisation suivante:

$$\begin{aligned} x_k &\approx f\left(\hat{x}_{k-1}, u_{k-1}, 0\right) + \frac{\partial f}{\partial x}\left(\hat{x}_{k-1}, u_{k-1}, 0\right)(\Delta x) + \frac{\partial f}{\partial u}\left(\hat{x}_{k-1}, u_{k-1}, 0\right)(\Delta u) \\ &\quad + \frac{\partial f}{\partial w}\left(\hat{x}_{k-1}, u_{k-1}, 0\right)(\Delta w) \\ &\approx \tilde{x}_k + A\left(x_{k-1} - \hat{x}_{k-1}\right) + W w_{k-1} \end{aligned} \tag{3.37}$$

$$\begin{aligned} z_k &\approx h(\tilde{x}_k, 0) + \frac{\partial h}{\partial x}(\tilde{x}_k, 0)(\Delta x) + \frac{\partial h}{\partial v}(\tilde{x}_k, 0)(\Delta v) \\ &\approx \tilde{z}_k + H(x_k - \tilde{x}_k) + V v_k \end{aligned} \tag{3.38}$$

où

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.39}$$

$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{3.40}$$

$$A_{[n,n]} = \frac{\partial f_{[n]}}{\partial x_{[n]}}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.41}$$

$$\tag{3.42}$$

$$W_{[n,nw]} = \frac{\partial f_{[n]}}{\partial w_{[nw]}}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.43}$$

$$\tag{3.44}$$

$$H_{[m,n]} = \frac{\partial h_{[m]}}{\partial x_{[n]}}(\tilde{x}_k, 0) \tag{3.45}$$

$$\tag{3.46}$$

$$V_{[m,nv]} = \frac{\partial h_{[m]}}{\partial v_{[nv]}}(\tilde{x}_k, 0) \tag{3.47}$$

$$\tag{3.48}$$

$$Q_{[nw,nw]} = E\left[ww^T\right] \tag{3.49}$$

$$\tag{3.50}$$

$$R_{[nv,nv]} = E\left[vv^T\right] \tag{3.51}$$

$$\tag{3.52}$$

Par hypothèse, les entrées sont connues parfaitement et l'erreur Δu est nulle. Si en pratique ce n'est pas le cas, cette erreur est ajoutée au bruit de processus. Les valeurs n, m, nw et nv correspondent respectivement au nombre d'états, au nombre

de mesures, au nombre de variables aléatoires des bruits de processus et au nombre de variables aléatoires des bruits de mesures. Les matrices Q et R contiennent respectivement la covariance des bruits de processus et la covariance des bruits de mesures.

L'étape de prédiction est décrite par les équations:

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.53}$$

$$\tilde{P}_k = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{3.54}$$

$$\tag{3.55}$$

L'étape de correction est décrite par les équations:

$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{3.56}$$

$$K_k = \tilde{P}_k H_k^T \left(H_k \tilde{P}_k H_k^T + V_k R_k V_k^T\right)^{-1} \tag{3.57}$$

$$\hat{x}_k = \tilde{x}_k + K_k (z_k - h(\tilde{x}_k, 0)) \tag{3.58}$$

$$P_k = (I - K_k H_k) \tilde{P}_k \tag{3.59}$$

$$\tag{3.60}$$

La syntaxe présentée dans cette section est celle présentée dans [21] et elle sera utilisée pour décrire tous les filtres de *Kalman* à l'étude dans ce mémoire. Toutefois, le vecteur d'état x est aussi appelé q pour éviter toute confusion avec la coordonnée en x de la position du robot ou d'un objet.

3.3.3 Modèle mathématique

Le système de localisation dispose de mesures d'odométries (déplacements tangentiel et angulaire) ainsi que d'un nombre variable de mesures de vision (les repères) pour estimer la position x, y et l'orientation θ du robot. Les mesures d'odométrie sont précises pour de petits déplacements et simples à échantionner. Toutefois, la trajectoire calculée à partir de la sommation des mesures d'odométrie dérive rapidement de la trajectoire réelle. Les mesures de vision permettent d'obtenir la position absolue du robot dans l'environnement. Ces mesures moins précises ont l'avantage de ne pas dériver avec le temps. Par contre, ces mesures sont moins fréquentes et elles nécessitent un temps de traitement beaucoup plus long. La figure 3.8 illustre

des exemples de trajectoires basées uniquement sur les mesures d'odométrie et de vision. Le filtre de Kalman du système de localisation combine ces deux mesures afin de bénéficier de leurs avantages respectifs.

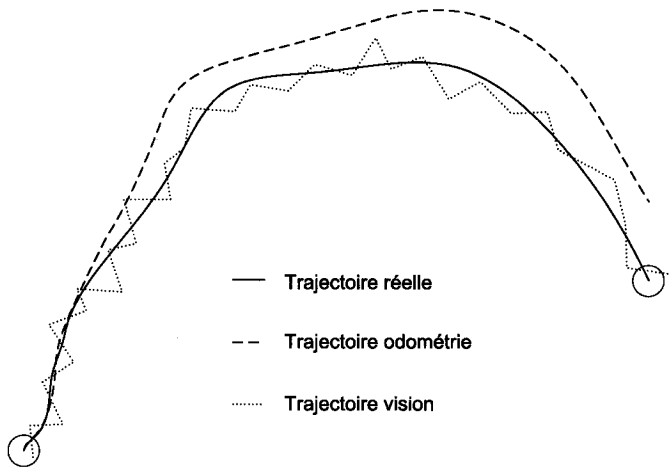

Figure 3.8: Exemples de trajectoires mesurées par l'odométrie et la vision

Étant donné que les tensions appliquées aux moteurs sont inconnues, il est impossible de prédire la position du robot à partir du modèle dynamique du robot. Les mesures d'odométries sont donc utilisées pour prédire la nouvelle position du robot et, ainsi, assurer l'évolution du filtre de *Kalman* dans le temps. Les mesures de vision seront utilisées pour corrigée la trajectoire prédite. La figure 3.9 illustre la prédiction basée sur l'odométrie et la correction de la vision. Sur la figure, les cercles pointillés symbolisent l'erreur associée aux mesures à chaque itération. De plus, compte tenu que la caméra n'est pas parfaitement alignée avec le devant du robot, le filtre de *Kalman* estime le biais constant qui s'ajoute à toutes les mesures fournies par le système de vision.

Soit:

$$\hat{q} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \\ \hat{b} \end{bmatrix} \quad u = \begin{bmatrix} \Delta v \\ \Delta w \\ \Delta t \end{bmatrix}$$

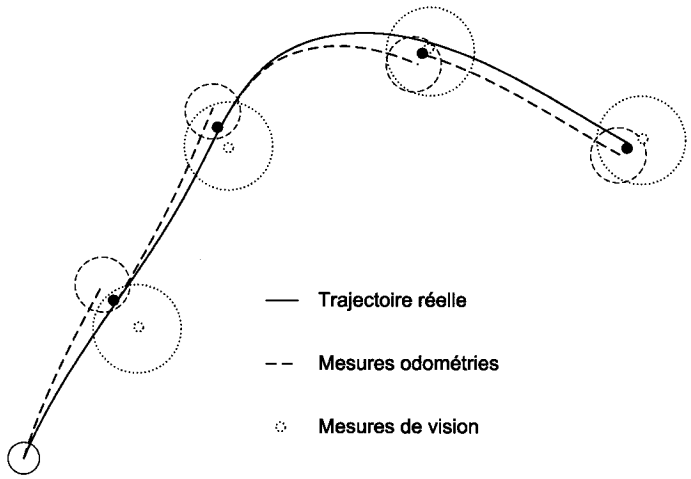

Figure 3.9: Combinaison des mesures d'odométrie et de vision par le filtre de Kalman

Les expressions du filtre de *Kalman* sont données par:

$$f\left(\hat{q}_{k-1}, u_{k-1}, 0\right) = \begin{bmatrix} \hat{x} + \cos\theta\Delta v \\ \hat{y} + \sin\theta\Delta v \\ \hat{\theta} + \Delta w \\ \hat{b} \end{bmatrix} \tag{3.61}$$

$$A = \begin{bmatrix} 1 & 0 & -\sin\theta\Delta v & 0 \\ 0 & 1 & \cos\theta\Delta v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.62}$$

$$W = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.63}$$

$$Q = \begin{bmatrix} B_v|\Delta v| + B_{vs}|\Delta t| & 0 & 0 & 0 \\ 0 & B_g|\Delta v| + B_{gs}|\Delta t| & 0 & 0 \\ 0 & 0 & B_w|\Delta w| + B_{ws}|\Delta t| & 0 \\ 0 & 0 & 0 & B_b \end{bmatrix} \tag{3.64}$$

$$h(\tilde{q})_{[m,1]} = \begin{bmatrix} \arctan 2(y_1 - \tilde{y}, x_1 - \tilde{x}) - \tilde{\theta} - \tilde{b} \\ \vdots \\ \arctan 2(y_m - \tilde{y}, x_m - \tilde{x}) - \tilde{\theta} - \tilde{b} \end{bmatrix} \tag{3.65}$$

$$H_{[m,4]} = \begin{bmatrix} \frac{y_1 - \tilde{y}}{x_1^2 - 2x_1\tilde{x} + \tilde{x}^2 + y_1^2 - 2y_1\tilde{y} + \tilde{y}^2} & \frac{-x_1 + \tilde{x}}{x_1^2 - 2x_1\tilde{x} + \tilde{x}^2 + y_1^2 - 2y_1\tilde{y} + \tilde{y}^2} & -1 & -1 \\ \vdots & \vdots & \vdots & \\ \frac{y_m - \tilde{y}}{x_m^2 - 2x_m\tilde{x} + \tilde{x}^2 + y_m^2 - 2y_m\tilde{y} + \tilde{y}^2} & \frac{-x_m + \tilde{x}}{x_m^2 - 2x_m\tilde{x} + \tilde{x}^2 + y_m^2 - 2y_m\tilde{y} + \tilde{y}^2} & -1 & -1 \end{bmatrix} \tag{3.66}$$

$$V_{[m,m]} = \begin{bmatrix} 1 & 0 & & 0 \\ 0 & 1 & & 0 \\ & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.67}$$

$$R_{[m,m]} = \begin{bmatrix} B_z & 0 & & 0 \\ 0 & B_z & & 0 \\ & & \ddots & \vdots \\ 0 & 0 & \cdots & B_z \end{bmatrix} \tag{3.68}$$

Étant donné le nombre variable de mesures de vision m, la taille du vecteur $h(\tilde{q})$ et des matrices H, V et R est variable et proportionnelle à m. Ces matrices doivent donc être redimensionnées à chaque itération en fonction du nombre de repères. De plus, pour effectuer la différence entre les mesures réelles et celles estimées, il faut prendre en compte qu'il s'agit d'angles et éviter les multiples de 2π.

Les bruits de processus sont définis dans la matrice Q en fonction du déplacement et du temps plutôt qu'en fonction d'une itération. En effet, il est illogique d'associer la même erreur (par exemple $\pm 2cm$) à un déplacement de $1cm$ et à un déplacement de $2m$. De plus, afin d'éviter que le filtre soit complètement inactif lorsque le robot est à l'arrêt ($\Delta v = 0$ et $\Delta w = 0$), des bruits de processus en fonction du temps écoulé depuis la dernière mise à jour ont été ajoutés. Le filtre peut alors converger plus

rapidement après un glissement des roues ou une collision. De la même façon, bien que le biais estimé est constant, il est préférable de conserver un bruit B_b légèrement supérieur à 0 afin de garder le filtre actif. De plus, le bruit de mesure B_z sur les mesures de vision est considéré constant d'une mesure à l'autre et l'erreur sur la position des balises est considérée nulle. Ainsi, une balise mal positionnée fausse inévitablement la position estimée du robot. En résumé, le tableau 3.1 décrit les symbôles utilisés pour les états, les entrées et les bruits.

Tableau 3.1: États, entrées, bruits de processus et bruits de mesures

Symbôles	***Description***	***Unités***
$\hat{x}$	Estimation de la position en x du robot	m
$\hat{y}$	Estimation de la position en y du robot	m
$\hat{\theta}$	Estimation de l'orientation du robot	rad
Δv	Mesure d'odométrie du déplacement tangentiel	m
Δw	Mesure d'odométrie du déplacement angulaire	rad
Δt	Temps écoulé depuis la dernière mise à jour	s
B_v	Bruit sur le déplacement tangentiel (déplacement effectué)	$\frac{m^2}{m}$
B_w	Bruit sur le déplacement angulaire (déplacement effectué)	$\frac{rad^2}{rad}$
B_g	Bruit sur le glissement latéral (déplacement effectué)	$\frac{m^2}{m}$
B_{vs}	Bruit sur le déplacement tangentiel (temps écoulé)	$\frac{m^2}{s}$
B_{ws}	Bruit sur le déplacement angulaire (temps écoulé)	$\frac{rad^2}{s}$
B_{gs}	Bruit sur le glissement latéral (temps écoulé)	$\frac{m^2}{s}$
B_b	Bruit de processus sur l'estimation du biais	rad^2
B_z	Bruit sur les mesures de vision	rad^2

3.3.4 Mise en oeuvre et ajustement des paramètres

Plutôt que d'utiliser une librairie de calculs matriciels en *C++* et de réaliser le filtre de *Kalman* selon les expressions présentées à la section 3.3.1, nous avons préféré utiliser notre propre librairie basée sur des algorithmes différents. Les algorithmes utilisés, basés sur une décomposition UDU, ont été optimisés spécifiquement pour

l'application d'un filtre de *Kalman* étendu. Les résultats sont identiques mais les calculs sont plus rapides et plus robustes numériquement. Cette librairie *C++* a été réalisée à partir des algorithmes décrits dans [22], en collaboration avec le professeur M. Gourdeau et un confrère, Vincent Zalzal, tous deux de l'École Polytechnique de Montréal.

Le contexte réel de la mise en oeuvre du filtre de *Kalman* a nécessité quelques modifications. Dans un premier temps, la période de la boucle de contrôle du robot est en moyenne d'environ 20 ms, mais elle varie continuellement entre 5 ms et 40 ms. La décision prise par le robot à chacune des itérations de la boucle de contrôle doit être basée sur la position courante du robot et nécessite donc l'utilisation des plus récentes mesures d'odométrie. De plus, la sommation des mesures d'odométrie est plus précise lorsqu'elle est effectuée à un intervalle rapproché. Dans un second temps, l'ordinateur du robot parvient à traiter les images fournies par la caméra à un intervalle d'environ 100 ms. Ainsi, l'acquisition des mesures d'odométrie doit être réalisée à un intervalle irrégulier environ 5 fois plus court que l'intervalle d'acquisition des mesures de vision. L'acquisition des deux types de mesures ne peut donc pas se faire en même temps et il devra être effectué dans deux processus concurrentiels différents. Dans un troisième temps, étant donné que les mesures de vision nécessitent un temps de traitement non-négligeable d'environ 100 ms, la correction apportée est en retard par rapport à la position réelle du robot.

La solution suivante a donc été retenue:

- À chaque acquisition (odométrie ou image), on effectue aussi l'acquisition d'une référence absolue dans le temps.
- Toutes les mesures d'odométrie sont préservées dans un vecteur avec leur référence dans le temps.
- Le filtre de *Kalman* du système de localisation fonctionne en retard, synchronisé avec l'acquisition et le traitement des images.
- La position actuelle du robot est déterminée à partir de l'estimé du filtre et de la sommation des mesures d'odométrie depuis la dernière mise à jour du filtre.

- Pour synchroniser les mesures d'odométrie avec l'acquisition de l'image, une approximation linéaire entre les deux mesures d'odométrie les plus proches est effectuée.

- Après la mise à jour du filtre, les mesures d'odométries précédant la mise à jour sont retirées du vecteur.

- Lorsqu'aucune mesure de vision n'est disponible, le filtre évolue avec la prédiction de l'odométrie sans correction par la vision. Cela évite de remplir inutilement le vecteur de mesures d'odométrie.

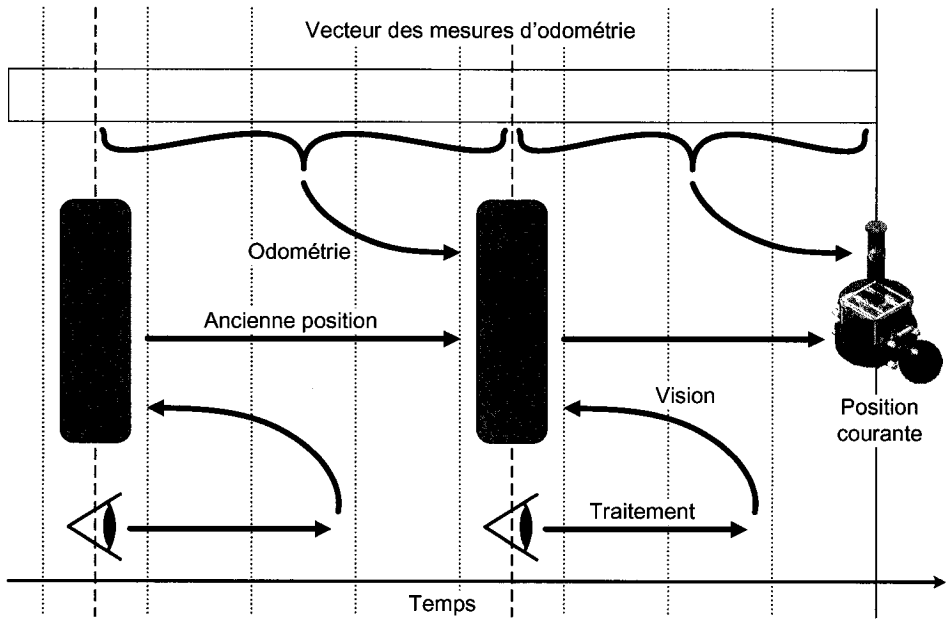

Figure 3.10: Filtre de Kalman synchronisé avec les mesures de vision en retard

Étant donné qu'il y a plus de mesures d'odométrie, le filtre de *Kalman* les incorpore en faisant plusieurs prédictions subséquentes avant d'effectuer la correction basée sur les mesures de vision. Le filtre réalisé en *C++* a été validé en comparant ses résultats avec un filtre mis en oeuvre sous *Matlab*. Les entrées, les mesures et les états estimés ont été sauvegardés pendant l'exécution pour permettre aux scripts *Matlab* d'effectuer le même traitement et de comparer les résultats. Cette procédure fastidieuse a permi de vérifier le bon fonctionnement du filtre et d'ajuster les paramètres.

Les paramètres du filtre dépendent de plusieurs facteurs et il a été jugé préférable de les ajuster principalement par essais et erreur. Pour quantifier les bruits sur les déplacements mesurés par l'odométrie (B_v et B_w), on a mesuré grossièrement l'erreur de position après un long déplacement tangentiel de type aller et retour ainsi qu'après un long déplacement angulaire. Par la suite, des trajectoires réelles mesurées à l'aide de l'odométrie ont été comparées avec des trajectoires simulées pour valider les paramètres trouvés. Le bruit B_g qui représente le glissement latéral en fonction du déplacement a été posé arbitrairement en évaluant son impact sur les trajectoires simulées. Il a été remarqué qu'une sous-estimation du bruit sur l'odométrie peut entraîner une dérive ou un écart constant par rapport à la position réelle. Le bruit d'odométrie ne suit pas une distribution normale de moyenne nulle et de variance constante. Il est aussi fonction de la précision des paramètres et de la surface de roulement. Par exemple, un tapis ou une surface cahoteuse peuvent aussi réduire la précision de l'odométrie. De plus, l'odométrie n'est pas infaillible. Les roues peuvent glisser lors d'accélérations prononcées ou de collisions. Les bruits de mesure liés à l'odométrie doivent être augmentés lorsqu'il y a des risques de glissements ou de collisions afin de limiter leurs conséquences. Les bruits B_{vs}, B_{gs} et B_{ws} ont été ajoutés pour assurer une estimation minimale du bruit et limiter l'impact des imprécisions, du glissement et des collisions. Ces bruits ont été déterminés par essais erreur en observant l'écart entre la trajectoire estimée et la trajectoire mesurée par la vision ainsi qu'en observant l'impact des collisions.

Le bruit de mesures B_z sur les angles retournées par le système de vision peut être évalué à partir de la résolution de l'image panoramique. Étant donné que les mesures de vision sont légèrement biaisées par le mauvais alignement du miroir par rapport à la caméra, il est important de s'assurer que l'estimation du bruit B_z est suffisamment élevée pour limiter l'impact de ces biais. L'ajustement des bruits doit permettre un bon fonctionnement du filtre et une estimation réaliste de l'erreur de position. Le tableau 3.2 présente les covariances retenues pour les bruits de processus et les bruits de mesure du filtre de *Kalman* du système de localisation.

Tableau 3.2: Estimations des bruits de processus et des bruits de mesure

Bruits	***Valeurs***
B_v	$0.0001\ \frac{m^2}{m}$
B_w	$0.03\ \frac{rad^2}{rad}$
B_g	$0.000025\ \frac{m^2}{m}$
B_{vs}	$0.0005\ \frac{m^2}{s}$
B_{ws}	$0.01523085\ \frac{rad^2}{s}$
B_{gs}	$0.0005\ \frac{m^2}{s}$
B_b	$0.0000001\ rad^2$
B_z	$0.000438649\ rad^2$

3.4 Résultats

3.4.1 Simulation

Les simulations sont effectuées à l'aide de l'application temps-réel de la même manière que les expérimentations à l'exception que les mesures d'odométrie et de vision sont simulées. Les mesures simulées sont les mesures exactes calculées à partir de la position exacte du robot additionnées des bruits de mesures. La simulation permet alors d'évaluer les performances du filtre dans des conditions optimales de fonctionnement. En pratique, les performances sont moins bonnes car les bruits réels ne sont généralement pas blanc en plus de présenter de légers biais. Toutefois, les mesures de vision sont simulées avec un biais constant afin d'évaluer la capacité du filtre à estimer le biais entre la caméra et l'orientation du robot. Étant donnée que la position exacte du robot est connue en simulaton, il est possible de comparer l'erreur estimée avec l'erreur réelle entre la position estimée et la position du robot. L'erreur de position estimée est calculée à partir des termes sur la diagonale de la matrice P.

$$E_{position} = \left((\sigma_x^2)^2 + (\sigma_y^2)^2\right)^{\frac{1}{4}}$$

La figure 3.11 présente une trajectoire simulée (position et orientation) en comparaison avec les trajectoires basées uniquement sur l'odométrie ou la vision et la trajectoire estimée à partir des deux types de mesure par le filtre de *Kalman*. La

dérive rapide de la trajectoire d'odométrie est causée par l'intégrale du bruits de mesure sur l'odométrie et par l'erreur sur la position et l'orientation initiale du robot. Les mesures de vision sont simulées en considérant un environnement similaire aux figures 3.1 et 3.2 où le terrain mesure $6m$ par $9m$. La figure 3.12 présente les erreurs de position et d'orientation. On remarque que l'erreur estimée est similaire à l'erreur sur la position estimée. Tel que prévu, le filtre estime la position du robot plus précisément que la position basée uniquement sur les mesures de vision. La figure 3.13 présente l'estimation du biais et le nombre de repères en fonction du temps.

