Navigation par enchaînement de tâches référencées multi-capteurs
Rapport de stage de DEA

David FOLIO
dfolio@laposte.net

25 juin 2003

Encadré par Viviane CADENAT
au sein du groupe RIA
# Table des matières

Table des figures iii

Remerciements v

Introduction 1

I Le contexte d’étude 3
   I.1 Le robot 4
   I.2 $\text{Gen}^{\text{om}}$: Generator of modules 5
      I.2.1 $\text{Gen}^{\text{om}}$: environnement de développement de module 5
      I.2.2 Fonctionnement des modules 6

II Commande référencée capteurs 9
   II.1 Présentation 9
   II.2 Modélisation 11
      II.2.1 Modèle du robot 11
      II.2.2 Modèle de la caméra 14
      II.2.3 Interaction caméra/environnement 15
   II.3 Synthèse d’une loi de commande référencée vision 17
      II.3.1 Formalisme des fonctions de tâches 17
      II.3.2 La commande référencée vision 18

III Commande référencée multi-capteurs 21
   III.1 Évitement d’obstacles par champ de potentiel 21
      III.1.1 Méthode des potentiels 22
      III.1.2 Méthode des potentiels rotatifs 24
   III.2 Commande référencée multi-capteurs en environnement encombré 27
      III.2.1 Tâche de positionnement dans l’espace cartésien 28
      III.2.2 Tâche référencée vision 30

IV Implémentation et aspect expérimentaux 32
   IV.1 Implémentation des modules 33
      IV.1.1 Le module extract 34
      IV.1.2 Le module VSRP 36
<table>
<thead>
<tr>
<th>Tableau</th>
<th>Titre</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV.2</td>
<td>Résultats expérimentaux</td>
<td>37</td>
</tr>
<tr>
<td>IV.2.1</td>
<td>La tâche de positionnement dans l'espace cartésien</td>
<td>37</td>
</tr>
<tr>
<td>IV.2.2</td>
<td>La commande référencée vision</td>
<td>40</td>
</tr>
<tr>
<td>Conclusion et perspective</td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Références bibliographiques</td>
<td></td>
<td>47</td>
</tr>
</tbody>
</table>
# Table des figures

<table>
<thead>
<tr>
<th>Section</th>
<th>Titre</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>Le SuperScout II</td>
<td>4</td>
</tr>
<tr>
<td>I.2</td>
<td>Génération d'un module $G^{\text{GM}}$</td>
<td>6</td>
</tr>
<tr>
<td>I.3</td>
<td>Structure d'un module $G^{\text{GM}}$</td>
<td>7</td>
</tr>
<tr>
<td>I.4</td>
<td>États et transitions d'une activité</td>
<td>7</td>
</tr>
<tr>
<td>II.1</td>
<td>Modélisation du robot muni d'une caméra montée sur une platine</td>
<td>11</td>
</tr>
<tr>
<td>II.2</td>
<td>Modèle sténopé de la caméra</td>
<td>15</td>
</tr>
<tr>
<td>III.1</td>
<td>Formulation du potentiel-rotation</td>
<td>23</td>
</tr>
<tr>
<td>III.2</td>
<td>Comparaison des méthodes des potentiels classiques et du potentiel rotation</td>
<td>24</td>
</tr>
<tr>
<td>III.3</td>
<td>Évitement d'obstacle avec potentiel rotatif</td>
<td>25</td>
</tr>
<tr>
<td>III.4</td>
<td>Exemple de situation avec un obstacle concave</td>
<td>27</td>
</tr>
<tr>
<td>IV.1</td>
<td>Organisation des modules</td>
<td>34</td>
</tr>
<tr>
<td>IV.2</td>
<td>L'algorithme d'extraction de tâche</td>
<td>35</td>
</tr>
<tr>
<td>IV.3</td>
<td>Tâche de positionnement dans l'espace cartésien seul</td>
<td>38</td>
</tr>
<tr>
<td>IV.4</td>
<td>Tâche de positionnement en présence d'obstacle</td>
<td>39</td>
</tr>
<tr>
<td>IV.5</td>
<td>Evolution du robot en asservissement seul</td>
<td>40</td>
</tr>
<tr>
<td>IV.6</td>
<td>Évolution des indices visuels en asservissement visuel seul</td>
<td>41</td>
</tr>
<tr>
<td>IV.7</td>
<td>Commandes du robot en asservissement visuel seul</td>
<td>41</td>
</tr>
<tr>
<td>IV.8</td>
<td>Évolution du robot en asservissement en environnement contraint</td>
<td>42</td>
</tr>
<tr>
<td>IV.9</td>
<td>Évolution des indices visuels en asservissement visuel en environnement</td>
<td>42</td>
</tr>
<tr>
<td>IV.10</td>
<td>Évolution de $\mu$</td>
<td>43</td>
</tr>
<tr>
<td>IV.11</td>
<td>Données du robot en asservissement seul</td>
<td>43</td>
</tr>
<tr>
<td>IV.12</td>
<td>Commandes du robot en asservissement visuel et évitement d'obstacle</td>
<td>44</td>
</tr>
</tbody>
</table>
Remerciements