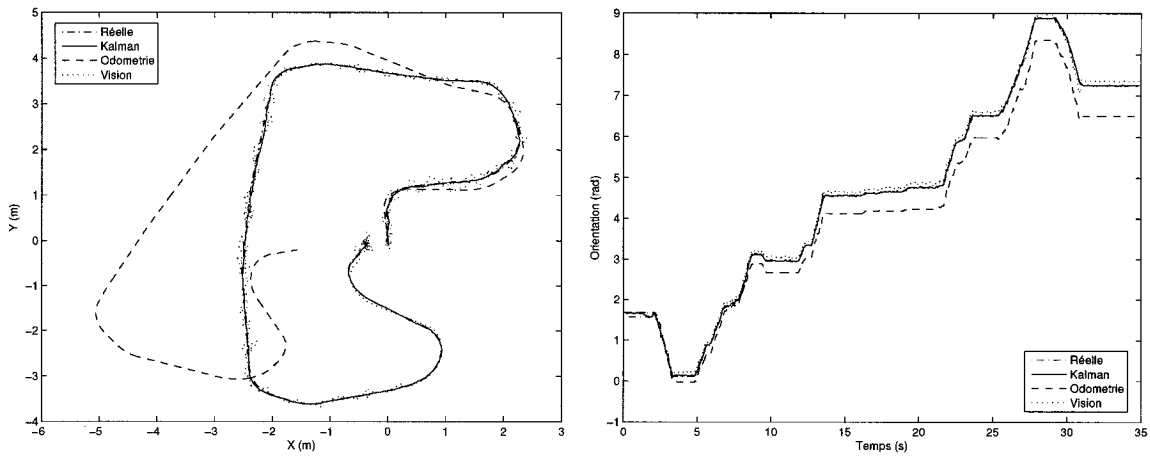

Figure 3.11: Trajectoire et orientation du robot

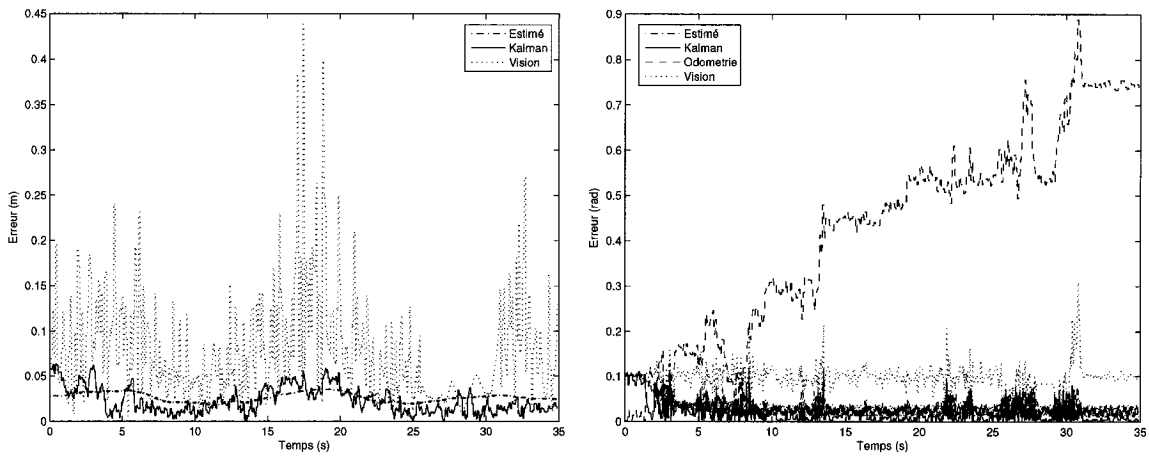

Figure 3.12: Erreurs de position et d'orientation

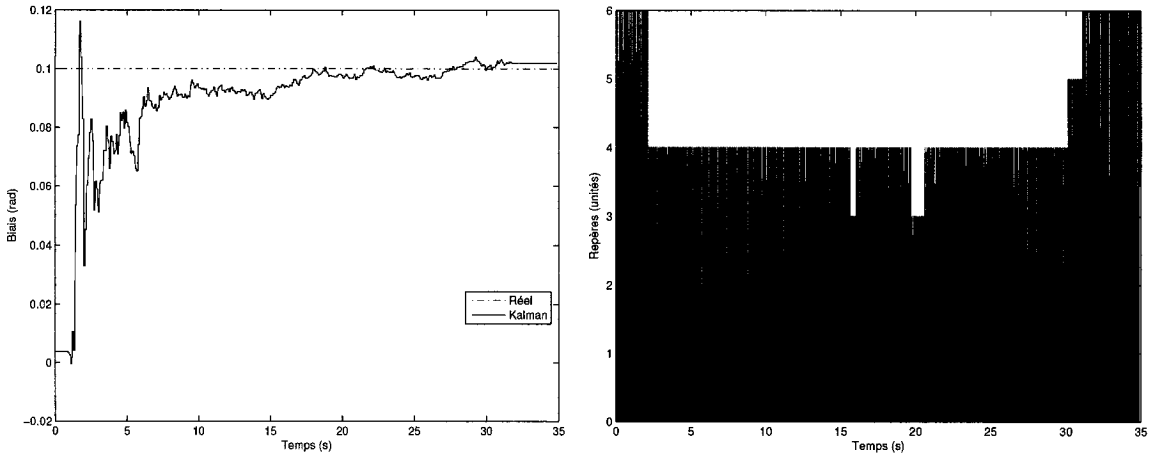

Figure 3.13: Biais et nombre de repères

3.4.2 Expérimentation

Les figures 3.14 et 3.15 présentent les mesures et les estimations effectuées pendant que le robot PGE3 a parcouru une trajectoire donnée. La taille de l'environnement est de $3m$ par $6m$ et quatre balises sont disposées, une dans chaque coin. Étant donné que la trajectoire réelle est inconnue, il est plus difficile d'évaluer la performance du filtre. On remarque toutefois que la trajectoire estimée est plus lisse que la trajectoire basée sur la vision et elle ne dérive pas comme la trajectoire de l'odométrie.

La trajectoire basée sur les mesures de vision a été construite à partir de l'algorithme décrit à la section 3.2.2. Puisque l'algorithme nécessite un estimé initial de la position du robot, les positions sans mesure de vision sont prédites par l'ajout du déplacement d'odométrie à la position précédente. Ces prédictions font en sorte que la trajectoire de vision est en forme de dents de scie.

Bien que la trajectoire réelle soit inconnue, l'erreur estimée semble approximativement du même ordre de grandeur que l'erreur réelle. Cette approximation est basée sur la comparaison de la position estimée avec des positions connues marquées dans l'environnement. La forme en dents de scie de la courbe de l'erreur estimée est causée par la prédiction basée sur l'odométrie et de la correction moins fréquente apportée par les mesures de vision. Chaque prédiction ajoute une imprécision à la position es-

timée et, par conséquent, à l'erreur estimée. L'erreur estimée est diminuée à la suite d'une correction, car elle devient alors inférieure à la plus petite des deux erreurs entre l'erreur de la position prédite et l'erreur de la position pouvant être calculée par les mesures de vision. Cette estimation de l'erreur est utilisée par le système de vision pour déterminer la taille des zones de recherches dans l'image (voir 2.2.1).

Les performances du filtre mis en oeuvre sur le robot sont généralement moins bonnes que les performances du filtre simulé. En simulation, on obtient les performances escomptées car toutes les hypothèses de base sont respectées. En pratique, ce n'est pas le cas et les performances du filtre sont limitées dans certaines conditions. La section 3.5 traite des problèmes rencontrés et des limitations du système de localisation.

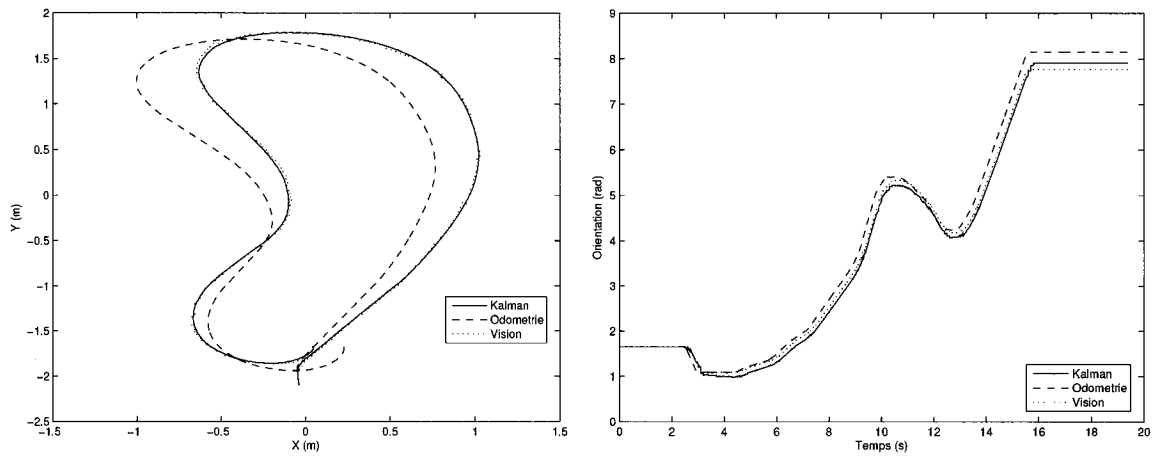

Figure 3.14: Trajectoire et orientation du robot

Les figures 3.16 et 3.17 présentent une situation où, pendant un intervalle d'environ $4s$, il n'y a plus de correction apportée par les mesures de vision. La trajectoire illustrée commence à droite pour tracer grossièrement un cercle dans le sens anti-horaire. L'intervalle sans mesure de vision est délimité par deux + sur les trajectoires basées sur les mesures de vision. Avant la perte des mesures, le filtre est en quelque sorte en régime permanent. Les corrections sont faibles et l'erreur estimée varie peu. Pendant l'intervalle sans mesure de vision, le filtre de *Kalman* poursuit son estimation de la position du robot en n'utilisant cette fois que les mesures d'odométrie. Étant

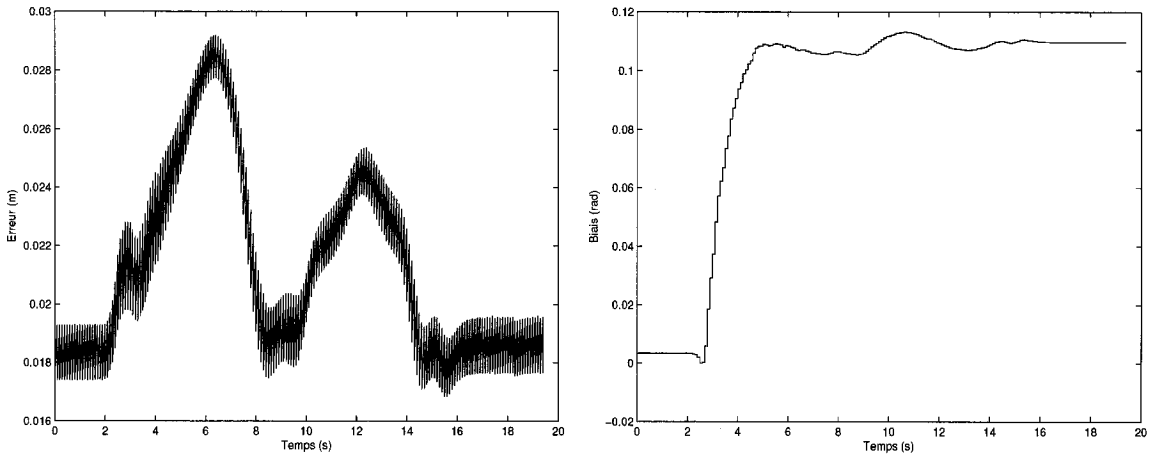

Figure 3.15: Erreurs sur la position et le biais estimé

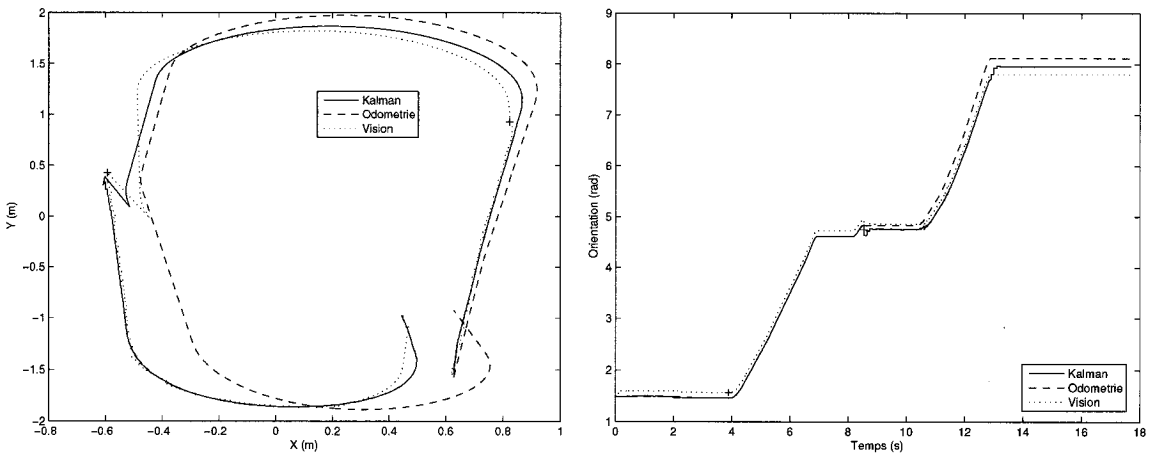

Figure 3.16: Trajectoire et orientation du robot avec perte de la vision

donné qu'il n'y a pas de correction, l'erreur augmente rapidement en fonction du temps écoulé et du déplacement effectué. Lorsque des mesures de vision sont à nouveau disponibles, les corrections apportées sont importantes afin d'éliminer l'erreur accumulée. Ensuite, l'estimation se poursuit et le filtre revient rapidement en régime permanent. Cet essais a été réalisé en éteignant simplement les lumières afin de rendre la reconnaissance des balises impossibles. Ceci démontre bien que, sans prédisposition particulière, le système de localisation estime correctement la position du robot même si le nombre de mesures diminue et devient nul.

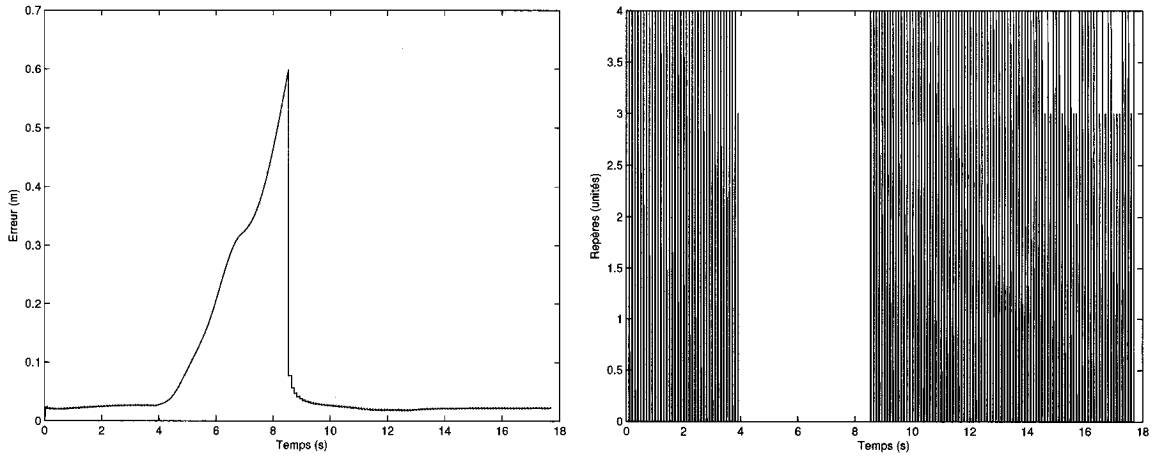

Figure 3.17: Erreur de position estimée et nombre de repères avec perte de la vision

La correction apportée lorsque les mesures de vision sont à nouveau disponibles est d'environ $0.15m$. On remarque toutefois que l'erreur estimée est alors d'environ $0.6m$. Cela démontre que les bruits de processus sont surestimés. Cette surestimation assure toutefois une correction rapide et ainsi, une plus grande robustesse par rapport aux glissements et collisions.

Des essais similaires ont permis de remarquer qu'il est généralement préférable de n'avoir aucune mesure de vision plutôt qu'une seule. Sans correction, l'erreur estimée augmente et, par conséquent, la correction qui sera apportée devient de plus en plus importante. Une correction importante basée sur une seule mesure n'assure pas nécessairement la convergence du filtre. Ce phénomème sera discuté plus en détail dans la section 3.5.4.

3.5 Problèmes rencontrés et limitations

Cette section traite des principaux problèmes rencontrés lors de la mise en oeuvre du système de localisation. Elle présente les conséquences de différents problèmes sur les performances du système, les méthodes utilisées pour contourner ces problèmes et les conclusions qui ont été tirées quant aux limitations du système.

3.5.1 Biais dans les mesures de vision

3.5.1.1 Biais causé par une rotation du montage

Le biais discuté dans cette section est l'angle de rotation entre la caméra et l'avant du robot tel qu'illusté par la figure 3.18. Ce biais est un angle constant qui s'additionne aux mesures effectuées par le système de vision. L'orientation calculée est alors celle du montage de vision et non celle du robot.

$$\alpha_{Biais} = \alpha_{Reel} + b$$

Lors de la première mise en oeuvre du filtre, croyant que les caméras étaient bien alignées avec l'avant du robot, le biais a été négligé. Cependant, les premiers essais ont confirmé la nécessité d'estimer le biais. La figure 3.19 présente deux trajectoires en ligne droite sans la correction et avec la correction du biais. On remarque dans les deux cas que les trajectoires basées sur l'odométrie sont orientées différemment étant donné l'erreur sur l'orientation initiale du robot et le biais. Le biais n'affecte pas la précision de la position mesurée par la vision, il fausse toutefois l'orientation mesurée. Lorsque le biais n'est pas corrigé, la prédiction ne s'effectue pas selon l'orientation réelle du robot, mais selon l'orientation de la caméra. Cela fait en sorte que la trajectoire estimée est décalée par rapport à la trajectoire réelle. Les corrections apportées par la vision réduisent ce décalage et rendent alors la trajectoire estimée en forme de dents de scie. L'estimation du biais permet de corriger l'orientation et ainsi faire coincider la trajectoire estimée avec la trajectoire réelle. On peut observer la transitoire de l'estimation du biais dans la courbe corrigée. Lorsque le robot se met en mouvement, le biais converge et la trajectoire se corrige.

Un biais qui n'est pas corrigé peut compromettre le fonctionnement du système de localisation lorsqu'aucune mesure de vision n'est disponible pendant un certain temps. La position prédite par l'odométrie est erronée et l'erreur associée à cette position est sous-estimée. Puisque la reconnaissance des objets est basée sur la position et l'erreur prédite, les objets ne sont pas cherchés aux bons endroits dans l'image et les zones de recherche sont trop étroites. Certaines balises peuvent ainsi ne plus être perçues et la position estimée du robot dérive alors davantage.

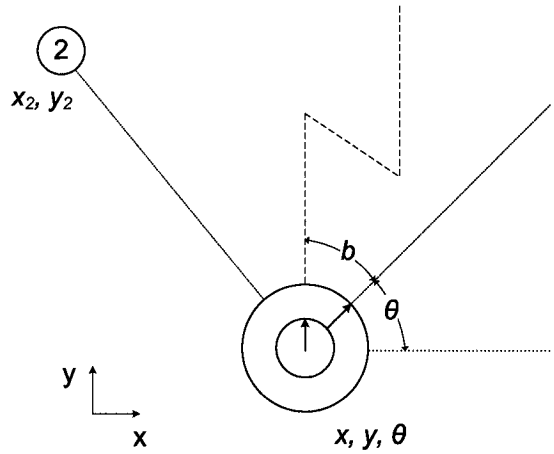

Figure 3.18: Écart entre l'orientation de la caméra et l'orientation réelle du robot

L'estimation du biais fonctionne parfaitement et elle évite l'étalonnage préalable de ce paramètre. De plus, peu importe le biais présent sur chacun des robots, le système de localisation est en mesure de l'estimer rapidement. De plus, cette estimation reste valide aussi longtemps que le système de localisation est en fonction.

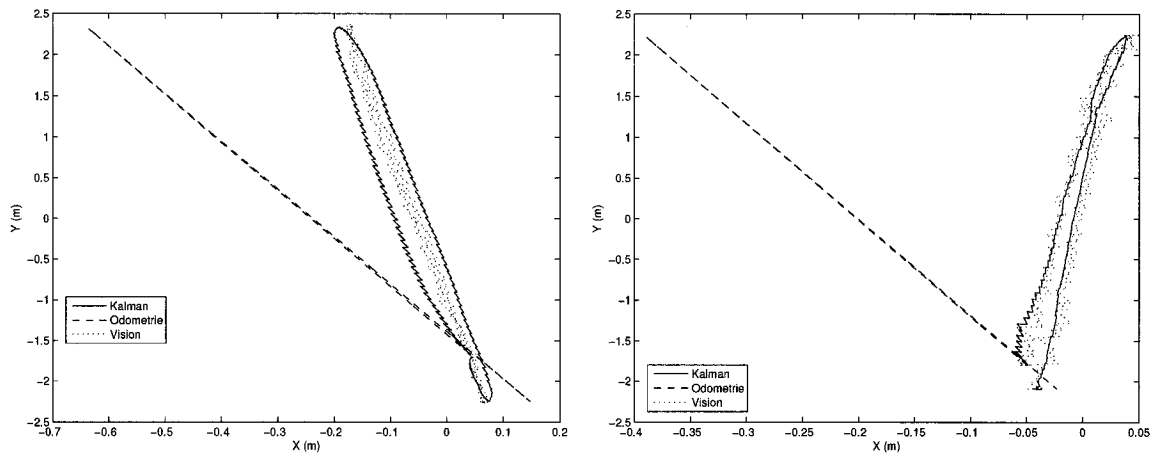

Figure 3.19: Ligne droite sans et avec correction du biais

3.5.1.2 Mauvais alignement et mauvais positionnement du montage

Les imprécisions dans l'alignement du miroir avec la caméra ainsi que celles dans l'alignement et le positionnement du montage sur le robot faussent les mesures re-

tournées par le système de vision. De plus, cet étalonnage est complexe et les alignements sont différents d'un robot à l'autre. Étant donné que le système de localisation n'utilise que les mesures d'angles horizontaux α, le paramètre le plus important à étalonner est la position du centre du miroir dans l'image. Lorsque cet étalonnage est incorrect et que les imprécisions sont importantes, les performances du système de localisation sont grandement dégradées. Le système de localisation considère, par hypothèse, que le bruit de mesure B_z sur une mesure de vision respecte une distribution normale de moyenne nulle et de covariance B_z. En considérant la fonction $h(\tilde{q})$ actuelle, l'écart entre les mesures estimées et les mesures réelles est fonction de ces imprécisions et la moyenne est variable et non-nulle.

On remarque bien l'effet de ces imprécisions sur les performances du système de localisation sur la figure 3.20. Cette figure présente les résultats d'une rotation sur place ($\Delta v = 0$) de deux robots différents: un robot avec un montage dont l'alignement est imprécis et un autre avec un alignement précis. La trajectoire de vision du robot avec le montage imprécis a la forme d'une ellipse et la position mesurée varie alors d'environ $\pm 0.1m$. La position estimée est donc imprécise et elle varie en fonction de l'orientation du robot. Ainsi, la position du robot semble se modifier lorsque le robot tourne légèrement. Toutefois, on remarque que la trajectoire mesurée du robot avec un montage bien aligné a la forme d'un nuage de points. La position estimée est alors conforme à nos attentes et elle est beaucoup plus précise à $\pm 0.02m$.

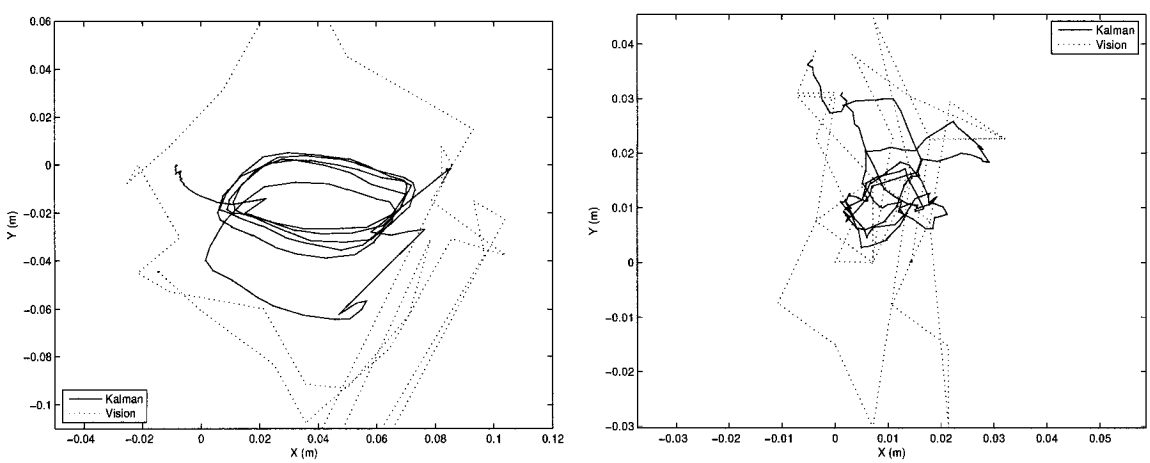

Figure 3.20: Rotation sur place avec un montage de vision mal et bien aligné

3.5.2 Retard dans l'acquisition de l'image

Par hypothèse, dans un filtre de *Kalman*, la correction apportée aux états prédits doit être parfaitement synchronisée avec la prédiction. Ainsi, une correction basée sur des mesures en retard ou en avance sur la prédiction fausse inévitablement l'estimation résultante. Dans le système de localisation, les mesures d'odométrie sont synchronisées avec les mesures de vision en récupérant dans un vecteur les mesures d'odométrie effectuées entre l'acquisition de l'ancienne image et la nouvelle. Toutefois, il a été remarqué qu'un délai non négligeable et variable est présent entre le moment où l'image est retournée à l'application par le pilote de la caméra et celui où l'acquisition réelle de l'image est effectuée par la caméra. Ce délai d'environ $0.2s$ est la somme du délai de traitement par la webcam, du délai de transfert à l'ordinateur, du délai de traitement par le pilote et la latence avant que l'image soit rendue disponible et que l'acquisition du temporisateur soit effectuée. On peut observer ce délai sur la figure 3.21 lorsque le robot a une accélération angulaire. On remarque que la trajectoire de vision présente un changement d'orientation en retard par rapport à l'odométrie. Cela a pour conséquence que la position estimée est en retard et donc moins précise lorsque le robot a des accélérations brusques et fréquentes.

Ce retard a des conséquences importantes et sa correction par une valeur constante n'a pas présentée d'améliorations très significatives ce qui porte à croire que le retard est variable. On remarque toutefois une nette amélioration lorsque la résolution de l'image est réduite et que la fréquence d'acquisition augmente à 30 images/s. Cependant, la reconnaissance des balises cylindriques et du ballon devient alors impossible lorsque ces elements sont situés à plus de $3m$ du robot, car leur taille dans l'image n'est plus que de quelques pixels. Il faudrait une référence temporelle sur le moment précis de l'acquisition de l'image par la caméra pour éviter ce délai.

3.5.3 Mesures erronées

Le système de vision, n'étant pas infaillible, retourne parfois des mesures erronées. Une mesure erronée correspond généralement à la mesure d'une balise partiellement obstruée ou à la mesure d'une mauvaise balise à la suite d'une confusion (voir 2.2.4). De plus, si une balise n'est pas située à l'endroit prévu, la mesure retournée ne con-

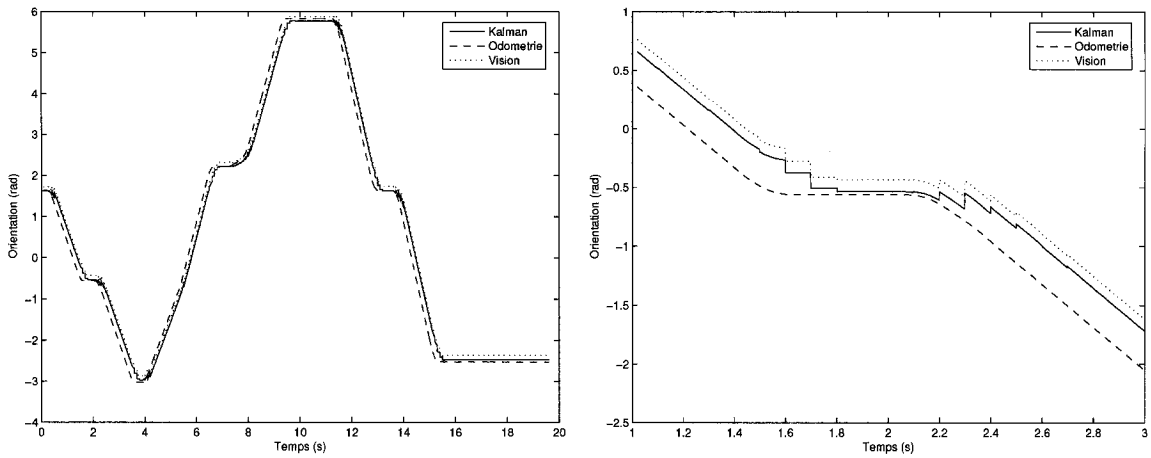

Figure 3.21: Orientation du robot avec retard dans les mesures de vision

corde pas avec la mesure estimée et les conséquences sont similaires à celles une mesure erronée. Les effets d'une mesure erronée sont inversement proportionnels au nombre de repères retournés. Étant donné la résolution par moindre carré, plus le nombre de repères retournés est grand, plus petit sera l'impact de la mesure erronée sur la position estimée. Ainsi, avec plusieurs balises, l'impact de quelques mesures erronées de temps à autre est négligeable.

Toutefois, lorsqu'une balise est mal positionnée, même si l'impact est diminué par les autres mesures, sa mesure biaisera constamment la position estimée. Il est donc important de s'assurer que la position des balises soit connue avec précision. Ainsi, il est utopique d'espérer une précision réelle de $\pm 0.02m$ sur la position estimée du robot si la position des balises est connue à $\pm 0.15m$.

3.5.4 Peu de mesures et mesures redondantes

Une ou deux mesures ne suffisent pas pour assurer la convergence d'un algorithme tel que celui décrit à la section 3.2.2. Le filtre de *Kalman* du système de localisation dispose d'informations supplémentaires par rapport à cet algorithme tel la position prédite et l'erreur qui lui est associée. Ceci lui permet d'apporter des corrections avec seulement une ou deux mesures. Cependant, il est nécessaire d'avoir plus de mesures dans certaines conditions. Lorsque les mesures de la vision sont disponibles après

un long moment et que la position estimée est fortement erronée, il peut être risqué d'apporter une correction avec seulement une ou deux mesures, car l'importante correction apportée n'est plus conforme au modèle non-linéaire du système. La correction apportée dans un filtre de *Kalman* étendu est basée sur la linéarisation du modèle non-linéaire autour de la position prédite. La correction est donc juste en autant que la position prédite soit suffisamment près de la position réelle et que la correction soit suffisamment petite pour que l'approximation du modèle non-linéaire par un modèle linéarisé reste valide. Généralement, des corrections fréquentes avec plusieurs mesures évitent ce genre de problèmes.

Il a été aussi remarqué que le filtre diverge lorsque le robot ne reçoit qu'une ou deux mesures provenant toujours des mêmes balises. À long terme, il manque d'informations pour faire converger à la fois la position et l'orientation du robot lorsque les corrections s'appliquent toujours de la même façon. Ainsi, le filtre peut très bien fonctionner s'il ne reçoit qu'une seule mesure à la fois en autant que la mesure provienne d'une balise différente à chaque fois. Cette conclusion est intéressante, car elle fait en sorte qu'une caméra omnidirectionnelle n'est pas indispensable pour la mise en oeuvre d'un filtre similaire. Plusieurs caméras unidirectionnelles orientées différemment pourraient être utilisées en autant que les mesures soient diversifiées et fréquentes.

3.5.5 Glissements et collisions

Le glissement des roues à la suite d'une accélération rapide, d'un virage prononcé ou d'une collision fausse grandement la position estimée. Lorsqu'il y a glissement, les mesures fournies par l'odométrie ne sont pas conformes au déplacement réel et la position prédite est erronée. Une erreur sur la position prédite peut être suffisante pour que le système de vision ne parvienne pas à identifier les balises et ainsi, compromettre la convergence du filtre. L'augmentation des bruits de processus liés à l'odométrie peuvent réduire les conséquences du glissement, mais ils diminuent la précision du filtre en régime permanent. Idéalement, les bruits de processus ou l'erreur sur la position prédite devraient être corrigés lorsqu'un glissement ou une collision est détecté. Ceci assurerait une convergence rapide en amenant le filtre à utiliser davantage les

mesures de vision pendant un court instant. Pour détecter les collisions ou les glissements, il faudrait utiliser des capteurs supplémentaires tel des accéléromètres. En pratique, le système est robuste et son bon fonctionnement n'est pas compromis pas les collisions. Toutefois, les systèmes de vision et de localisation sont basés sur une prédiction de la position du robot calculée à partir des mesures d'odométries. Les déplacements du robot doivent donc être mesurés par l'odométrie pour assurer le bon fonctionnement du système.

Chapitre 4

Perception d'objets dynamiques

4.1 Problématique

Pour interagir avec des objets présents dans son environnement, le robot doit être en mesure de les identifier et de trouver leur position. De la même façon, pour jouer au soccer, les robots doivent déterminer la position du ballon. Le système de perception a été conçu dans le but de permettre aux robots de participer à la compétition *Robocup* tout en leur permettant aussi de trouver des objets différents dans d'autres types d'applications.

4.1.1 Objet dynamique

Un **objet dynamique** est un objet de forme sphérique ou cylindrique dont la taille et la couleur contrastante sont connues, mais dont la position initiale est inconnue. Contrairement aux autres objets, il peut se mouvoir sur le sol et sa dynamique est connue. Par hypothèse, un objet dynamique se déplace librement à vitesse constante ou selon une accélération négative proportionnelle au coefficient de frottement visqueux de l'objet. De plus, la vitesse de l'objet peut être perturbée selon une accélération aléatoire de moyenne nulle et de variance connue. Un objet dynamique doit être différent des objets présents dans son environnement afin d'éviter toute confusion et permettre un recouvrement dans le cas où sa position est inconnue.

4.1.2 Performances et améliorations recherchées

Les objectifs de l'utilisation d'un filtre de *Kalman* pour estimer la position d'un objet dynamique sont les suivants:

- La réduction du bruit sur la position de l'objet en relation avec la prévision de sa dynamique.
- Une meilleure estimation de l'erreur sur la position de l'objet. L'erreur estimée doit être fonction de la dynamique de l'objet et de la précision de la correction apportée par les mesures. Une meilleure estimation de l'erreur permet une reconnaissance plus efficace par le système de vision.
- Une meilleure estimation de la vitesse de déplacement. Cette estimation permet de prédire la position de l'objet et la direction du déplacement.
- Une estimation du coefficient de frottement visqueux. La connaissance du coefficient de frottement visqueux permet une prédiction plus précise de la position et de la vitesse.

4.2 Solution par filtrage de Kalman

4.2.1 Modèle mathématique

La zone de pixels correspondant à un objet dynamique permet de déterminer deux mesures différentes. La mesure α est l'angle horizontal formé entre l'avant de la caméra et le centre de masse de l'objet. L'angle α est déterminé par la position du centre de masse des pixels de la zone. La mesure β est l'angle vertical formé entre l'horizon et le point le plus rapproché à la base de l'objet ou le point le plus rapproché à mi-hauteur. L'angle β est déterminé par la limite supérieure y_{max} de la zone de pixels ou par le centre de masse. Pour un objet de petite taille, le centre de masse permet une mesure d'une plus grande précision. Lorsque l'objet est grand, la mesure β varie moins (plus près de l'horizon) si le centre de masse est utilisé. En raison de la courbure du miroir et de la relation entre l'angle β et la distance relative à l'objet, il est alors préférable d'utiliser la base de l'objet afin de conserver β plus grand. L'angle

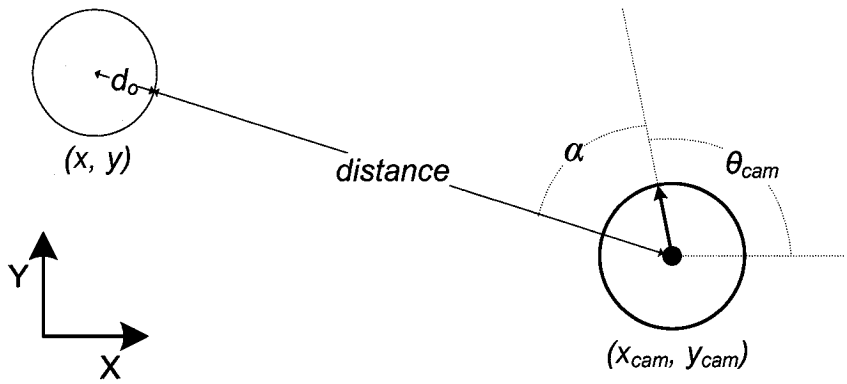

Figure 4.1: Objet dynamique et mesure α

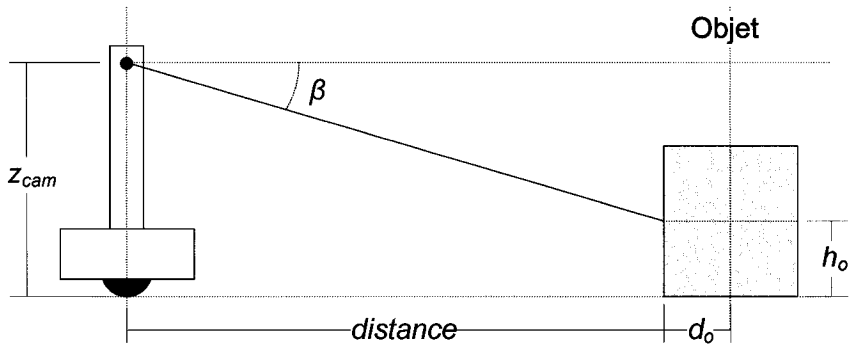

Figure 4.2: Objet dynamique et mesure β

β est défini par rapport à l'horizon et il est positif au-dessus de l'horizon (voir section 2.1.1). Les figures 4.1 et 4.2 illustrent les mesures α et β ainsi que les paramètres h_o et d_o.

Les symboles utilisés par le modèle mathématique du filtre de *Kalman* d'un objet dynamique sont décrits dans le tableau 4.1. Les paramètres h_o et d_o ont été ajoutés au modèle mathématique de l'objet dynamique afin de garder la solution générale et non spécifique à une forme d'objet en particulier. Les paramètres x_{cam}, y_{cam} et θ_{cam} correspondent à la position estimée de la caméra par le système de localisation. L'orientation θ_{cam} correspond à l'orientation de la caméra donc, l'orientation du robot avec le biais estimé par le système de localisation. Le paramètre z_{cam} correspond à

la hauteur du centre focal du miroir. Les bruits B_x, B_y et B_θ quant à eux, correspondent à la covariance de l'erreur estimée sur la position et l'orientation de la caméra.

Tableau 4.1: Constantes, états, entrées, bruits de processus et bruits de mesures

	Descriptions	***Unités***
d_o	Distance entre le point le plus rapporché et le centre de l'objet	m
h_o	Hauteur du centre de l'objet	m
x_{cam}	Estimation de la position en x de la caméra	m
y_{cam}	Estimation de la position en y de la caméra	m
θ_{cam}	Estimation de l'orientation de la caméra	rad
$\hat{x}$	Estimation de la position en x de l'objet dynamique	m
$\hat{y}$	Estimation de la position en y de l'objet dynamique	m
$\hat{\dot{x}}$	Estimation de la vitesse en x de l'objet dynamique	m/s
$\hat{\dot{y}}$	Estimation de la vitesse en y de l'objet dynamique	m/s
$\hat{\zeta}$	Estimation du coefficient de frottement visqueux	kg/s
Δt	Temps écoulé depuis la dernière mise à jour	s
B_{vx}	Bruit sur la vitesse en x (temps écoulé)	$\frac{m^2}{s}$
B_{vy}	Bruit sur la vitesse en y (temps écoulé)	$\frac{rad^2}{s}$
B_ζ	Bruit sur le coefficient de frottement visqueux (temps écoulé)	$\frac{kg^2}{s^2}$
B_α	Bruit sur une mesure α	rad^2
B_β	Bruit sur une mesure β	rad^2
B_x	Erreur sur la position de la caméra en x	m^2
B_y	Erreur sur la position de la caméra en y	m^2
B_θ	Erreur sur l'orientation de la caméra	rad^2

La position absolue d'un objet dynamique peut être calculée, à partir des mesures α et β et de la position de la caméra, de la manière suivante:

$$x = x_{cam} - \left(\frac{z_{cam} - h_o}{\tan\beta} + d_o\right)\cos(\theta_{cam} + \alpha) \tag{4.1}$$

$$y = y_{cam} - \left(\frac{z_{cam} - h_o}{\tan\beta} + d_o\right)\sin(\theta_{cam} + \alpha) \tag{4.2}$$

Soit:

$$\hat{q} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \dot{\hat{x}} \\ \dot{\hat{y}} \\ \hat{\zeta} \end{bmatrix} \quad u = \begin{bmatrix} \Delta t \end{bmatrix}$$

Les expressions du filtre de Kalman sont données par:

$$f\left(\hat{q}_{k-1}, u_{k-1}, 0\right) = \begin{bmatrix} \hat{x} + \dot{\hat{x}}\Delta t + \frac{\hat{\zeta}}{2}\dot{\hat{x}}\Delta t^2 \\ \hat{y} + \dot{\hat{y}}\Delta t + \frac{\hat{\zeta}}{2}\dot{\hat{y}}\Delta t^2 \\ \dot{\hat{x}} + \hat{\zeta}\dot{\hat{x}}\Delta t \\ \dot{\hat{y}} + \hat{\zeta}\dot{\hat{y}}\Delta t \\ \hat{\zeta} \end{bmatrix} \tag{4.3}$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t + \frac{\hat{\zeta}}{2}\Delta t^2 & 0 & \frac{1}{2}\dot{\hat{x}}\Delta t^2 \\ 0 & 1 & 0 & \Delta t + \frac{\hat{\zeta}}{2}\Delta t^2 & \frac{1}{2}\dot{\hat{y}}\Delta t^2 \\ 0 & 0 & 1 + \hat{\zeta}\Delta t & 0 & \dot{\hat{x}}\Delta t \\ 0 & 0 & 0 & 1 + \hat{\zeta}\Delta t & \dot{\hat{y}}\Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$W = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$Q = \begin{bmatrix} B_{vx}|\Delta t| & 0 & 0 \\ 0 & B_{vy}|\Delta t| & 0 \\ 0 & 0 & B_{\zeta}|\Delta t| \end{bmatrix} \tag{4.6}$$

$$h(\tilde{q}) = \begin{bmatrix} \arctan 2(\tilde{y} - y_{cam}, \tilde{x} - x_{cam}) - \theta_{cam} \\ \arctan 2\left(h_o - z_{cam}, \sqrt{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})} - d_o\right) \end{bmatrix} \tag{4.7}$$

$$H = \begin{bmatrix} \frac{\partial h_{[1]}}{\partial x} & \frac{\partial h_{[1]}}{\partial y} & 0 & 0 & 0 \\ \\ \frac{\partial h_{[2]}}{\partial x} & \frac{\partial h_{[2]}}{\partial y} & 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$V = \begin{bmatrix} 1 & 0 & \frac{\partial h_{[1]}}{\partial x_{cam}} & \frac{\partial h_{[1]}}{\partial y_{cam}} & 1 \\ \\ 0 & 1 & \frac{\partial h_{[2]}}{\partial x_{cam}} & \frac{\partial h_{[2]}}{\partial y_{cam}} & 0 \end{bmatrix} \tag{4.9}$$

$$D_{ro} = \sqrt{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})^2} \tag{4.10}$$

$$\frac{\partial h_{[1]}}{\partial x} = -\frac{\tilde{y} - y_{cam}}{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})^2} \tag{4.11}$$

$$\frac{\partial h_{[1]}}{\partial y} = \frac{\tilde{x} - x_{cam}}{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})^2} \tag{4.12}$$

$$\frac{\partial h_{[2]}}{\partial x} = \frac{(z_{cam} - h_o)(\tilde{x} - x_{cam})}{D_{ro}\left((z_{cam} - h_o)^2 + D_{ro}^2 + d_o^2 - 2d_o D_{ro}\right)} \tag{4.13}$$

$$\frac{\partial h_{[2]}}{\partial y} = \frac{(z_{cam} - h_o)(\tilde{y} - y_{cam})}{D_{ro}\left((z_{cam} - h_o)^2 + D_{ro}^2 + d_o^2 - 2d_o D_{ro}\right)} \tag{4.14}$$

$$\frac{\partial h_{[1]}}{\partial x_{cam}} = \frac{\tilde{y} - y_{cam}}{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})^2} \tag{4.15}$$

$$\frac{\partial h_{[1]}}{\partial y_{cam}} = -\frac{\tilde{x} - x_{cam}}{(\tilde{x} - x_{cam})^2 + (\tilde{y} - y_{cam})^2} \tag{4.16}$$

$$\frac{\partial h_{[2]}}{\partial x_{cam}} = -\frac{(z_{cam} - h_o)(\tilde{x} - x_{cam})}{D_{ro}\left((z_{cam} - h_o)^2 + D_{ro}^2 + d_o^2 - 2d_o D_{ro}\right)} \tag{4.17}$$

$$\frac{\partial h_{[2]}}{\partial y_{cam}} = -\frac{(z_{cam} - h_o)(\tilde{y} - y_{cam})}{D_{ro}\left((z_{cam} - h_o)^2 + D_{ro}^2 + d_o^2 - 2d_o D_{ro}\right)} \tag{4.18}$$

$$R = \begin{bmatrix} B_\alpha & 0 & 0 & 0 & 0 \\ 0 & B_\beta & 0 & 0 & 0 \\ 0 & 0 & B_x & 0 & 0 \\ 0 & 0 & 0 & B_y & 0 \\ 0 & 0 & 0 & 0 & B_\theta \end{bmatrix} \tag{4.19}$$

4.2.2 Mise en oeuvre et ajustement des paramètres

Le filtre de *Kalman* d'un objet dynamique a été mis en oeuvre en utilisant la même libraire que celle utilisée par le système de localisation. L'estimation de la position d'un objet est basée sur les mesures transmises par le système de vision et elle est donc synchronisée avec l'acquisition de la dernière image. La prédiction de la nouvelle position d'un objet est fonction de la vitesse estimée, du coefficient de frottement visqueux et du temps écoulé depuis la dernière mise à jour. Puisque la position courante de l'objet est inconnue, elle peut être prédite de la même façon, à partir de la dernière position estimée.

Puisque la position d'un objet est mesurée à partir de la position de la caméra, le filtre doit connaître la position estimée de la caméra et les erreurs qui sont estimées. Cette mise à jour est faite avant chaque itération à partir des dernières estimations fournies par le système de localisation. Les erreurs estimées sur la position de la caméra sont considérées comme étant des bruits altérant la qualité des mesures α et β. Ainsi, une erreur de $\pm 0.5m$ a une importance beaucoup plus grande sur la qualité des mesures si l'objet est près du robot que s'il est loin de ce dernier. Toutefois, l'erreur estimée sur la position d'un objet n'est pas nécessairement plus grande que celle sur la position du robot, car l'estimation de la position de l'objet tient compte de la prévision de sa dynamique. Un objet immobile ou presque (accélérations prévues très faibles) peut avoir une erreur estimée plus faible que celle du robot une fois que sa position a convergé. Par contre, lorsque les mesures de vision sont biaisées, la position de la caméra est aussi biaisée. Par conséquent, la position de l'objet est doublement biaisée. Il est alors préférable que les bruits de processus soient suffisamment élevés pour garder le filtre actif et ainsi éviter que la position de l'objet converge sur une position biaisée.

Un filtre de *Kalman* est associé à chacun des objets dynamiques observés. De plus, chaque objet a ses propres paramètres, soit son état initial (la position, la vitesse et le coefficient de frottement initial), l'erreur associée à l'état initial, ses bruits de processus (l'estimation de sa dynamique) et ses bruits de mesures. Bien que les bruits sur les mesures fournies par le système de vision soient constants, un objet peut les

augmenter si la mesure est jugée trop biaisée et qu'elle peut nuire à l'estimation de la position de l'objet. Les bruits B_{vx} et B_{vy} sur la dynamique sont décrits comme étant la covariance de la variation de la vitesse de l'objet pendant une seconde. Par exemple, si l'on considère que la vitesse de l'objet peut varier d'environ $\pm 0.5 m/s$ à chaque seconde, les bruits B_{vx} et B_{vy} sont égaux à 0.25. Si un objet est immobile et que sa position est inconnue, l'erreur initiale sur les vitesses $\dot{x}$ et $\dot{y}$ doit être nulle et les B_{vx} et B_{vy} doivent être très faibles (1^{-6}). Ainsi, le filtre estime seulement la position du robot et conserve une vitesse estimée nulle. Un exemple sera présenté dans la section 4.3.

4.3 Résultats

4.3.1 Simulation

Les figures 4.3, 4.4 et 4.5 présentent les résultats d'une trajectoire simulée d'un objet. L'objet se déplace en ligne droite, sa vitesse initiale est de $1.5m$ et son coefficient de frottement visqueux est de $-0.15 kg/s$. La position, la vitesse et le coefficient de frottement initial sont inexacts pour présenter la transitoire des états estimés par le filtre de *Kalman*. Les bruits de processus sur la vitesse en x et y sont nuls. On remarque que l'erreur sur la position estimée est plus importante que l'erreur sur la position déterminée à partir des mesures au début de l'essai. Toutefois, la position converge rapidement et l'erreur sur la position estimée devient inférieure à l'erreur sur la position mesurée. De la même façon, une fois la vitesse estimée, l'erreur est nettement inférieure à l'erreur sur la vitesse déterminée à partir de la dérivé de la position mesurée.

L'erreur estimée est supérieure et de forme similaire à l'erreur réelle. On remarque que l'erreur augmente lorsque l'objet s'éloigne du robot puisque la correction apportée par la mesure β est de plus en plus imprécise. L'erreur estimée, proportionnelle à la position de l'objet, est plus réaliste et elle permet une meilleure estimation de la taille de la zone de recherche du système de vision.

Le coefficient de frottement visqueux est estimé après environ $10s$ et l'erreur réelle sur le coefficient diminue plus rapidement que l'erreur estimée. En pratique, cette estimation fonctionne moins bien puisque les trajectoires ne sont généralement pas conformes au modèle établi. Cette simulation présente le fonctionnement du filtre dans les meilleures conditions puisqu'aucune accélération ne perturbe la trajectoire de l'objet.

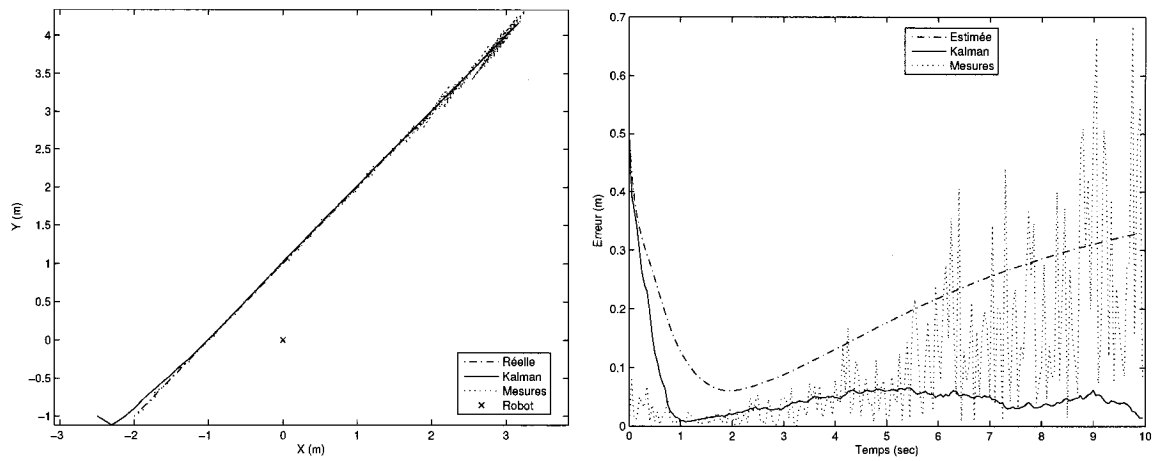

Figure 4.3: Trajectoire et erreur de position d'un objet dynamique

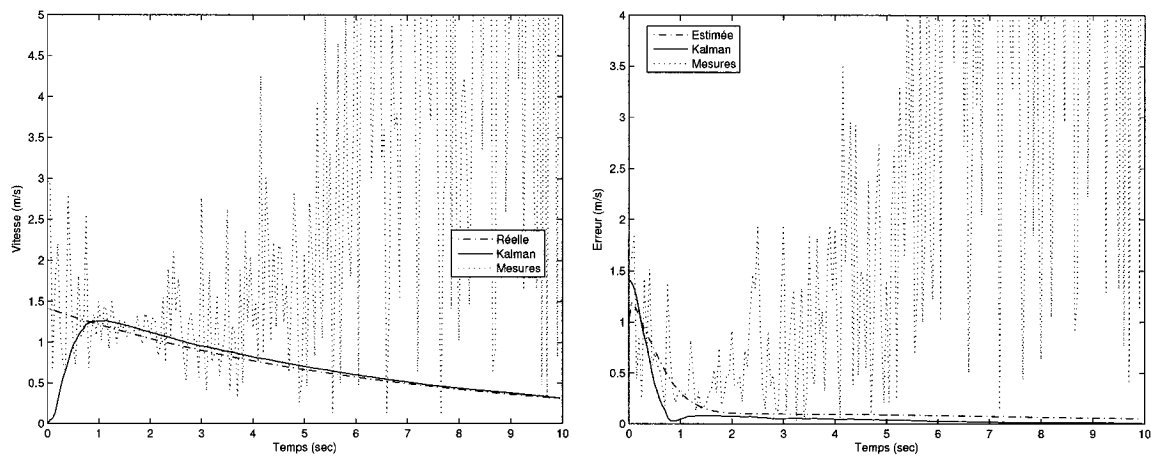

Figure 4.4: Vitesse et erreur sur la vitesse d'un objet dynamique

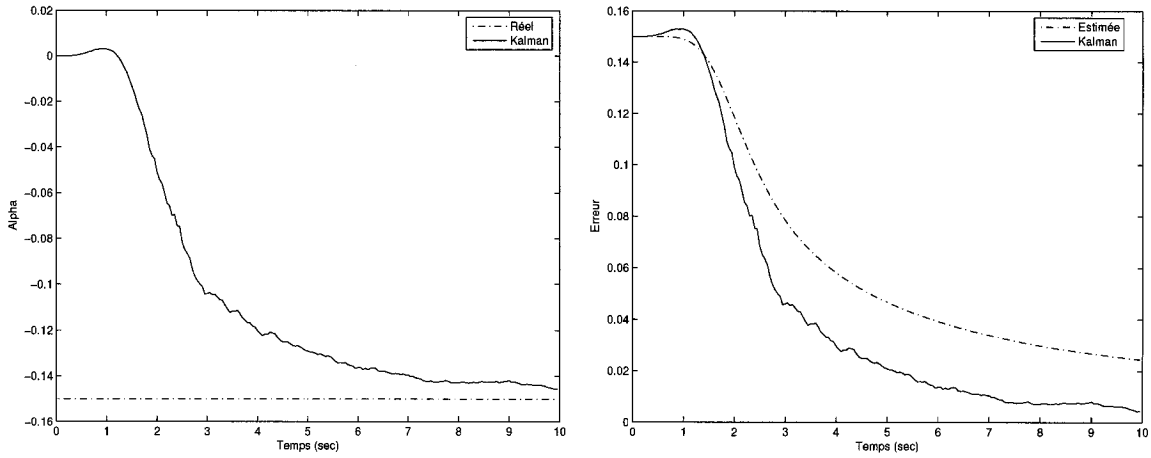

Figure 4.5: Estimation du coefficient de frottement visqueux d'un objet dynamique

4.3.2 Expérimentation

Les figures 4.6 et 4.7 présentent les résultats expérimentaux de la trajectoire linéaire d'un objet dynamique. L'objet est poussé rapidement au départ pour ensuite ralentir tranquilement. Bien entendu, l'accélération de départ n'est pas une accélération aléatoire suivant une distribution normale, mais elle représente une accélération plus typique des accélérations rencontrées dans les situations appliquées. Le filtre prévoit une variation d'accélération de $\pm 0.5 m/s^2$. L'amélioration sur l'estimation de la position de l'objet est difficile à remarquer sur la courbe de la trajectoire. Toutefois, l'estimation de la vitesse est nettement supérieure. Puisque l'accélération aléatoire prévue est inférieure à l'accélération réelle, on remarque un retard dans l'estimation de la vitesse de l'objet. Ce retard illustre les inconvénients d'une sous-estimation de l'accélération de l'objet. Les courbes de l'estimation des erreurs de position et de vitesse montrent que l'erreur sur la position de l'objet décroit lorsque l'objet s'approche du robot. De plus, il a été observé que ces estimations réalistes concordent avec la précision de la position réelle de l'objet.

La figure 4.8 présente la situation particulière d'un objet dynamique immobile de position inconnue. La vitesse initiale et l'accélération aléatoire de l'objet sont nulles. De plus, puisque la mesure β est beaucoup plus biaisée que la mesure α, le bruit prévu sur la mesure a été considérablement augmenté afin d'invalider la mesure. Le

filtre tente donc d'estimer la position de l'objet avec la mesure α seulement. Ainsi, lorsque le robot se déplace, la mesure α change et la position de l'objet peut être estimée avec une plus grande précision. On remarque sur la figure que la trajectoire mesurée de l'objet a la forme d'un cercle, car la mesure β est biaisée en relation avec la position du robot. Toutefois, en invalidant la mesure biaisée β, l'estimation de la position converge à $\pm 2cm$.

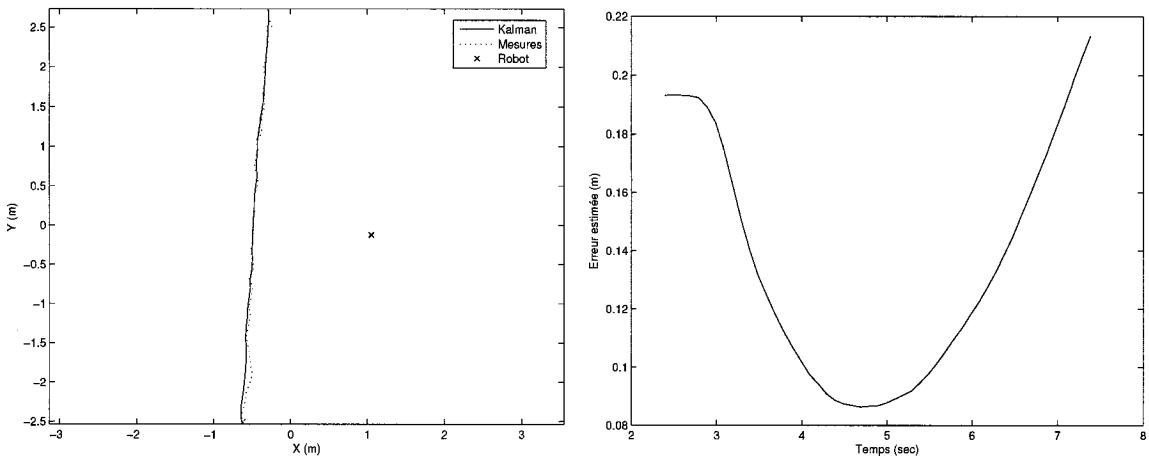

Figure 4.6: Trajectoire expérimentale d'un objet dynamique

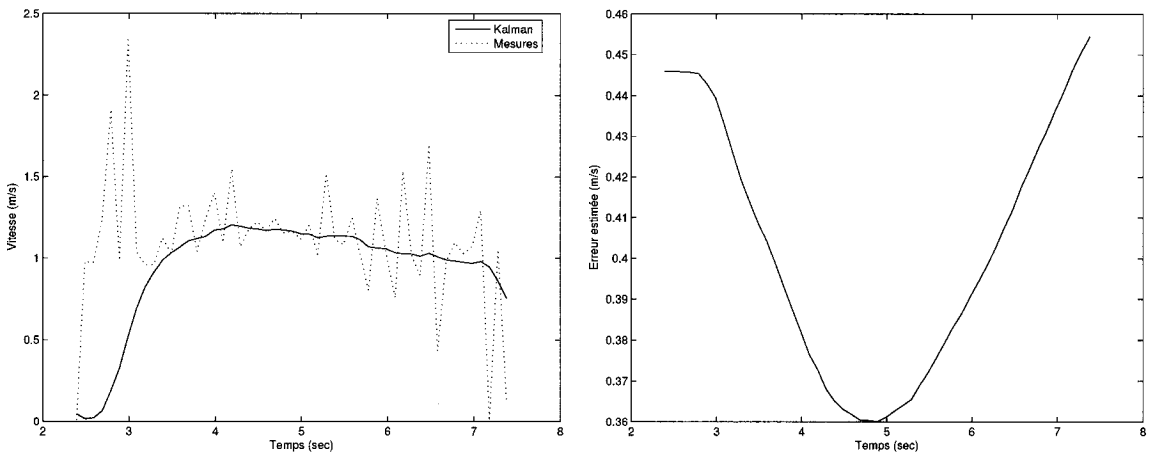

Figure 4.7: Vitesse expérimentale d'un objet dynamique

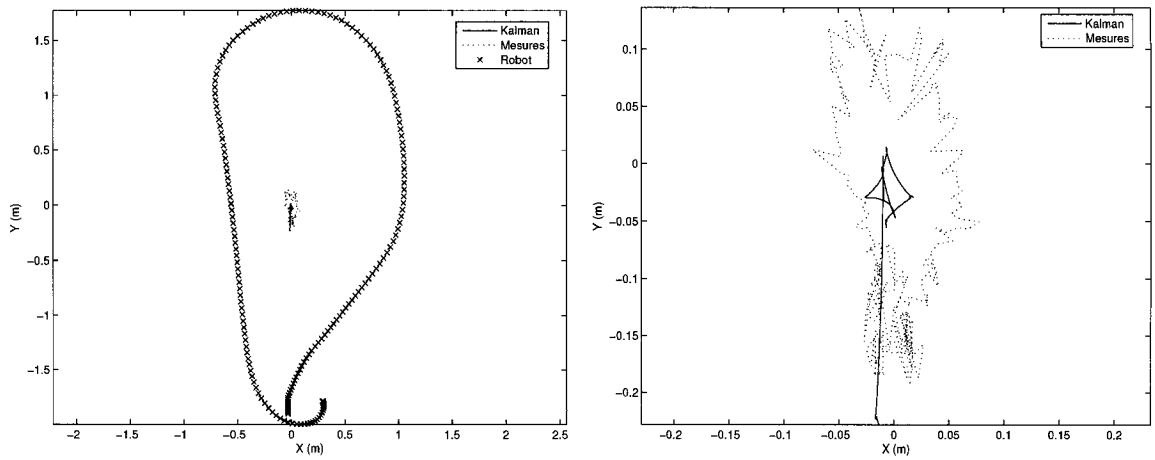

Figure 4.8: Trajectoire expérimentale d'un objet dynamique immobile

4.4 Problèmes et limitations

4.4.1 Connaissance de l'accélération de l'objet

Lorsque l'accélération aléatoire de l'objet est sous-estimée, la vitesse et la position estimées réagissent en retard à une accélération brusque. De plus, lorsque l'objet se déplacement librement ou qu'il est arrêté, une surestimation de l'accélération aléatoire rend la position estimée moins précise et plus bruitée. L'ajustement de l'accélération prévue impose de faire un compromis entre la vitesse de réaction aux accélérations brusques et la précision de l'estimation. Les figures 4.9 et 4.10 présentent la position et la vitesse estimées pour une même trajectoire simulée et différentes accélérations prévues.

On remarque que le filtre réagit plus rapidement pour une accélération prévue de $\pm 2m/s^2$, mais la position et la vitesse sont beaucoup plus bruitées par la suite. De plus, l'erreur estimée est nettement supérieure à l'erreur réelle sur la position de l'objet lorsque celui-ci se déplace librement. Pour une accélération prévue de $\pm 0.25m/s^2$, la position et la vitesse estimées sont beaucoup moins bruitées lorsque l'objet se déplace librement, mais le filtre réagit en retard à l'accélération. Les courbes idéales présentent la position et la vitesse estimées par le filtre lorsque la prévision de l'accélération est conforme à l'accélération réelle. L'erreur est alors réduite et la

réaction est plus rapide. Les performances du filtre sont donc nettement améliorées si la prévision de l'accélération est adaptée en temps-réel à l'accélération de l'objet. Dans le cas des robots joueurs de soccer, l'accélération du ballon devait être augmentée seulement lorsqu'un autre joueur est à proximité.

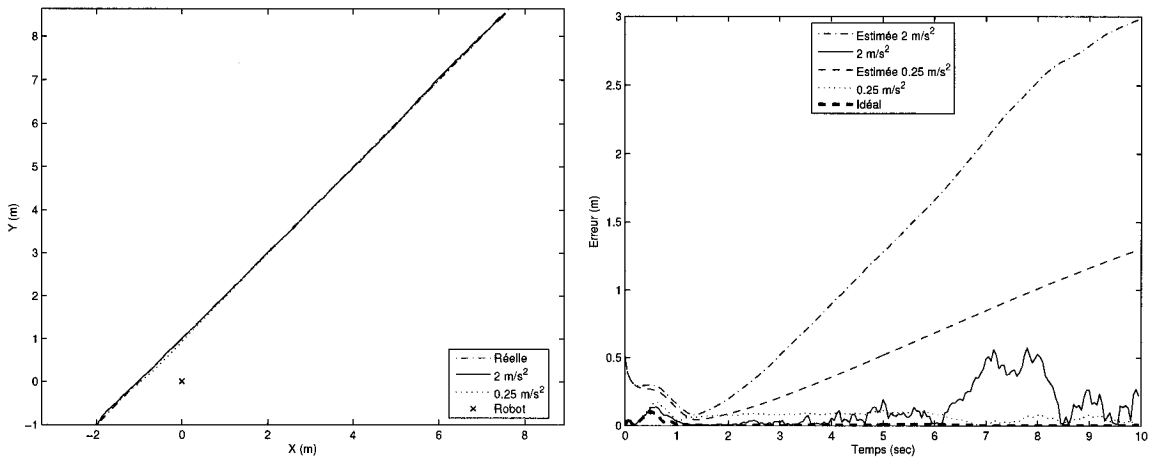

Figure 4.9: Estimation de la position pour différentes accélérations prévues

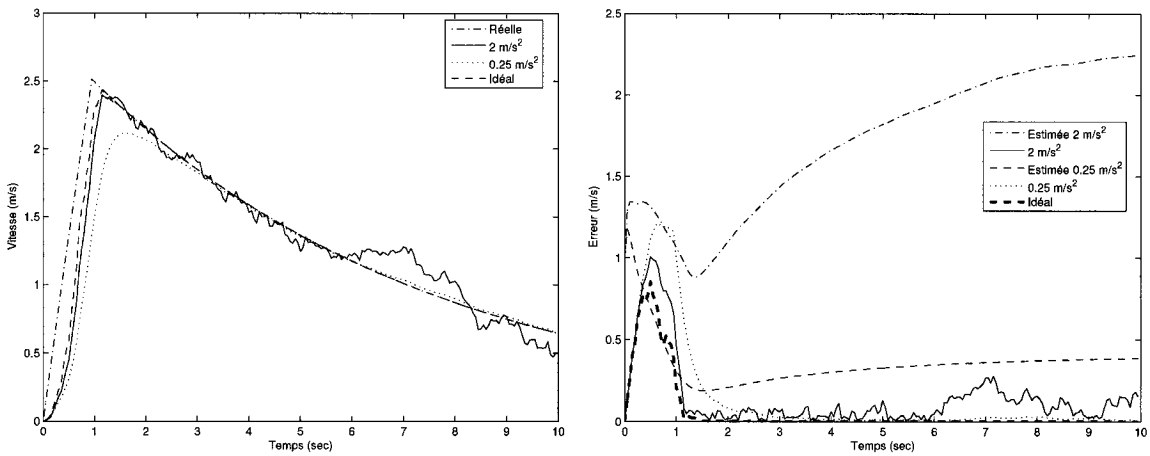

Figure 4.10: Estimation de la vitesse pour différentes accélérations prévues

4.4.2 Mauvais alignement et étalonnage du montage

Puisque la position d'un objet dynamique est déterminée seulement à partir des mesures du système de vision, les imperfections du montage de la caméra et du miroir

ont d'importantes répercussions sur la qualité de l'estimation. Malheureusement, les méthodes d'étallonnage du montage n'ont jamais permis d'obtenir des mesures β non biaisées. La distance relative d'un objet par rapport du robot est donc biaisée proportionnellement au biais de la mesure β.

Trois causes ont été identifiées:

- La hauteur du centre focal du miroir (de la caméra) par rapport au sol est erronée.
- La relation non-linéaire entre la hauteur d'un pixel et l'angle β correspondant est estimée par une droite et ne concorde donc pas parfaitement.
- Les imprécisions dans l'alignement de la caméra avec le miroir, et dans l'alignement du montage sur le robot, font en sorte que l'image panoramique est légèrement déformée. Par conséquent, la relation non-linéaire entre la hauteur d'un pixel et l'angle β correspondant varie aussi en fonction de l'angle α.

La première cause rend l'estimation de la distance constamment plus courte ou plus grande que la distance réelle de l'objet. La position est biaisée mais elle reste constante pour une distance donnée. Les deux autres causes font en sorte que le biais sur la distance change en fonction de la distance réelle et de l'angle α par rapport à l'objet. Par conséquent, la distance mesurée varie lorsque le robot tourne sur lui-même. On remarque aussi les effets des biais sur la figure 4.8 où la position mesurée de l'objet varie en fonction de la position du robot alors qu'en réalité l'objet est immobile. Toutefois, la relation entre l'angle β et la distance de l'objet fait en sorte que les biais ont peu d'influence sur la position de l'objet lorsque celui-ci est près du robot. Par contre, une trajectoire rectiligne est perçue légèrement courbe par le robot à cause des biais, ce qui fausse l'estimation de la vitesse de l'objet et rend l'estimation du coefficient de frottement difficile. De plus, la position biaisée d'un objet aura des répercussions sur le partage d'information avec les autres robots décrits dans la section 5.

4.4.3 Retard dans l'acquisition de l'image

La section 3.5.2 énonce les conséquences du retard dans l'acquisition de l'image sur la position estimée. Ce retard, attribuable au délai entre l'acquisition réelle de l'image et le moment où l'image est retournée à l'application, a une double conséquence sur l'estimation de la position d'un objet dynamique. Puisque la position du robot devient erronée lorsque celui-ci accélère, la position de l'objet dynamique le devient aussi car elle est calculée à partir de la position du robot. De plus, cette erreur sur la position estimée fait en sorte que l'objet semble se déplacer alors que ce n'est pas le cas. Ce déplacement engendre l'estimation d'une vitesse et la trajectoire estimée du robot n'est pas conforme à la trajectoire réelle.

Lorsque le robot est à l'arrêt et que l'objet se déplace librement, les références temporelles associées à chacune des positions estimées sont aussi en retard. Ainsi, si la position courante de l'objet est prédite en fonction du temps écoulé depuis la dernière mise à jour, la position prédite sera aussi en retard. De plus, pour fournir une position courante exacte de l'objet, le délai entre l'acquisition de l'image et l'estimation de la position après le traitement de l'image doit être très court. Dans le cas contraire, la position estimée est en retard et une prédiction s'impose pour connaître la position courante de l'objet. Cette prédiction ajoute une imprécision proportionnelle au temps écoulé. Ce problème ne se pose pas avec le système de localisation puisque la prédicition est alors basée sur les mesures d'odométrie et non pas seulement sur la vitesse estimée. Ainsi, le délai entre l'acquisition réelle de l'image et le moment où l'image est retournée à l'application dégrade les performances du système, mais le temps de traitement les dégrade aussi. Cette constatation impose de faire un compromis dans le choix de la méthode de reconnaissance de l'objet et le temps de traitement résultant.

4.4.4 Recouvrement

Un objet dynamique est recherché par le système de vision en fonction de sa position et de l'erreur prédite. Ainsi, plus l'erreur de position est élevée, plus la zone de recherche est grande. Toutefois, lorsque la différence entre la position prédite et la position réelle de l'objet est trop grande pour que l'objet soit dans la zone de

recherche, un autre algorithme doit être utilisé pour retrouver la position de l'objet. Le système de vision procède au recouvrement d'un objet dans la situation particulière où l'objet est considéré perdu. Pour qu'un objet soit perdu, l'erreur estimée sur sa position doit être supérieure à l'erreur limite fixée et le temps écoulé depuis le moment où l'objet a été perçu pour la dernière fois doit être supérieur au délai limite fixé.

Lors d'un recouvrement, toute la zone de l'image correspondant aux objets situés à une distance donnée du robot est traitée. Ensuite, toutes les zones de la couleur de l'objet recherché sont répertoriées et la taille de chacune est vérifiée en fonction de la distance évaluée. Si plusieurs zones sont valides, celle située le plus près du robot est choisie. Ce traitement exigeant n'est pas infaillible et l'objet recherché peut être confondu avec d'autres objets présents dans l'environnement. Pour cette raison, il est préférable qu'un objet dynamique soit différent de tous les autres objets et qu'il reste toujours dans le champ de vision du robot.

4.4.5 Grandes corrections

Lorsque la différence entre la position prédite et la position réelle d'un objet est grande, la correction apportée par le filtre est incorrecte. La relation entre les mesures et la position d'un objet est fortement non-linéaire et les corrections sont appliquées selon un modèle linéarisé autour de la position prédite. Lorsque l'erreur est grande, l'importante correction apportée excède l'intervalle de validité du modèle linéarisé. La position est alors mal corrigée et l'erreur estimée devient largement inférieure à l'erreur réelle. Ces deux effets combinés peuvent faire en sorte que l'objet ne soit pas perçu par le système de vision.

Pour contourner le problème, si la somme des écarts entre les mesures estimées et les mesures effectuées est supérieure à 0.5, la correction n'est pas appliquée et la position est estimée directement à partir des mesures sans considérer la position prédite. Cette procédure particulière empêche le filtre de diverger et favorise une estimation plus rapide de la nouvelle position. Il existe toutefois d'autres filtres de *Kalman* plus évolués qui tiennent compte des non-linéarités du système, mais leur utilisation est complexe et le temps de traitement est plus long. La solution utilisée est simple et

les résultats sont forts acceptables. De plus, cette situation particulière survient pas lorsque l'objet est perçu régulièrement.

Chapitre 5

Partage d'informations

5.1 Problématique

Lorsque plusieurs robots sont présents dans le même environnement, leur coopération présente plusieurs avantages. Outre le partage des tâches à effectuer, les robots peuvent coopérer en partageant leur perception respective de l'environnement. Ce partage d'informations permet à chacun des robots de connaître la position de tous les robots en plus de la position des objets dynamiques perçus par chacun d'eux. En partageant l'information sur leur position respective, les robots évitent de mesurer par eux-même la position de chacun des robots. De plus, le partage de la position d'un objet dynamique permet à un robot d'améliorer son estimation de la position de l'objet. La position d'un objet est alors estimée à partir des mesures effectuées par le système de vision et des mesures reçues des autres robots. La position de l'objet peut ainsi être estimée plus précisément. Si l'objet n'est pas perçu par le système de vision, la position est alors estimée par les mesures reçues des autres robots. Le robot peut alors fonctionner comme si l'objet avait été perçu.

L'utilisation d'un lien de communication *Ethernet* sans-fil impose un délai de transmission non-négligeable lors du partage d'informations. L'information reçue est alors toujours en retard. De plus, pour combiner deux mesures différentes et réduire ainsi l'incertitude associée à l'estimation, ces mesures doivent être parfaitement synchronisées. La technologie utilisée ne permet pas de synchroniser l'acquisition des

images de tous les robots. Les mesures asynchrones doivent alors être synchronisées en appliquant une prédiction sur les mesures moins récentes pour permettre une mise en commun efficace. L'utilisation de mesures et d'informations asychrones réduit considérablement les avantages de la mise en commun de l'information. Le robot réagit alors maladroitement à un environnement grandement dynamique puisque sa perception est basée sur des informations en retard. Pour éviter ce retard et garder une bonne précision, il est alors préférable pour un robot d'utiliser ses propres mesures. Il n'y a donc plus de mise en commun et l'information partagée n'est considérée que si aucune mesure n'est disponible.

Les figures 5.1, 5.2 et 5.3 illustrent par un exemple la problématique. La figure 5.1 présente la situation idéale où les mesures des robots A et B sont effectuées en même temps. Le robot A mesure la position du ballon avec une précision plus grande que le robot B puisqu'il est plus près du ballon. La précision des mesures est illutrée par une ellipse autour du ballon. Le robot B peut améliorer la précision de sa mesure en combinant la mesure du robot A à la sienne. En pratique, lorsque le ballon est en mouvement, les délais de traitement et de transmission font en sorte que le robot B reçoit la mesure du robot A en retard. La figure 5.2 illustre la trajectoire du ballon et le retard entre la mesure effectuée par le robot B et la mesure reçue du robot A. En combinant ces deux mesures, la position estimée est alors plus près de la position mesurée par robot A étant donné sa plus grande précision. Toutefois, cette mesure est en retard et elle ne correspond plus à la position réelle du ballon. La combinaison des deux mesures ne présente alors aucun avantage. Pour rendre avantageuse la mise en commun des mesures des deux robots, il faut les synchroniser. Ainsi, une prédiction est effectuée en fonction du temps écoulé sur la position du ballon mesurée par le robot A afin de la synchroniser avec la mesure effectuée par le robot B. La précision de la position mesurée par le robot A sera alors réduite étant donné les incertitudes associées à la prédiction effectuée. Toutefois, la mesure du robot A n'est plus en retard et elle peut être mise en commun avec la mesure du robot B tel qu'illustré à la figure 5.3. La précision de la position du ballon après la mise en commun ne sera pas aussi grande que la précision de la position déterminée dans le cas idéal, mais elle sera plus précise que celle mesurée par le robot B.

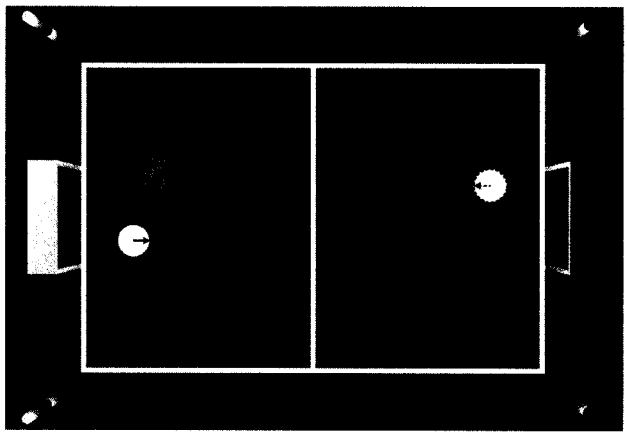

Figure 5.1: Problématique - Situation idéale

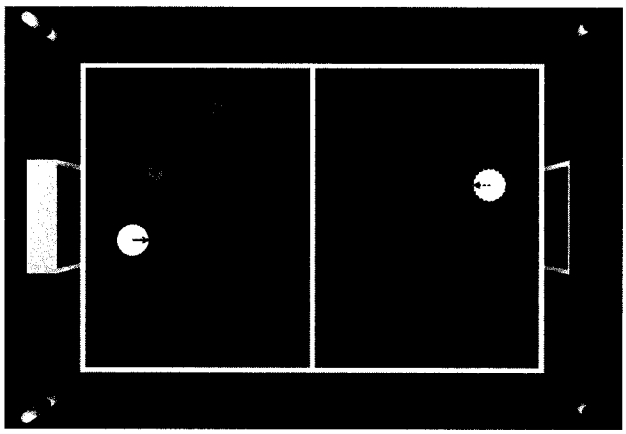

Figure 5.2: Problématique - Retard

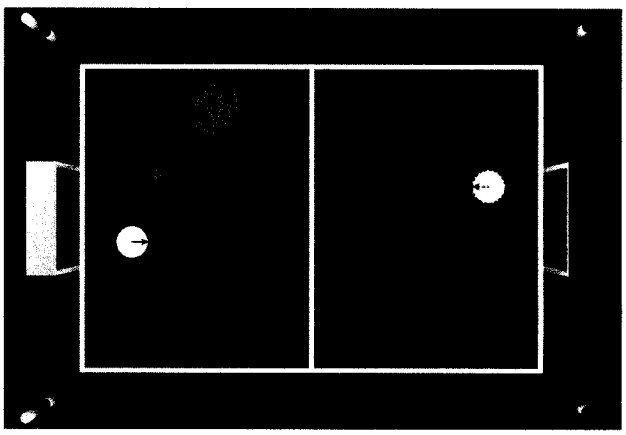

Figure 5.3: Problématique - Synchronisation

Cette section propose donc une méthode pour partager efficacement la perception des robots même si celle-ci est asynchrone et qu'un délai de transmission fait en sorte que l'information partagée est en retard. Les documents [23] et [24] présentent la problématique du partage d'informations via un réseau de type Internet.

5.1.1 Performances et améliorations recherchées

Les objectifs du partage d'informations et de l'utilisation d'un filtre de *Kalman* pour mettre en commun plusieurs informations redondantes sont les suivants:

- La synchronisation d'informations asynchrones à l'aide d'une prédiction basée sur le modèle dynamique de l'objet. Cette synchronisation permet de limiter le retard engendré par le délai de transmission entre les robots. Afin d'évaluer la précision de l'information prédite, les erreurs associées à l'information sont prédites aussi.

- La mise en commun d'informations relatives à un même objet. Cette mise en commun permet d'améliorer la précision de la perception d'un seul robot en bénéficiant de la perception de tous les robots. Ainsi, la précision de la perception de chacun des robots après le partage et la mise en commun devient similaire à celle du robot ayant la meilleure précision avant le partage. L'état d'un objet peut être estimé à partir de l'information reçue même s'il n'est pas mesuré par le système de vision. Il suffit qu'un seul robot mesure la position d'un objet pour que tous les autres robots en connaissent aussi la position.

- Lors d'une mise en commun, l'apport d'une information reçue est pondéré en fonction de sa précision estimée par rapport à celle des autres informations après la synchronisation.

5.2 Architecture client-serveur

Étant donné que la vitesse de transmission des liens de communication est limitée, une architecture client-serveur est proposée pour réduire la bande-passante nécessaire pour le partage d'informations. Deux types différents d'informations sont utilisés

pour partager la position d'un robot et la position d'un objet dynamique. Dans les deux cas, l'information partagée contient l'état d'un objet ou d'un robot (position et vitesse), une référence temporelle, la matrice de covariance des erreurs associées à l'état ainsi que les bruits de processus à utiliser pour effectuer une prédiction. Cette prédiction est nécessaire pour prédire l'état future et l'erreur associée, pour synchroniser les informations redondantes et, ainsi, permettre une estimation optimale lors de la mise en commun de ces informations.

Lors de la connection d'un robot au serveur, le minuteur du robot est synchronisé avec celui du serveur. Ainsi, les références temporelles associées à chacune des informations transmises sont synchronisées par rapport au même minuteur. Une synchronisation précise est requise pour que le système de partage d'informations fonctionne correctement.

Les figures 5.4 et 5.5 présentent un exemple typique de partage de la position des robots et de la position du ballon. Le robot transmet la dernière mise à jour de sa position et celle du ballon. Il reçoit ensuite la position des deux autres robots et la position du ballon résultant de la mise en commun des positions mesurées par ceux-ci. Le robot pourra ensuite réutiliser les positions des robots et combiner la position du ballon avec sa position mesurée. De plus, si aucune position n'a été mesurée pour le ballon, la position reçue sera utilisée pour mettre à jour la position estimée.

Des filtres de *Kalman* sont utilisés pour réaliser la mise en commun des informations. Ces filtres seront présentés dans la section 5.3. Pour combiner efficacement plusieurs informations relatives à un même objet, il faut s'assurer que ces informations correspondent à l'état de cet objet au même moment. Si ce n'est pas le cas, les informations sont synchronisées avec l'information la plus récente. Pour ce faire, on effectue une prédiction sur chacune des informations en fonction de l'écart entre la référence temporelle de l'information et celle de l'information la plus récente. Cette prédiction est effectuée sur l'état et la matrice des erreurs associées à l'état de la même façon que l'étape de prédiction décrite à la section 3.3.2. Les bruits de processus transmis avec l'état sont utilisés pour effectuer la prédiction. Les erreurs

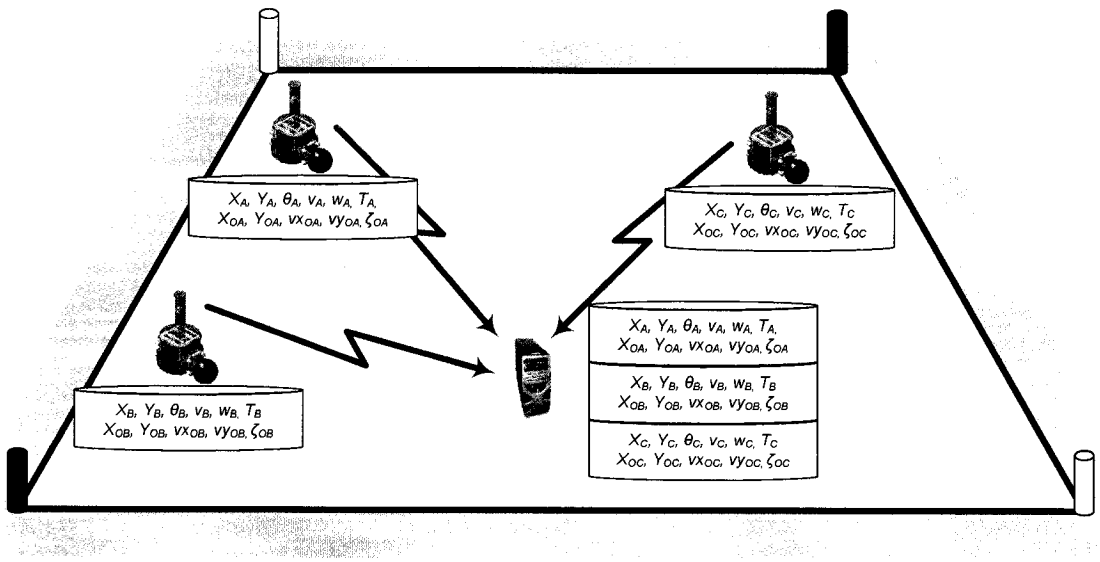

Figure 5.4: Partage d'informations - Exemple de transfert au serveur

associées à l'état augmentent en fonction de l'intervalle de temps de la prédiction. Ainsi, une information moins récente sera peu considérée par rapport à une information récente ayant des erreurs associées aussi élevées. Lors de la mise en commun, le filtre est initialisé avec l'information la plus récente. Les autres informations sont d'abord prédites pour ensuite être considérées comme des mesures. Les erreurs qui leurs sont associées sont alors considérées comme des bruits de mesures. La correction apportée est ainsi pondérée en fonction des erreurs associées à chacune des informations. L'état résultant de la mise en commun est l'état qui minimise la covariance des erreurs estimées et il est plus précis que le plus précis des états utilisés.

Les tableaux 5.1 et 5.2 présentent les informations partagées pour les types robot et objet dynamique. Les vitesses font partie des états partagés, car elles permettent d'effectuer la prédiction de la position lorsque l'objet est en mouvement. Les matrices E sont les matrices de covariance des erreurs estimées pour les états. Par définition, ces matrices sont symétriques et seul la diagonale et la partie supérieure sont transférées. Pour que la mise en commun effectuée par le filtre de *Kalman* soit optimale, les informations transmises doivent être indépendantes. Ainsi, l'état et les erreurs déterminés à partir des mesures doivent être transmises au serveur plutôt que

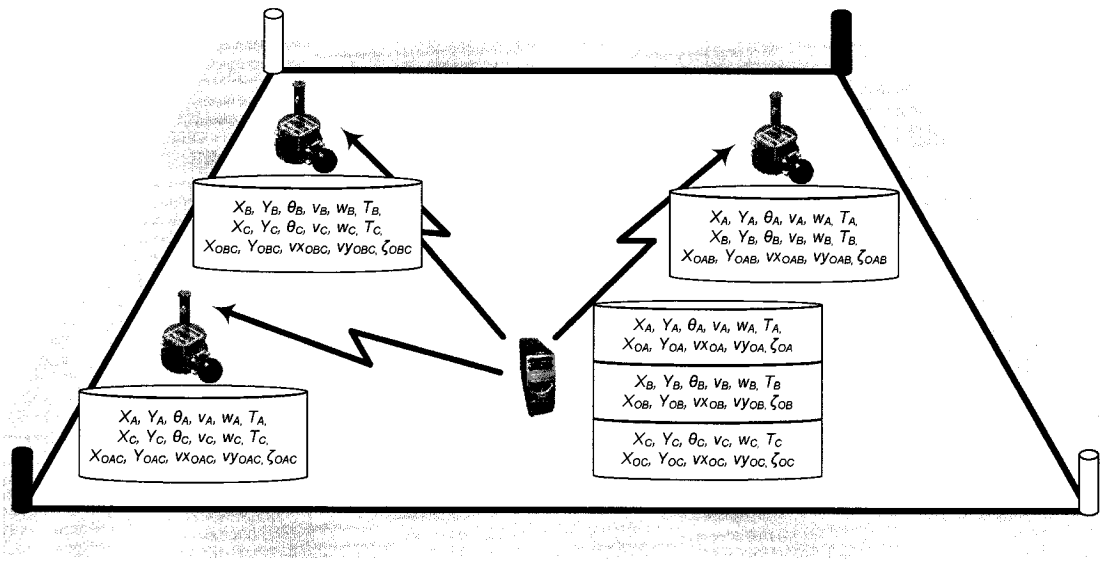

Figure 5.5: Partage d'informations - Exemple de réponse du serveur

l'état estimé par la prédiction de l'état précédent et la correction par les mesures. En pratique, tous les états ne peuvent être mesurés en une seule itération. Il faudrait donc transmettre au serveur les mesures effectuées ainsi que tous les paramètres permettant d'utiliser ces mesures. De plus, comme les taux de variation des mesures et des paramètres sont inconnus, il serait impossible d'effectuer une prédiction pour synchroniser les mesures reçues et les mettre en commun. Pour ces raisons, les états estimés sont utilisés et la mise en commun résultante n'est pas optimale. La mise en commun d'états estimés plutôt que mesurés fait en sorte que l'apport de nouvelles informations est moins important. Cela a pour effet que l'état estimé après le partage a été globalement moins corrigé par les nouvelles mesures et l'erreur estimée est sous-estimée. Ces effets négatifs peuvent être réduits en augmentant les bruits de processus ou les erreurs associées aux états lors du transfert de l'information au serveur. Les robots utilisent alors davantage leurs propres mesures augmentant ainsi l'apport de nouvelles informations aux états partagées.

La figure 5.6 présente l'architecture client-serveur utilisée pour effectuer le partage d'informations. Sur la figure, un disque représente une structure d'information à partager. Une structure est de type robot ou objet dynamique et elle contient la

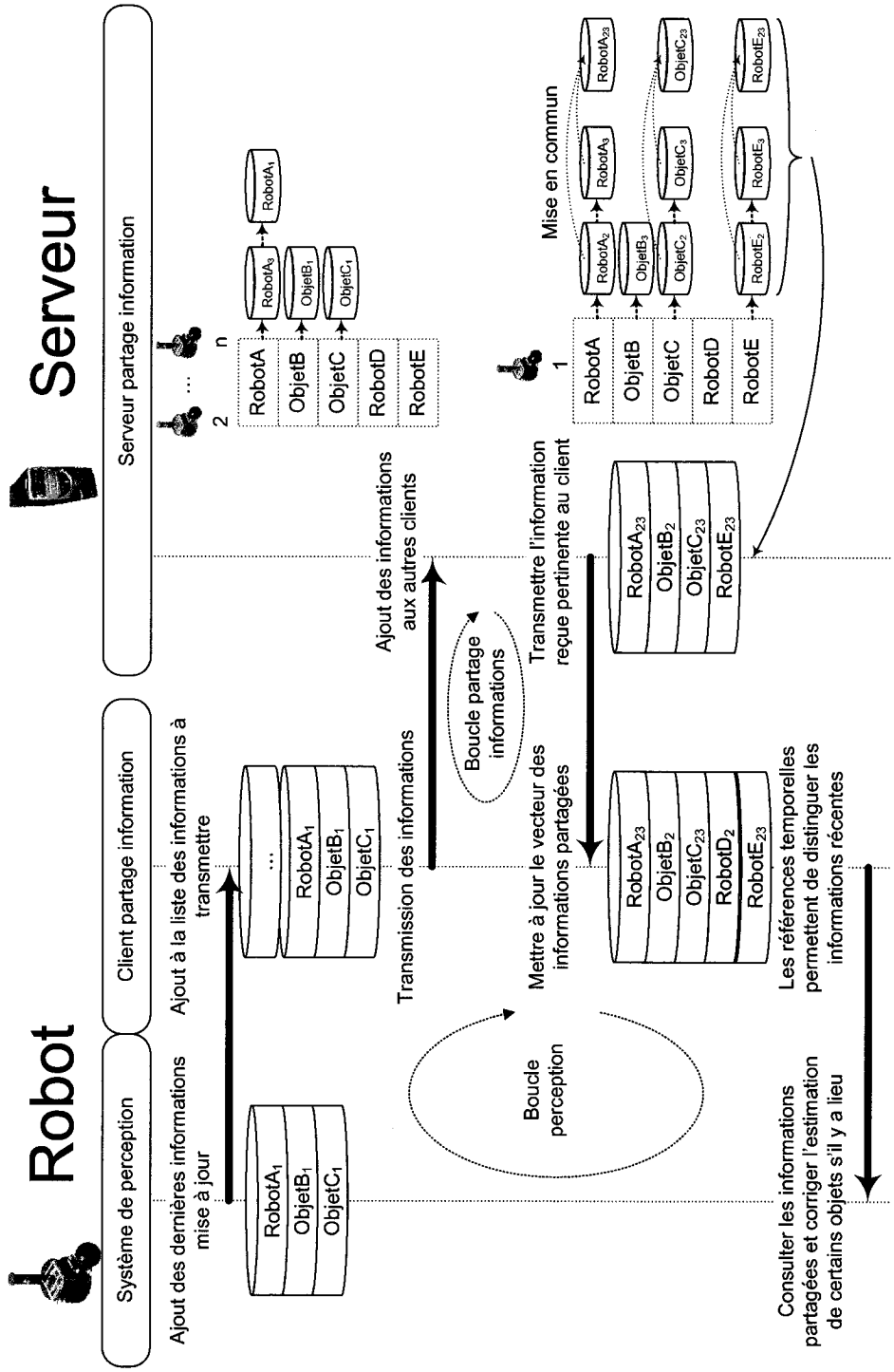

Figure 5.6: Partage d'informations - Architecture client-serveur

Tableau 5.1: Informations partargées pour le type robot

Symbole	***Descriptions***	***Unités***
t	Référence temporelle	s
x	Coordonnée en x de la position	m
y	Coordonnée en y de la position	m
θ	Orientation du robot	rad
v	Vitesse tangentielle	$\frac{m}{s}$
w	Vitesse angulaire	$\frac{rad}{s}$
B_v	Bruit de processus sur la vitesse tangentielle	$\frac{m^2}{s}$
B_w	Bruit de processus sur la vitesse angulaire	$\frac{rad^2}{s}$
$E(\vec{q}\vec{q}^T)$	Matrice de covariance des erreurs des états x, y, θ, v et w	

référence temporelle, l'état, les bruits de processus à considérer lors de la prédiction ainsi que la matrice de la covariance des erreurs associées aux états. Lorsqu'un client se connecte au serveur, il se synchronise avec le serveur. Ensuite, il authentifie les objets pour lesquels il désire fournir et recevoir de l'information. Un nom est associé à chacun des objets pour les différentier. Si un objet est inconnu, le serveur l'ajoute aux vecteurs de chacun des robots connectés. Ensuite, le seveur retourne au client les numéros d'identification des objets demandés. Chaque numéro est unique et il est utilisé pour transmettre les informations, car le transfert du nom est de taille variable et généralement plus long. Un objet peut être ajouté au client à n'importe quel moment. L'objet est alors authentifié suivant le même processus. Sur la figure 5.6, le numéro d'identification est représenté par les lettres majuscules et le chiffre en indice identifie le client qui a envoyé l'information.

Le partage d'informations est effectué dans un processus concurrentiel différent afin d'éviter que le processus principal (période d'environ $20ms$) et le processus de perception (période d'environ $100ms$) soient dépendant du partage de l'information avec le serveur. Ces deux processus partagent l'information directement avec le client et le client assure ensuite la connection avec le serveur et le partage de l'information. Lorsqu'une nouvelle information est ajoutée au client par un des processus du robot,

Tableau 5.2: Informations partargée pour le type objet dynamique

Symbole	***Descriptions***	***Unités***
t	Référence temporelle	s
x	Coordonnée en x de la position	m
y	Coordonnée en y de la position	m
$\dot{x}$	Vitesse de l'objet par rapport à l'axe des x	$\frac{m}{s}$
$\dot{y}$	Vitesse de l'objet par rapport à l'axe des y	$\frac{m}{s}$
ζ	Coefficient de frottement visqueux	$\frac{kg}{s}$
B_{vx}	Bruit de processus sur la vitesse par rapport à l'axe des x	$\frac{m^2}{s}$
B_{vy}	Bruit de processus sur la vitesse par rapport à l'axe des y	$\frac{m^2}{s}$
B_{ζ}	Bruit de processus sur le coefficient de frottement visqueux	$\frac{kg^2}{s}$
$E(\vec{q}\vec{q}^T)$	Matrice de covariance des erreurs des états x, y, $\dot{x}$, $\dot{y}$ et ζ	

elle est ajoutée à une liste d'attente avant d'être transférée. Le client transfère l'information le plus rapidement possible après l'avoir reçue. Il attend cependant pendant un délai minimum fixé entre chaque transferts pour limiter la bande passante utilisée. Lorsqu'une nouvelle information est ajoutée au client, l'ancienne information relative au même objet est retirée de la liste si elle n'a pas encore été transférée. Cette démarche particulière évite de surcharger la liste d'attente.

Sur le serveur, chaque client possède un vecteur contenant, pour chacun des objets, une liste des informations reçues depuis la dernière mise à jour. Ainsi, lorsque le serveur reçoit une nouvelle information pour un objet, il l'ajoute à la liste de cet objet de chacun des clients. À titre d'exemple, on remarque sur la figure qu'une information avait déjà été ajoutée pour l'objet *RobotA*, par le client 3, dans la liste du client 2. Après avoir ajouté les nouvelles informations, le serveur construit à partir des informations contenues dans les listes, le vecteur contenant les informations à retourner au client. Si plusieurs informations sont disponibles pour un même objet, elles sont mises en commun en fonction du type d'objet et seulement l'information résultante est retournée.

À partir de l'information retournée par le serveur, le client met à jour son vecteur contenant les dernières informations reçues. Ce vecteur peut être accédé en tout temps par les autres processus du robot et sa taille ne change que si un nouvel objet est authentifié. Les références temporelles doivent être utilisées pour déterminer si une nouvelle information a été reçue. Si le robot n'estime pas l'état d'un objet partagé, il peut utiliser directement l'information reçue. Toutefois, si le robot estime l'état d'un objet partagé, il doit utiliser le même filtre de *Kalman* que celui utilisé par le serveur pour mettre à jour sa propre estimation.

Ce mécanisme de partage d'informations peut sembler complexe à première vue, mais il simplifie grandement la gestion de la connexion avec le serveur et il permet d'utiliser facilement plusieurs sources de mesures différentes pour un même objet. Un système de vision global peut être ajouté au système sans avoir à modifier l'application des robots. Les mesures du système de vision global doivent être transmises au serveur et elles seront combinées et utilisées de la même manière que les autres mesures par les robots. Si l'on désire que les robots utilisent seulement l'information transmise par une source en particulier, il suffit d'associer une erreur et des bruits de processus nuls à l'information transmise. L'algorithme utilisé pour le partage de l'information permet donc d'ajouter facilement plusieurs sources d'informations et de pondérer leur utilisation.

5.3 Solution par filtrage de Kalman

5.3.1 Filtre de synchronisation

Plusieurs techniques peuvent être utilisées pour synchroniser les minuteurs des robots avec celui du serveur. Toutefois, la synchonisation des minuteurs n'est pas permanente et elle doit être effectuée à chaque fois que l'application d'un robot est relancée. Une technique simple de synchronisation pouvant être effectuée rapidement consiste à utiliser le réseau sans-fil pour estimer l'écart entre la valeur du minuteur du serveur et celle du minuteur local (t_s). La figure 5.7 présente la technique utilisée. Le client envoie une requête au serveur pour obtenir la valeur de son minuteur. Par hypothèse, la valeur reçue correspond à la valeur du minuteur du serveur après la moitié du délai de transmission et de traitement ($\frac{1}{2}t_d$). Une seule requête n'est pas suffisante pour obtenir une estimation précise de t_s et t_d. Toutefois, en utilisant un filtre de *Kalman*, une estimation précise peut être obtenue après plusieurs itérations.

Les états estimés par le filtre de *Kalman* de la synchronisation sont l'écart de temps entre le minuteur du serveur et le minuteur local ainsi que le délai de transmission et de traitement.

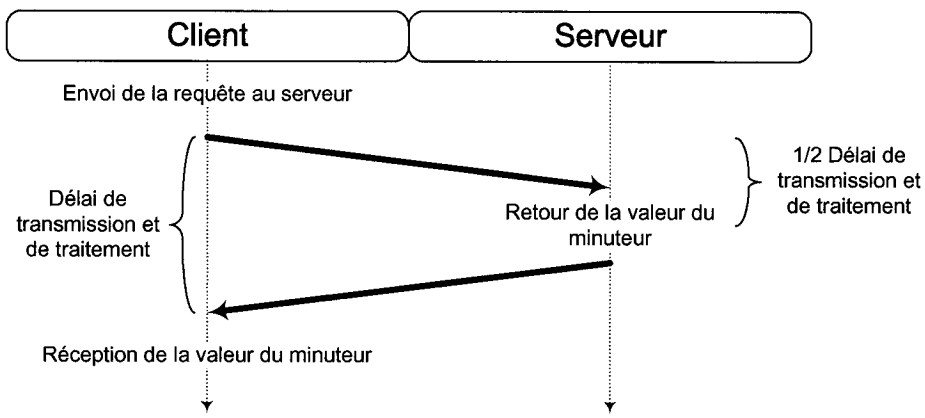

Figure 5.7: Partage d'informations - Synchronisation

Soit:

$$\hat{q} = \begin{bmatrix} \hat{t_s} \\ \hat{t_d} \end{bmatrix}$$

Les expressions du filtre de *Kalman* sont données par:

$$f\left(\hat{q}_{k-1}, 0, 0\right) = \begin{bmatrix} \hat{t_s} \\ \hat{t_d} \end{bmatrix} \tag{5.1}$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.2}$$

$$W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.3}$$

$$Q = \begin{bmatrix} 1 \times 10^{-12} & 0 \\ 0 & 1 \times 10^{-12} \end{bmatrix} \tag{5.4}$$

$$h(\tilde{q}) = \begin{bmatrix} t_{local} + \tilde{t_s} + \frac{1}{2}\tilde{t_d} \\ \tilde{t_d} \end{bmatrix} \tag{5.5}$$

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.6}$$

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.7}$$

$$R = \begin{bmatrix} B_m & 0 \\ 0 & B_d \end{bmatrix} \tag{5.8}$$

Les valeurs estimées sont constantes. Ainsi, il est inutile d'effectuer une prédiction et les bruits de processus sont nuls. Le bruit B_d représente la variation du délai de

Tableau 5.3: États, constantes et bruits de mesures

Symboles	***Descriptions***	***Unités***
t_s	Écart entre le minuteur du serveur et le minuteur local.	s
t_d	Délai moyen de transmission et de traitement.	s
t_{local}	Valeur du minuteur local.	s
B_s	Bruit de mesure sur la valeur du minuteur du serveur	s^2
B_d	Bruit de mesure sur le délai de transmission et de traitement	s^2

transmission et de traitement. Lorsque la bande passante du réseau est peu utilisée et que le serveur n'est pas sollicité, le délai de transmission et de traitement varie généralement très peu. Toutefois, il est judicieux de surestimer le bruit pour s'assurer que l'estimation converge correctement, même si le délai est supérieur à la normale pendant quelques itérations. Par hypothèse, le serveur retourne la valeur de son minuteur après que la moitié du délai de transmission et de traitement se soit écoulée. Le bruit B_s représente donc la variation du temps où la valeur du minuteur a réellement été retournée. Encore une fois, il peut être judicieux de surestimer cette valeur. En surestimant les bruits, la robustesse de l'estimation augmente, car l'erreur estimée décroit moins rapidement d'une itération à l'autre.

5.3.2 Filtre des robots mobiles

Ce filtre permet la prédiction et la mise en commun d'objets dont la dynamique est similaire à celle d'un robot mobile avec une contrainte non-holonome. Les états sont la position, l'orientation, la vitesse tangentielle et la vitesse angulaire. Par hypothèse, les vitesses tangentielles et angulaires sont constantes. Lors d'une prédiction, la position évolue en fonction de l'orientation de l'objet et de la vitesse tangentielle. L'orientation évolue en fonction de la vitesse angulaire. Lorsque l'intervalle de temps d'une prédiction est relativement long, il est préférable de le séparer en plusieurs petits intervalles et d'effectuer des prédictions subséquentes. De cette façon, l'état et l'erreur associée sont prédits plus précisément. Puisque les mesures sont simplement un autre vecteur d'états, l'algorithme de correction du filtre est dans sa forme

la plus simple lors de la mise en commun. La matrice H est donc une matrice identité.

Lors de la conception du système de localisation, il a été jugé inutile d'inclure les vitesses tangentielle et angulaire au vecteur d'états du système. L'odométrie permet de les mesurer précisément et la vision n'apporte pas plus de précision. Toutefois, pour prédire la position du robot sans les mesures d'odométrie, il faut connaître la vitesse du robot. Ainsi, lors du partage d'information concernant la position, l'orientation et les erreurs associées sont données par le système de localisation. Les vitesses tangentielles et angulaires sont déterminées à partir des mesures d'odométrie. Les erreurs sur les vitesses sont estimées en fonction de la précision des encodeurs optiques et du temps écoulé entre les deux mesures utilisées. Les bruits de processus B_v et B_w correspondent respectivement aux bruits sur la vitesse tangentielle et à ceux sur la vitesse angulaire. Ils quantifient les accélérations prévues du robot. Ils influencent le rythme auquel l'erreur estimée augmente en fonction de l'intervalle de temps d'une prédiction.

Soit:

$$\hat{q} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \\ \hat{v} \\ \hat{w} \end{bmatrix} \quad u = \begin{bmatrix} \Delta t \end{bmatrix}$$

Les expressions du filtre de *Kalman* sont données par:

$$f\left(\hat{q}_{k-1}, u_{k-1}, 0\right) = \begin{bmatrix} \hat{x} + \hat{v}\Delta t \cos\hat{\theta} \\ \hat{y} + \hat{v}\Delta t \sin\hat{\theta} \\ \hat{\theta} + \hat{w}\Delta t \\ \hat{v} \\ \hat{w} \end{bmatrix} \tag{5.9}$$

$$A = \begin{bmatrix} 1 & 0 & -\hat{v}\Delta t \sin\theta & \Delta t \cos\theta & 0 \\ 0 & 1 & \hat{v}\Delta t \cos\theta & \Delta t \sin\theta & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.11}$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & B_v|\Delta t| & 0 \\ 0 & 0 & 0 & 0 & B_w|\Delta t| \end{bmatrix} \tag{5.12}$$

$$h(\tilde{q}) = \tilde{q} \tag{5.13}$$

$$H = I_{[5,5]} \tag{5.14}$$

$$V = I_{[5,5]} \tag{5.15}$$

5.3.3 Filtre des objets dynamiques

L'étape de prédiction et les états utilisés pour le partage des objets dynamiques sont les mêmes que ceux du filtre présenté à la section 4.2.1. Seule l'étape de correction est différente puisque la correction n'est pas apportée à partir des mesures du système de vision, mais à partir d'un autre vecteur d'états. Le filtre présenté dans cette section est le filtre utilisé par le serveur pour mettre l'information en commun. Sur le robot, le filtre (développé dans la section 4.2.1) a été modifié pour permettre de corriger l'état estimé à l'aide des informations reçues par le serveur et à l'aide des mesures du système de vision.

Soit:

$$\hat{q} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \dot{\hat{x}} \\ \dot{\hat{y}} \\ \hat{\zeta} \end{bmatrix} \qquad u = \begin{bmatrix} \Delta t \end{bmatrix}$$

Les expressions du filtre de *Kalman* sont données par:

$$f\left(\hat{q}_{k-1}, u_{k-1}, 0\right) = \begin{bmatrix} \hat{x} + \dot{\hat{x}}\Delta t + \frac{\hat{\zeta}}{2}\dot{\hat{x}}\Delta t^2 \\ \hat{y} + \dot{\hat{y}}\Delta t + \frac{\hat{\zeta}}{2}\dot{\hat{y}}\Delta t^2 \\ \dot{\hat{x}} + \hat{\zeta}\dot{\hat{x}}\Delta t \\ \dot{\hat{y}} + \hat{\zeta}\dot{\hat{y}}\Delta t \\ \hat{\zeta} \end{bmatrix} \tag{5.16}$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t + \frac{\hat{\zeta}}{2}\Delta t^2 & 0 & \frac{1}{2}\dot{\hat{x}}\Delta t^2 \\ 0 & 1 & 0 & \Delta t + \frac{\hat{\zeta}}{2}\Delta t^2 & \frac{1}{2}\dot{\hat{y}}\Delta t^2 \\ 0 & 0 & 1 + \hat{\zeta}\Delta t & 0 & \dot{\hat{x}}\Delta t \\ 0 & 0 & 0 & 1 + \hat{\zeta}\Delta t & \dot{\hat{y}}\Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

$$W = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

$$Q = \begin{bmatrix} B_{vx}|\Delta t| & 0 & 0 \\ 0 & B_{vy}|\Delta t| & 0 \\ 0 & 0 & B_{\zeta}|\Delta t| \end{bmatrix} \tag{5.19}$$

$$h(\tilde{q}) = \tilde{q} \tag{5.20}$$

$$H = I_{[5,5]} \tag{5.21}$$

$$V = I_{[5,5]} \tag{5.22}$$

5.4 Résultats

5.4.1 Synchronisation

Les figures 5.8 et 5.9 présentent les résultats obtenus lors d'une synchonisation simulée avec *Matlab*. Les courbes *Mesures* représentent le résultat du calcul, en une seule itération, de l'écart entre les deux minuteurs (t_s) ainsi que du délai de transmission et de traitement (t_d). L'estimé initial du filtre est calculé de cette façon et l'erreur initiale est surestimée pour assurer qu'elle soit supérieure à l'erreur réelle de l'estimé initial. On remarque que l'erreur estimée décroit très rapidement lors des itérations et qu'elle décroit beaucoup moins rapidement par la suite. La décroissance de l'erreur estimée est proportionnelle aux bruits associés aux mesures. Ainsi, plus les bruits de mesures sont élevés, plus le nombre d'itérations nécessaires pour obtenir une erreur estimée donnée est grand.

Les figures 5.10 et 5.11 présentent une situation où le délai de transmission et le délai de traitement du serveur sont grandement modifiés pendant les itérations de 10 à 20. La figure 5.10 présente les résultats avec des bruits de mesures inchangés tandis que la figure 5.11 présente les résultats pour des bruits deux fois plus grand. On remarque que la convergence de l'erreur réelle n'est pratiquement pas affectée par la valeur des bruits de mesures. Toutefois, la convergence de l'erreur estimée est beaucoup plus lente lorsque les bruits de mesures sont plus élevés. Ainsi, si la valeur de l'erreur estimée est le critère d'arrêt, le nombre d'itérations sera alors plus grand et l'erreur réelle finale sera inférieure à l'erreur estimée.

Par hypothèse, le serveur retourne la valeur de son minuteur lorsque la moitié du délai de transmission et de traitement est écoulé. La figure 5.12 présente les conséquences sur la synchronisation d'une variation de 20% de la fraction de 0.5 prévue. On remarque sur la figure que l'écart entre les deux minuteurs est faussé ce qui entraîne inévitablement une erreur de synchronisation. En pratique, il a été impossible d'évaluer la précision de la synchronisation. La précision espérée doit être de quelques millisecondes sans quoi la précision de la mise en commun de l'information sera réduite par l'erreur de synchronisation.

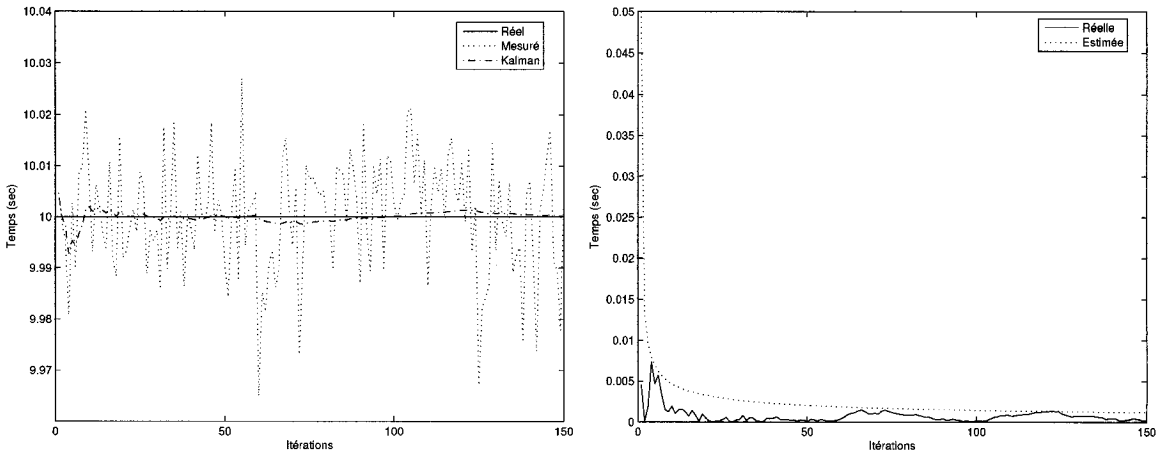

Figure 5.8: Estimation de l'écart entre les deux minuteurs

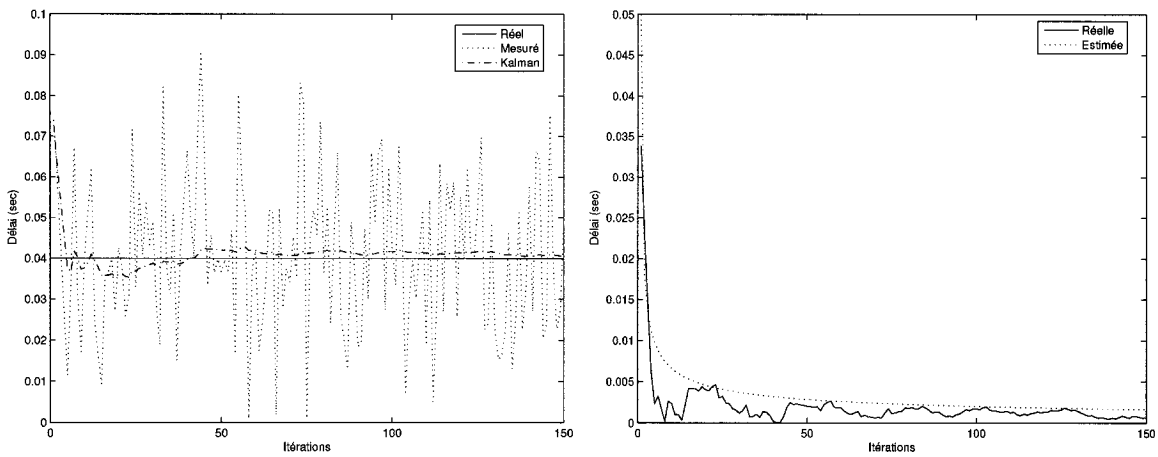

Figure 5.9: Estimation du délai de transmission et de traitement

5.4.2 Partage d'informations

La principale application du système est le soccer robotisé. L'information partagée contient ainsi la position de chaque robot et la position du ballon. Puisque chaque robot mesure sa propre position, aucune mise en commun n'est nécessaire lors du partage des positions. Toutefois, le ballon est un objet dynamique dont la position est estimée par chacun des robots. Les résultats présentés dans cette section

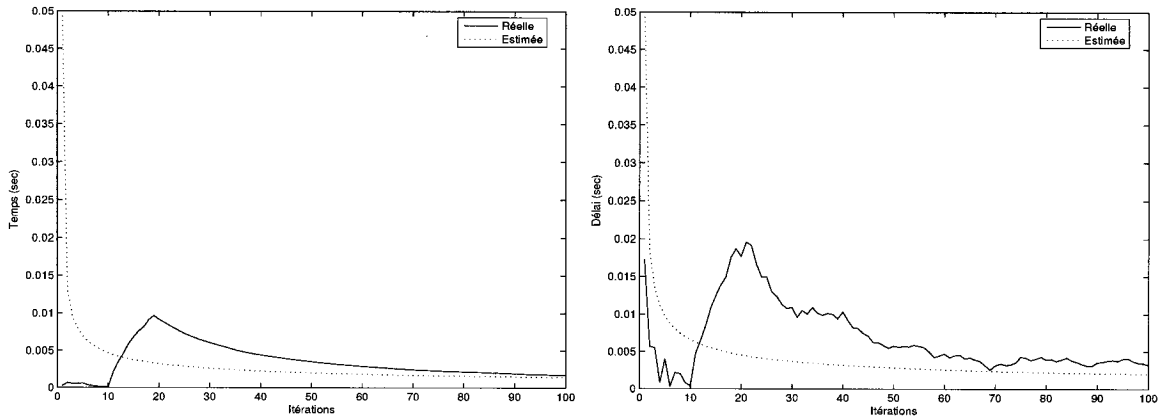

Figure 5.10: Erreurs sur l'écart et le delai avec perturbations

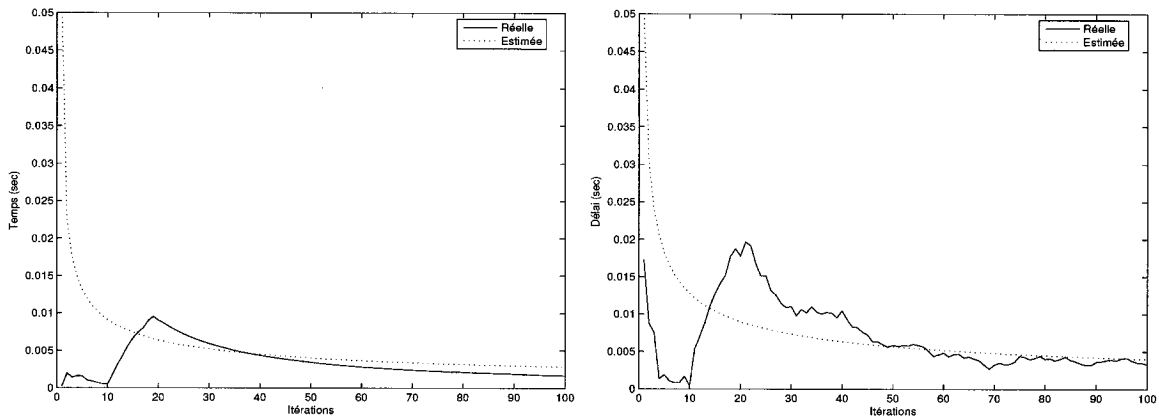

Figure 5.11: Erreurs, écart et delai, avec perturbations et surestimation des bruits

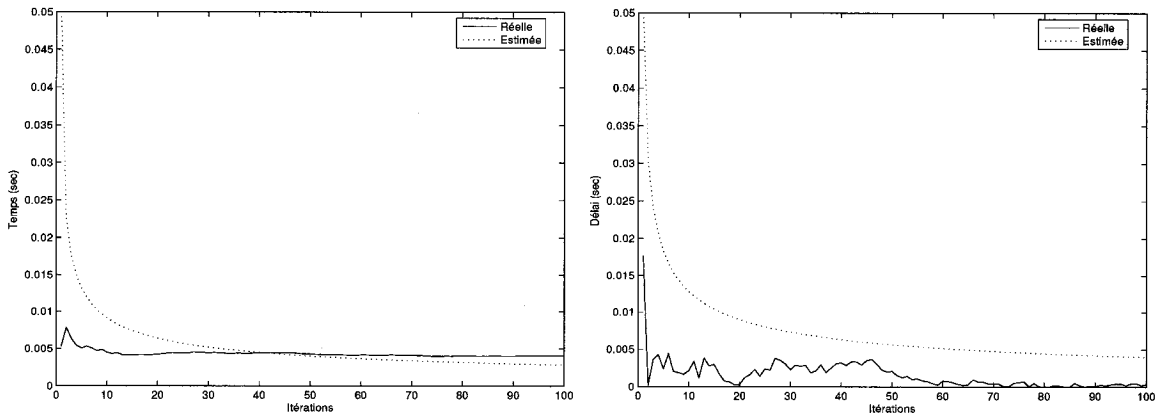

Figure 5.12: Erreurs sur l'écart et le delai lorsque l'hypothèse n'est pas respectée

représentent des situations différentes où la position du ballon est partagée. Le partage d'informations concernant la position d'un robot a aussi été implanté. Cependant, les résultats sont similaires à ceux du partage du ballon et ils ne seront pas présentés dans cette section.

5.4.2.1 Simulations

Afin de bien comprendre les objectifs et les performances attendues avec le partage d'informations, il est préférable d'observer une situation idéale. La figure 5.13 présente la trajectoire de l'objet dynamique et la position de deux robots. La trajectoire de l'objet débute près du robot B pour se terminer près du robot A. On remarque que la trajectoire a été perturbée afin de rendre les résultats plus significatifs. On remarque aussi que l'erreur de position de l'objet est beaucoup plus faible au départ pour le robot B que pour le robot A étant donné que le robot B est plus près de l'objet. De la même façon, l'erreur est plus faible pour le robot A lorsque l'objet se trouve près de lui. Ces observations sont valables autant pour l'erreur réelle de position que pour l'erreur estimée.

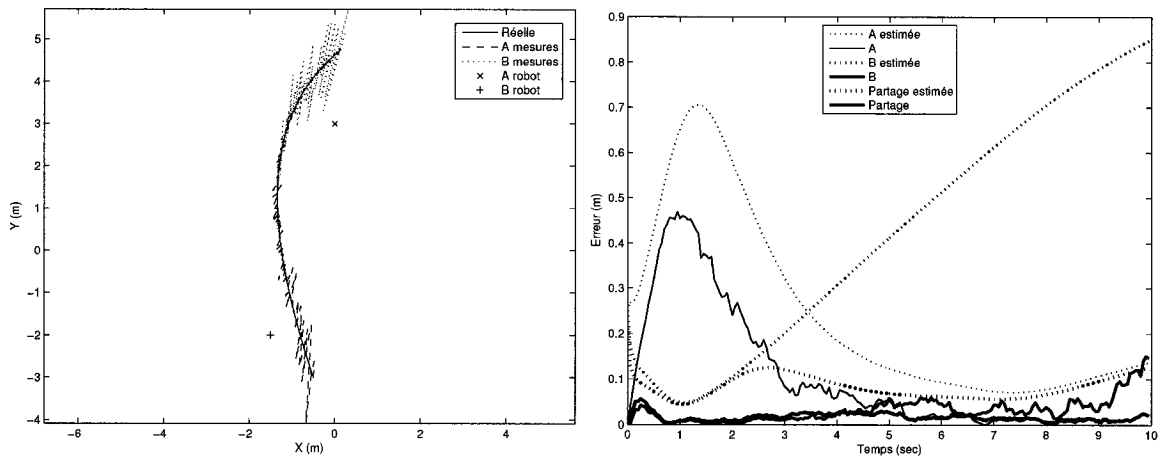

Figure 5.13: Estimation de la position avec le partage d'informations

Dans la situation idéale, il n'y a pas de délai de transmission. Ainsi, les mesures sont effectuées et mises en commun en même temps par les deux robots. L'erreur de position résultant de la mise en commun est à tout moment inférieure à l'erreur de

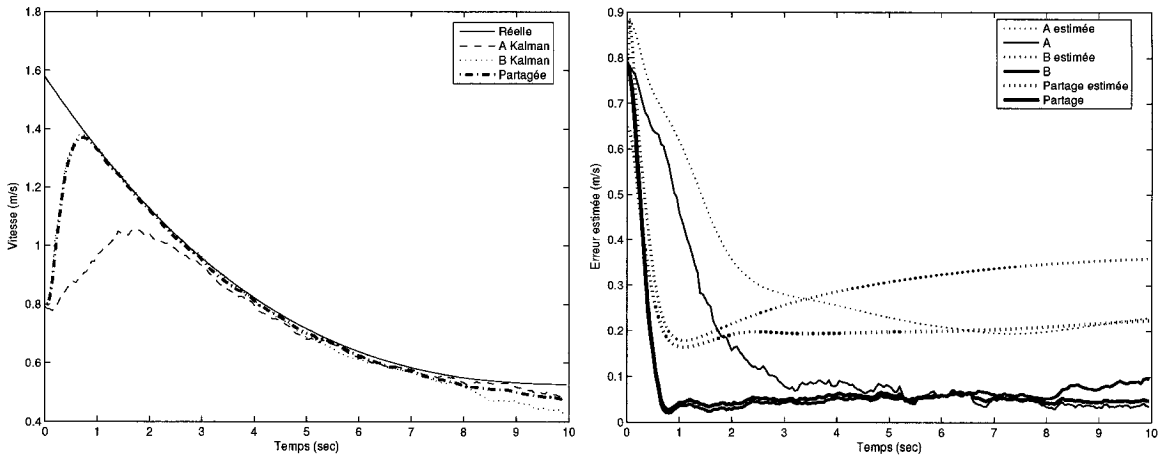

Figure 5.14: Estimation de la vitesse avec le partage d'informations

position la plus précise entre celle mesurée par robot A et celle mesurée par le robot B. Le partage d'informations améliore aussi grandement l'estimation de la vitesse comme on peut le constater sur la figure 5.14. Les mesures du robot B sont plus précises étant donné qu'il se trouve plus près de l'objet. Ainsi, il estime plus rapidement la vitesse de cet objet. L'estimation partagée de la vitesse converge alors aussi rapidement que l'estimation effectuée par le robot B. Lorsque l'objet s'éloigne du robot B, l'estimation partagée de la vitesse demeure précise puisque qu'elle bénéficie alors des mesures plus précises du robot A.

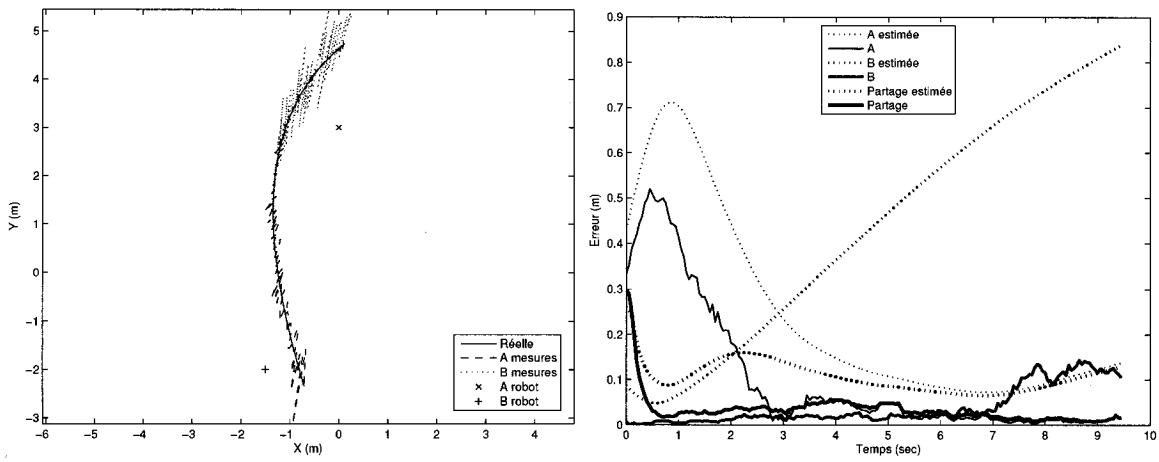

Figure 5.15: Estimation de la position avec le partage d'informations avec délai

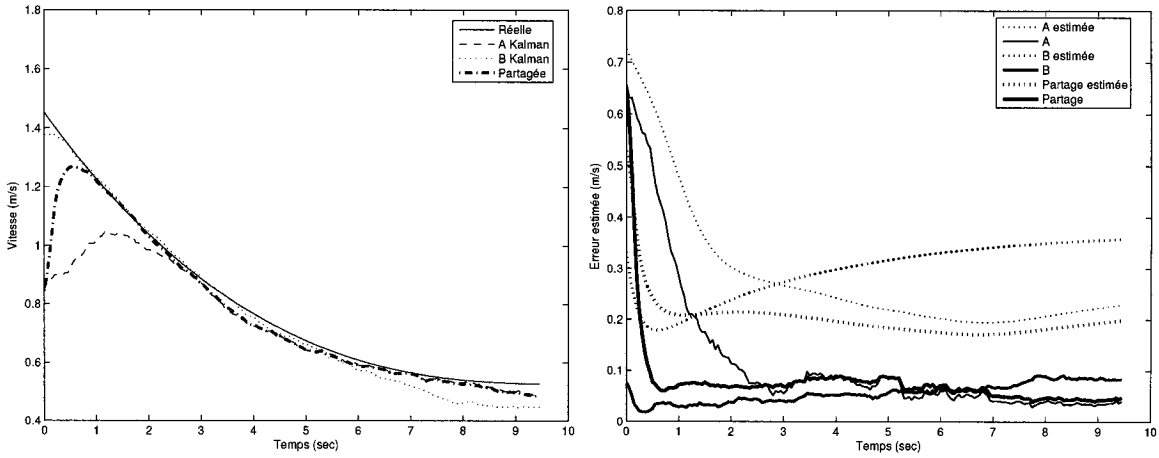

Figure 5.16: Estimation de la vitesse avec le partage d'informations avec délai

Les figures 5.15 et 5.16 présentent les résultats d'une simulation avec un délai de communication. Le robot A reçoit les informations du robot B avec un retard de $500ms$. La position et la vitesse courantes de l'objet doivent donc être prédites à partir de la position et de la vitesse reçues. Étant donné que la précision de la position et de la vitesse après la prédiction est moins grande, la précision après la mise en commun sera moins grande que la précision obtenue dans le cas idéal. Sur les figures, les courbes des erreurs réelles et estimées par le robot B correspondent aux courbes des erreurs de position et de vitesse, tel qu'elles ont été mesurées par le robot B avant la transmission et la prédiction. Lorsque le robot A combine la position et la vitesse reçues par le robot B avec les siennes, les erreurs réelles et estimées de la position et de la vitesse reçues par le robot B après prédiction sont plus grandes que celles représentées par les courbes. Toutefois, même si l'information est reçue après un certain délai, elle permet d'améliorer la précision des estimations de la position et de la vitesse du robot A. Lorsque le délai de transmission est très court, les résultats sont comparables au cas idéal. Lorsqu'il est très long, la prédiction engendre trop d'imprécisions et le partage n'améliore plus de manière significative la précision de l'estimation.

Pour évaluer correctement le délai entre deux informations provenant de deux robots différents, il est primordial que la synchronisation entre les deux robots soit précise. Dans le cas contraire, l'intervalle de temps utilisé pour effectuer la prédiction

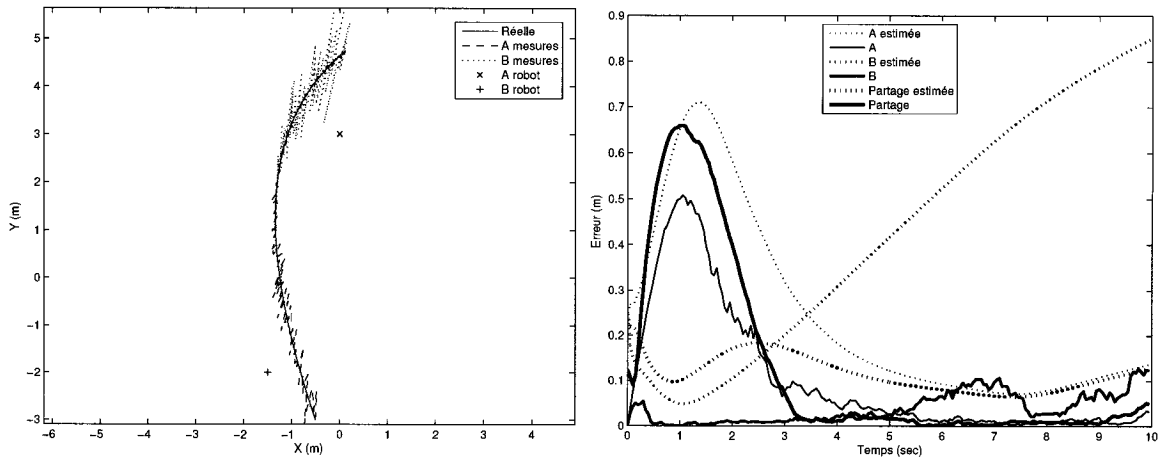

Figure 5.17: Estimation de la position avec le partage d'informations et retard

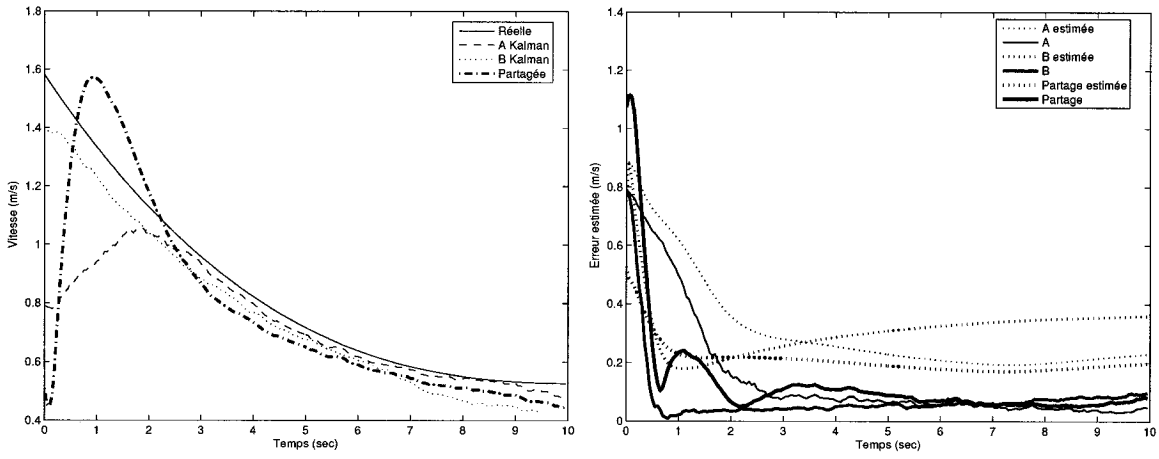

Figure 5.18: Estimation de la vitesse avec le partage d'informations et retard

est incorrect et la mise en commun est alors effectuée avec des informations asynchrones. Les figures 5.17 et 5.18 présentent les conséquences de l'utilisation sans synchronisation d'une information en retard. La position et la vitesse estimées par le robot B sont reçues par le robot A après un délai de $500ms$. Elles sont ensuite mises en commun avec la position et la vitesse mesurées par le robot A sans qu'une prédiction ne soit effectuée pour les synchroniser.

La position résultant de la mise en commun d'une position courante avec une position en retard est inévitablement en retard par rapport à la position courante de l'objet. Ce retard est proportionnel au retard de la position utilisée et à la précision qui lui est associée. Ainsi, si le robot A mesure la position de l'objet avec une grande précision et qu'il reçoit une position en retard de faible précision par le robot B, le retard a peu d'impact puisque la position reçue est peu considérée. Une vitesse reçue après un retard significatif est problématique, car elle fausse l'estimation de la vitesse de l'objet et par conséquent la prédiction de la position future. Sur la figure 5.18 le phénomène est important puisque la vitesse reçue en retard a une précision estimée relativement grande par rapport à celle mesurée directement par le robot A. La vitesse est donc surestimée et l'erreur réelle après la mise en commun est fortement supérieure à l'erreur estimée.

L'utilisation d'une information en retard peut dégrader grandement la précision des informations estimées par le robot. Plus le retard est important et plus la vitesse réelle d'un objet est grande, plus la qualité de l'estimation sera dégradée. Toutefois, en synchronisant les informations à l'aide d'une prédiction, on évite les effets néfastes du retard. L'information en retard et l'erreur associée sont actualisées par rapport au référentiel temporelle de l'information la plus récente. L'erreur associée augmente alors en fonction du retard et l'apport de l'information reçue est réduit en conséquence.

5.4.2.2 Expérimentations

Les figures 5.19 et 5.20 présentent les résultats du partage de la position et de la vitesse du ballon. Les deux robots sont statiques et le ballon passe près de l'un puis près de

de l'autre. En tout temps, chaque robot peut voir le ballon et en estimer la position. Ainsi, le partage de la position devrait leur permettre d'améliorer la précision de leur perception respective du ballon. Toutefois, on remarque, en observant la figure de la trajectoire du ballon, que les performances du partage d'informations sont moins bonnes qu'espérées. Dans un premier temps, les courbes de la perception du ballon effectuée par chacun des robots ne sont pas superposées. Normalement, les courbes devraient être superposées et bruitées différemment, en fonction de la précision des mesures de chacun des robots pour une position donnée du ballon.

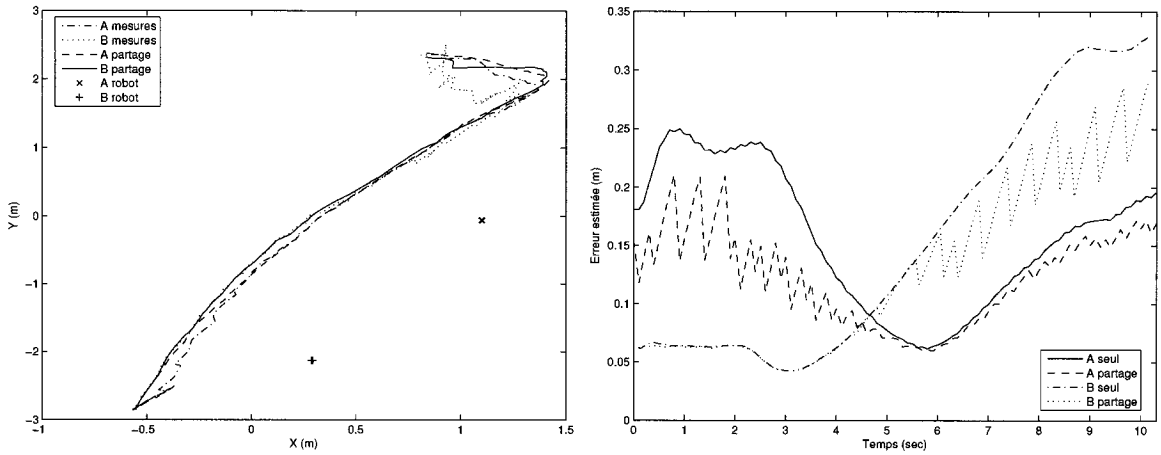

Figure 5.19: Estimation de la position avec le partage d'informations

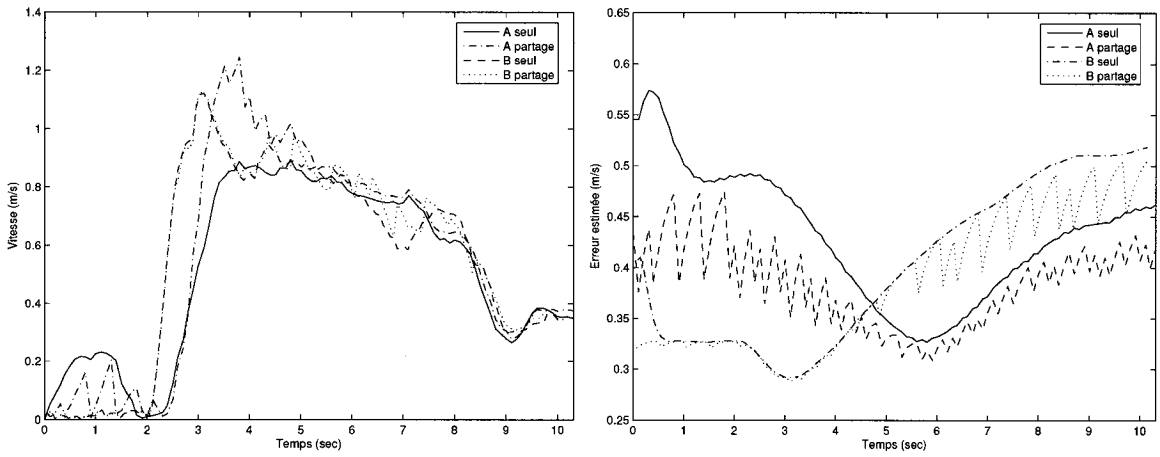

Figure 5.20: Estimation de la vitesse avec le partage d'informations

L'écart entre les deux courbes peut être attribuable principalement à deux facteurs: une erreur sur la position des robots et une erreur sur la perception du ballon causée par un mauvais étalonnage du montage de vision. La position des robots est précise à $\pm 2cm$ ce qui peut engendrer un écart de $\pm 4cm$ au maximum entre les deux courbes. Tel que discuté dans la section 4.4.2, la perception d'un objet dynamique est dégradée par un mauvais alignement ou un mauvais étalonnage du montage de vision principalement lorsque l'objet est loin du robot. La perception erronée et biaisée d'un objet, faite par un robot éloigné de cet objet, peut dégrader la précision de la perception d'un robot situé à proximité de ce même objet. Heureusement, l'erreur associée à la position erronée évite de trop dégrader la précision lors de la mise en commun. L'écart entre les positions perçues peut engendrer l'estimation d'une vitesse faible mais de moyenne non nulle même si l'objet est statique.

Les courbes d'erreurs présentées sont les courbes des erreurs estimées sur la position ou la vitesse du ballon avec et sans partage d'informations. Les courbes des erreurs réelles ne sont pas présentées puisque la trajectoire réelle du ballon est inconnue. Les courbes d'erreurs estimées avec partage d'informations sont en forme de dents de scie car les informations partagées sont reçues moins fréquemment que les mesures effectuées par le système de vision. Ainsi, lorsqu'une information partagée est reçue, l'erreur diminue rapidement pour ensuite augmenter à chaque mesure effectuée. Si le partage d'informations est interrompu pendant un moment, l'erreur estimée augmente jusqu'à devenir similaire à l'erreur estimée sans partage. De la même façon, si aucune mesure n'est disponible pendant un moment, l'erreur estimée devient similaire à l'erreur associée à l'information reçue, augmentée de l'erreur de prédiction nécessaire pour compenser le délai de transmission.

En comparant la figure 5.20 avec la figure 5.18, obtenue par simulation, on remarque qu'il semble y avoir un retard entre les positions mesurées localement et les positions reçues. Ce retard peut être attribuable à une erreur de synchronisation entre les clients et le serveur ou à une référence temporelle incorrecte. Les sections 4.4.3 et 3.5.2 démontrent la présence d'un délai entre l'acquisition de la référence temporelle et l'acquisition de l'image par la caméra. De plus, ces sections présentent

les conséquences sur l'estimation de la position du robot et sur l'estimation de la position d'un objet dynamique. Ce délai affecte la précision de la localisation du robot et, par le fait même, la précision de la position de l'objet dynamique. Ce délai fausse aussi la référence temporelle associée aux mesures. Par conséquent, la mise commun de la position d'un robot ou d'un objet dynamique est également erronée.

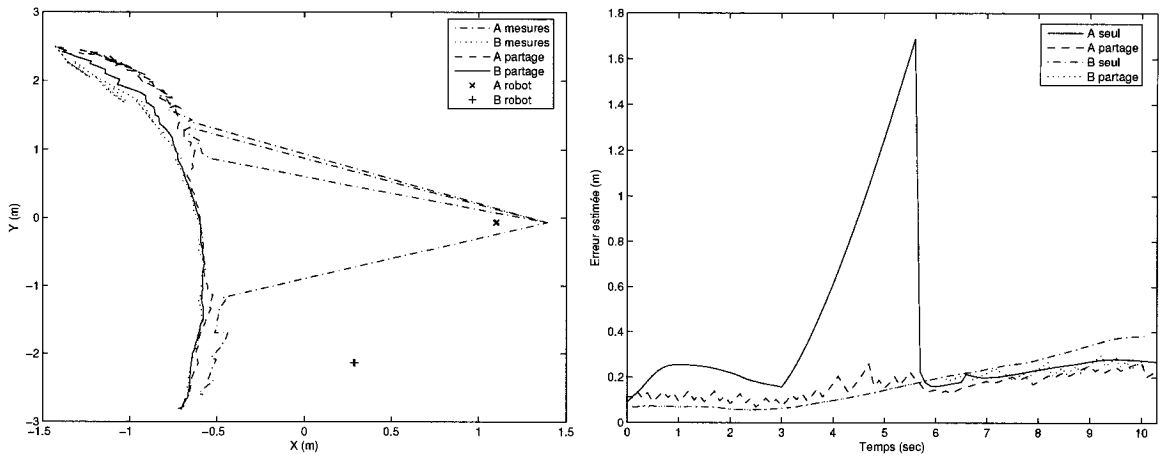

Figure 5.21: Estimation de la position avec le partage d'informations et obstruction

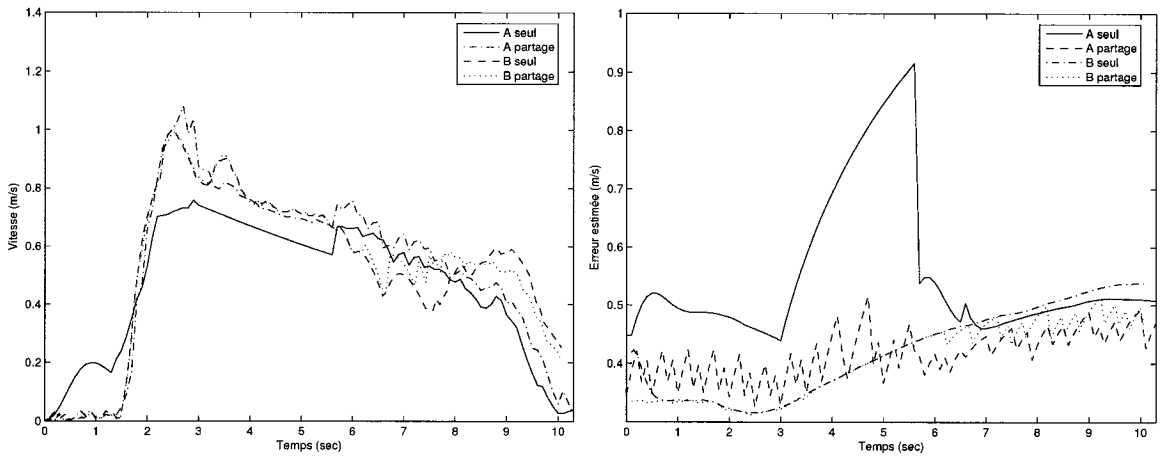

Figure 5.22: Estimation de la vitesse avec le partage d'informations et obstruction

Les figures 5.21 et 5.22 présentent une situation où la trajectoire du ballon fait en sorte que la vision du ballon est partiellement obstruée par un obstacle. Le ballon ne peut donc pas être observé par le robot A sur la totalité de la trajectoire. Les trajectoires présentées sont les trajectoires du ballon, mesurées par les systèmes de vision de

chacun des robots, ainsi que les trajectoires du ballon après le partage d'informations. Au départ, le robot A perçoit le ballon avec une moins grande précision que le robot B. Il bénéficie alors des informations envoyées par le robot B pour améliorer la précision de son estimation. Sans partage d'informations, le robot A ne peut corriger son estimation de la position du ballon lorsque ce dernier est caché par l'obstacle. La position du ballon est alors estimée à partir de la prédiction seulement. L'erreur estimée sur la position du ballon augmente rapidement à cause de la prédiction. Avec le partage d'informations, le robot A peut estimer la position de l'objet à partir des informations reçues par le robot B. L'erreur estimée augmente alors légèrement pour être comparable à l'erreur associée à l'information reçue avec le délai de transmission. Le système de vision poursuit son traitement normalement et aucun recouvrement n'est nécessaire pour repérer le ballon lorsqu'il apparait de l'autre côté de l'obstacle.

Sur l'image 5.22 on remarque clairement le moment pendant lequel seule la prédiction est utilisée par le robot A pour estimer la vitesse du ballon. Dans cet exemple, la prédiction du déplacement est relativement juste puisque le ballon se déplace librement. Toutefois, si la trajectoire du ballon avait été modifiée pendant que le ballon était dernière l'obstacle, la trajectoire avec le partage aurait été encore plus pertinente. On remarque également que la vitesse estimée après le partage est plus juste que celle mesurée par le robot A. Par contre, cela est moins évident dans le cas du robot B puisque le retard et les mesures faussées ajoutent des imprécisions à l'estimation après le partage.

Conclusion

Les trois principaux objectifs ayant motivé la réalisation de ce projet sont la localisation d'un robot dans son environnement à l'aide de balises, l'estimation de la position et du déplacement d'un objet ainsi que le partage d'informations entre les robots afin que chacun puisse bénificier de la perception des autres. Étant donné que la perception est basée principalement sur une caméra, ces objectifs n'auraient pas pu être atteints sans la réalisation, au préalable, d'un système de vision capable de reconnaître en temps réel des objets dans une image.

La reconnaissance d'un objet dans une image omnidirectionnelle est un problème complexe et sa réalisation en temps réel sur un robot autonome impose qu'elle soit à la fois rapide et robuste. Pour simplifier le problème, les objets recherchés sont par hypothèse de forme simple (cylindrique, sphérique ou rectangulaire), de taille connue et de couleur uniforme, constrastante et définie. De plus, pour alléger l'exécution et augmenter la robustesse, la reconnaissance et la recherche d'un objet sont basées sur la prédiction de sa position dans l'image. L'image n'est donc pas traitée en entier et la taille de la portion utilisée est proportionnelle à la précision estimée de la prédiction.

La représentation HSI de l'image a été préférée aux représentations RGB et YUV car elle découple l'information relative à la luminance et à la chrominance. Elle permet ainsi une segmentation couleur simple et efficace. Les zones des pixels regroupés par la segmentation sont validées en comparant leur taille et leur aire à celles qui avaient été prédites. Cette validation permet d'éliminer facilement et efficacement toutes les zones aberrantes. Lorsque la couleur et la forme des objets recherchés diffèrent beaucoup de celles des autres objets présents dans l'environnement, cette procédure de validation est suffissante pour les identifier. Le système de vision fonctionne cor-

rectement lorsque les hypothèses établies sont respectées. Le système est très sensible à la précision du montage de vision, aux caractéristiques des objets recherchés et à son environnement. Ainsi, pour permettre une plus grande flexibilité au système, pour rendre les hypothèses établies moins strictes, plusieurs améliorations peuvent être apportées.

Le système de vision retourne des mesures qui sont proportionnelles à la position de l'objet recherché dans l'image. Ces mesures sont utilisées par la suite en relation avec la position de la caméra lors de l'acquisition de l'image traitée. La position et l'orientation de la caméra doivent donc être connues avec précision afin que les mesures soient utilisées correctement. De plus, l'image retournée par une caméra omnidirectionnelle varie selon la lentille de la caméra, la courbure, la position et l'orientation du miroir utilisé. La précision de ces paramètres affectent directement la précision des mesures retournées. La précision du système de vision pourrait donc être grandement améliorée si certains de ces paramètres étaient estimés à partir de la position, dans l'image, d'objets dont la position dans l'environnement est connue avec précision.

Le système de vision est conçu pour un environnement dont l'éclairage est constant et uniforme. De plus, il est très sensible aux variations de l'intensité et de la couleur des sources lumineuses. Cela rend l'utilisation du système très difficile dans un environnement éclairé par une source de lumière naturelle. Lorsque l'éclairage de l'environnement est modifié, les couleurs perçues changent et s'écartent des intervalles attendus. La couleur des pixels correspondant à un objet devient alors différente de la couleur attendue et l'objet n'est plus reconnu. En mesurant, en temps-réel, la couleur des objets, le système de vision pourrait corriger les intervalles attendus et s'adapter à de faibles variations d'éclairage. De la même façon, le temps d'exposition d'une image pourrait être modifié en fonction de la luminosité des objets perçus afin de permettre une plus grande capacité d'adaptation.

Le système de localisation estime la position du robot en combinant de manière optimale les mesures asynchrones d'odométrie et de vision. Les mesures d'odométrie sont précises pour un court déplacement et elles peuvent être acquises rapidement.

Toutefois, pour déterminer la position du robot à partir des mesures d'odométrie, il faut additionner chaque mesure aux mesures précédentes, ce qui cumule les imprécisions. De leur côté, ces mesures de vision sont moins précises, moins fréquentes et elles demandent beaucoup plus de traitement. Toutefois, elles permettent de déterminer en une seule itération la position absolue du robot dans son environnement. Les mesures retournées par le système de vision sont les angles horizontaux (α_n) entre l'avant de la caméra et chacune des balises. Une balise est un assemblage d'objets dont la position est connue. Les mesures des angles verticaux (β_n) ne sont pas utilisées, car elles sont trop sensibles à l'étalonnage du montage de vision. Puisque l'étalonnage du montage est généralement peu précis, l'utilisation de ces mesures dégrade la qualité de la position estimée.

L'utilisation d'un filtre de *Kalman* permet une combinaison optimale des mesures d'odométrie et de vision. Le filtre estime précisément la position du robot et l'erreur qui lui est associée. Cette erreur estimée est ensuite réutilisée par le système de vision pour définir la largeur des portions de l'image où les objets sont recherchés. De plus, le filtre permet l'utilisation d'un nombre variable de balises. Ainsi, l'estimation se poursuit normalement même si plusieurs balises ne sont pas visibles. Le filtre estime aussi l'écart entre l'orientation de la caméra et l'orientation du robot. Cet écart provoque un biais dans les mesures de vision et il est corrigé automatique par le système.

Suite à un étalonnage précis du montage de vision, une amélioration significative pourrait être apportée. Elle consiste à incorporer au système les mesures de la largeur des objets et les mesures des angles verticaux β. Ces mesures permettraient d'estimer plus précisément la position du robot, notamment lorsque peu de balises sont visibles.

Un inconvénient du système actuel est la nécessité d'estimer la position initiale du robot lors de la réinitialisation du système. Pour éviter de devoir fournir au système la position du robot, il faudrait développer un algorithme de recouvrement capable de déterminer la position du robot à partir d'une image omnidirectionnelle de l'environnement du robot.

Dans ce travail, une caméra omnidirectionnelle a été utilisée, car elle a l'avantage de retourner une image de l'ensemble de l'environnement du robot. Cela permet donc de repérer, dans l'image, plusieurs balises et de calculer analytiquement la position du robot au moment où l'image a été acquise. Toutefois, l'utilisation d'un filtre de *Kalman* combinant les mesures de vision aux mesures d'odométrie a permis de constater qu'il n'est pas nécessaire d'acquérir en même temps toutes les mesures de vision. Ainsi, l'utilisation d'une caméra omnidirectionnelle n'était pas essentielle et plusieurs caméras unidirectionnelles pourraient très bien convenir au système développé.

Le système de perception d'objets dynamiques estime la position d'un objet à partir des mesures de vision en relation avec la prévision de la dynamique de l'objet. Ainsi, la précision de la position estimée d'un objet subissant de fortes accélérations est similaire à la précision de la position déterminée directement par les mesures, tandis que la précision de la position estimée d'un objet subissant de très faibles accélérations est grandement améliorée. Le filtre de *Kalman* d'un objet dynamique fournit aussi une estimation de l'erreur sur la position de l'objet en fonction de la dynamique prévue et de la manière dont la correction est apportée par les mesures de vision. Ainsi, l'erreur estimée est plus grande si l'objet est éloigné ou si sa dynamique est élevée. Une meilleure estimation de l'erreur permet une reconnaissance plus efficace par le système de vision.

Le filtre estime aussi la vitesse de l'objet en fonction de sa dynamique. La vitesse estimée est alors beaucoup plus précise que la vitesse calculée par la variation de la position dans un intervalle de temps donné. Cette estimation permet de prédire la position de l'objet et la direction du déplacement. Un coefficient de frottement visqueux est utilisé pour modéliser une accélération négative en fonction de la vitesse de l'objet. Ce coefficient peut être estimé par le filtre et son utilisation permet une prédiction plus précise de la position et de la vitesse de l'objet.

La précision de la position et de la vitesse estimées dépend grandement de la précision de l'étalonnage du système de vision. Un mauvais étalonnage fausse inévita-

blement la position estimée et il rend l'estimation du coefficient de frottement peu précis. De plus, le temps de traitement du système de vision fait en sorte que la position estimée est en retard par rapport à la position réelle de l'objet. Plus le temps de traitement est grand, plus la position estimée est en retard. Pour actualiser la position estimée, il est nécessaire d'effectuer une prédiction ce qui réduit grandement la précision de l'estimation.

Le partage d'informations entre les robots permet la mise en commun d'informations relatives à un même objet afin d'améliorer la précision de la perception des robots. La précision de la perception de chacun des robots après le partage et la mise en commun est similaire à celle du robot ayant la meilleure précision avant le partage. En plus d'améliorer la perception, le partage d'informations permet à un robot d'estimer l'état d'un objet à partir de l'information reçue même si aucune mesure n'est fournie par le système de vision. Il suffit qu'un seul robot mesure la position d'un objet pour que tous les autres robots en connaissent aussi la position. Lors d'une mise en commun, l'apport d'une information reçue est pondéré en fonction de la précision estimée par rapport à celle des autres informations. Ainsi, les informations plus précises sont privilégiées.

Pour mettre en commun plusieurs informations asynchrones, il est important de les synchroniser. Au préalable, les robots et le serveur de partage d'informations sont synchronisés entre eux afin d'utiliser le même référentiel temporel. Avant une mise en commun, les informations les moins récentes sont actualisées à l'aide d'une prédiction basée sur le modèle dynamique de l'objet concerné. Cette prédiction permet de limiter le retard engendré par le délai de traitement et de transmission des robots. Afin d'évaluer la précision de l'information prédite, les erreurs associées à l'information sont prédites aussi. Ceci permet ensuite de pondérer l'apport de chaque information lors de la mise en commun. Ce type d'algorithme de partage d'informations apporte une solution à plusieurs problèmes rencontrés lors du contrôle de procédés via un réseau IP dont les délais de transmission sont variables.

Toutefois, pour qu'une mise en commun améliore la précision d'une estimation,

il est important que les informations utilisées ne soient pas biaisées. Une information biaisée modifie la position estimée et le déplacement résultant peut engendrer l'estimation d'une vitesse qui n'a pas lieu. De plus, chaque partage d'informations considère la référence temporelle associée à l'information. Lorsque la synchronisation entre les robots est incorrecte, les références sont erronées et la qualité de l'information résultant de la mise en commun est grandement dégradée.

La réalisation et l'intégration d'un tel système de perception a nécessité le développement de plusieurs modules informatiques qui n'ont pas été présentés dans ce mémoire. De plus, l'architecture logicielle des robots a été grandement modifiée afin de découpler les modules de perception de ceux de cognition. Une interface de contrôle et un module d'acquisition de données ont aussi été développés afin de commander les robots et de valider les algorithmes de perception. La caméra omnidirectionnelle et le système de perception développé présentent une solution très économique offrant de très bonnes performances. Ils ont permis aux robots de l'équipe *Robofoot* de participer à la compétition *Robocup* et de rivaliser avec les meilleurs équipes. À l'aide du système, un robot peut se localiser à plus ou moins 3 cm dans un environnement de 6 m par 9 m ce qui est largement suffisant pour ce type d'application. Il peut mesurer la position du ballon et la partager avec les autres joueurs de son équipe. Au sain de l'équipe, le joueur le moins avantagé est le gardien de but puisqu'il est situé à une extrémité du terrain et que la faible résolution de la caméra l'empêche de distinguer les balises situées de l'autre côté du terrain. Toutefois, grâce au partage d'informations, il est en mesure de connaitre la position du ballon même s'il ne le voit pas.

Tous les paramètres du système de configuration peuvent être facilement modifiés. Le système peut être utilisé dans différents environnements contrôlés en modifiant simplement les paramètres inscrits dans le fichier de configuration. À titre d'exemple, lors d'une démonstration à l'exposition *Robofolies* au centre des sciences de Montréal, le système éprouvait de la difficulté à identifier les balises de couleur bleu à cause des conditions d'éclairage. Pour contourner le problème, il a suffit d'ajouter deux balises rouges à l'environnement et de les enregistrer dans le fichier de configuration. Le

système disposait alors de deux balises supplémentaires pour localiser le robot. Ce système pourrait ainsi être utilisé pour permettre à des robots de se localiser et d'accomplir différentes tâches dans des environnements intérieurs structurées tel des hopitaux ou des ateliers de fabrication.

Au départ, avant la réalisation de ce projet, les robots n'avaient pas de système de vision embarqué. Ils dépendaient d'un système de vision global qui observait la scène et qui retournait aux robots leur position respective. La contribution principale de ce projet est donc la réalisation d'un système embarqué autonome de perception basé sur une caméra omnidirectionnelle permettant la reconnaissance d'objets, la localisation du robot, la perception du ballon et le partage d'informations. La localisation du robot est réalisée par un filtre de *Kalman* qui fusionne les mesures d'odométrie aux mesures de vision afin d'offrir une très grande précision. Les 6 robots de l'équipe *Robofoot* disposent maintenant de ce système simple et complet leur permettant de jouer au soccer et de participer à plusieurs compétitions internationales.

La perception de l'environnement et la localisation d'un robot sont des problèmes qui peuvent être très complexes. Le système présenté dans ce travail fonctionne bien dans un environnement controlé dans la mesure où les hypothèses de fonctionnement sont respectées. Les algorithmes utilisés sont fiables, performants et ils répondent aux exigences établies. Toutefois, il a été constaté que les hypothèses de fonctionnement sont bien souvent difficiles à respecter et elles constituent ainsi les principales limitations du système. Il serait alors intéressant de reconsidérer ces hypothèses et de s'assurer de les respecter parfaitement ou de développer des algorithmes supplémentaires permettant de les rendre moins contraignantes. Les performances du système seraient alors encore plus impressionnantes.

Références

[1] FARGEON, C. et QUIN, J.-P. (1993). *Robotique mobile.* Toulouse [France] : Teknea.

[2] GRABOWSKI, R., NAVARRO-SERMENT, L. E., PAREDIS, C. J. J. et KHOSLA, P. (2000). Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8, 293–308. http://www.ece.stevens-tech.edu/~ymeng/courses/robotics/papers/mapandexplore_MAS_Grabowski.pdf.

[3] WILLGOSS, R., ROSENFELD, V. et BILLINGSLEY, J. *High Precision GPS Guidance of Mobile Robots.* Technical report, UNSW, Sydney 2052, NSW, Australia et USQ, Toowoomba 4350, QLD, Australia. http://www.araa.asn.au/acra/acra2003/papers/25.pdf.

[4] HU, H. et GU, D. (2000). Landmark-based navigation of industrial mobile robots. *International Journal of Industry Robot*, 27(6), 458–467. http://cswww.essex.ac.uk/staff/hhu/Papers/Journal-IR27-6.pdf.

[5] GONZALEZ, J., STENTZ, A. et OLLERO, A. (1992). An iconic position estimator for a 2d laser rangefinder. 2646–2651. IEEE International Conference on Robotics and Automation, 1992. http://www.iut-amiens.fr/~clerentin/CLE00c.pdf.

[6] BETKE, M. et GURVITS, L. (1997). Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2), 251–263.

[7] ADORNI, G., CAGNONI, S., CARLETTI, M. et MORDONINI, M. *Omnidirectional vision algorithms in robotics.* Technical report, Università di Gen-

ova et Università di Parma. http://www-dii.ing.unisi.it/aiia2002/paper/ROBOTICA/cagnoni-aiia02.pdf.

[8] NAYAR, S. K. *Omnidirectional Vision.* Technical report, Department of Computer Science Columbia University New York. http://www.bmva.ac.uk/bmvc/1998/pdf/p021.pdf.

[9] FORSYTH, D. (2003). *Computer vision : a modern approach.* Upper Saddle River, N.J. : Prentice Hall.

[10] GONZALEZ, R. C. (2002). *Digital image processing Second Edition.* Upper Saddle River, N.J. : Prentice Hall.

[11] STEVENS, M. R. et BEVERIDGE, J. R. (2001). *Integrating graphics and vision for object recognition.* Kluwer Academic.

[12] BOW, S.-T. (2002). *Pattern Recognition and Image Processing Second Edition.* Marcel Barker.

[13] PAULUS, D. W. (2001). *Applied Pattern Recognition A Pratical Introduction to Image and Speech Processing in C++ Third Edition.* Vieweg.

[14] RUSS, J. C. (1999). *The Image Processing Handbook Third Edition.* CRC Press LLC.

[15] YOON, K.-J. et KWEON, I.-S. *Artificial Landmark Tracking Based on the Color Histogram.* Technical report, Robotics and Computer Vision Laboratory, Dept. of Electrical Engineering and Computer Science, KAIST, Korea. http://rcv.kaist.ac.kr/~kjyoon/IROS2001.pdf.

[16] SE, S., LOWE, D. et LITTLE, J. *Mobile Robot Localization And Mapping with Uncertainty using Scale-Invariant Visual Landmarks.* Technical report, MD Robotics, Department of Computer Science, University of British Columbia. http://www.cs.ubc.ca/spider/little/links/ijrr02-preprint.pdf.

[17] JANG, G., KIM, S., KIM, J. et KWEON, I. (2005). Metric localization using a single artificial landmark for indoor mobile robots. 3407–3412. IEEE International Conference on Intelligent Robots and Systems,

2005. http://rcv.kaist.ac.kr/publication/file/foreign_conference/88_GiJeongJang_IROS2005.pdf.

[18] WINTERS, N. et SANTOS-VICTOR, J. *Mobile Robot Navigation using Omni-directional Vision.* Technical report, University of Dublin Trinity College, Dublin Ireland et Instituto de Sistemas e Robotica, Lisboa Portugal. http://www.lkl.ac.uk/niall/imvip1999.pdf.

[19] KELLY, A. *Precision Dilution in Triangulation Based Mobile Robot Position Estimation.* Technical report, Robotics Institute Carnegie Mellon University Pittsburgh. http://www.frc.ri.cmu.edu/~alonzo/pubs/papers/pdf_files/ias8.pdf.

[20] ROUX, J. L. (2003). *An introduction to Kalman filter : probabilistic and deterministic approaches.* Technical report, University of Nice. http://www.essi.fr/~leroux/kalmanintroduction.ps.

[21] WELCH, G. et BISHOP, G. (2004). *An Introduction to the Kalman Filter.* Technical report, Department of Computer Science University of North Carolina at Chapel Hill. http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

[22] BIERMAN, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation.* Academic Press.

[23] NAVARRO-SERMENT, L. E., GRABOWSKI, R., PAREDIS, C. J. et KHOSLA, P. K. *Modularity in small distributed robots.* Technical report, Robotics Institute Carnegie Mellon University Pittsburgh. http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/user/cjp/www/pubs/SPIE99.pdf.

[24] BALCH, T. et ARKIN, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1), 27–52. http://www.cc.gatech.edu/ai/robot-lab/tmr/comm.pdf.