Ce rapport présente les travaux que j’ai effectués au sein du groupe RIA (Robotique et Intelligence Artificielle) du LAAS-CNRS dans le cadre de mon stage de DEA. J’exprime ici toute ma reconnaissance à Monsieur Malik GHALLAB, Directeur du LAAS, pour m’avoir accueilli au sein de son laboratoire, ainsi qu’à Monsieur Raja CHATILA, Responsable du groupe RIA, pour m’avoir permis de travailler dans son groupe et donné les moyens de réaliser ce stage.

Ma reconnaissance va aussi à Viviane CADENAT, maître de conférences à l’université Paul Sabatier et responsable de mon stage, pour m’avoir non seulement accueilli, mais aussi et surtout pour m’avoir soutenu et guidé tout au long de ces travaux.

Il m’est très difficile de remercier toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce stage. Je veux néanmoins exprimer toute ma reconnaissance à Michel TAIX et Frédéric LERASLE, tous deux maîtres de conférences à l’université Paul Sabatier, ainsi que Philippe SOUERES, Chargé de Recherches au CNRS, qui m’ont plus particulièrement apporté leur pour la réalisation de mes travaux. Je n’oublie pas Stéphane MAS qui a travaillé avec moi sur les robots SuperScout II.

Lire et juger un rapport de stage n’est pas une tâche facile à accomplir. Aussi, je tiens à remercier les membres de mon jury qui ont bien voulu porter un jugement sur mon travail, et plus particulièrement Monsieur Robert VALETTE, Directeur de Recherche au CNRS et responsable du DEA Systèmes Informatiques, pour son aide et sa disponibilité à mes questions. J’adresse également dans une plus large mesure mes remerciements à toutes les personnes qui, au sein de l’École Doctorale Systèmes, ont eu la patience de me former à la recherche durant cette année de DEA.

Enfin, un merci tout particulier à Sara et Anthony pour leurs conseils éclairés sur l’utilisation de G²M², ainsi qu’à tous ceux avec qui j’ai partagé ces quelques mois de stage : Jib, Delphine, Claudia, Nadhem, Kaouther… Je remercie également toute l’équipe du groupe RIA dont l’ambiance a donné une autre dimension à ce travail.

Enfin, mes salutations vont à tous les autres membres du DEA ainsi qu’aux autres stagiaires et thésard du groupe à qui je souhaite bon courage et bonne continuation pour la suite.
À la mémoire de mon pépé...
L’un des plus grands rêves de l’Homme est de pouvoir faire exécuter par des acteurs autres que lui-même des tâches ou activités qu’il considère comme aliénantes, dangereuses ou simplement ennuyeuses et non gratifiantes. L’apparition progressive de machines puis de la notion d’automatisation a permis de modifier la nature des travaux effectués par les hommes, donc de déporter cette aspiration vers des réponses issues des progrès des techniques et des sciences.

Ainsi, l’émergence de la robotique au cours des années 1960 a su apporter de nouvelles réponses à cette attente. Avec les progrès constants dans ce domaine, les robots tendent à accomplir des tâches complexes avec une intervention moindre de l’homme pour les guider. Ils deviennent ainsi plus autonomes, et ils interagissent de mieux en mieux avec leur environnement pour accomplir la mission qui leur a été assignée. Le robot mobile devient donc une « machine intelligente ». Ceci sous-entend que cette machine est capable de percevoir l’environnement qui l’entoure et de raisonner sur les données acquises par ses capteurs. Sur cette base, il est capable de planifier les déplacements nécessaires en vue de l’accomplissement de sa mission. Il reste alors à synthétiser la loi de commande permettant d’exécuter les mouvements planifiés. C’est pourquoi, le problème de la commande des robots mobiles qui sera développé ici, suscite l’intérêt des chercheurs en Automatique depuis plusieurs décennies. En effet, classiquement, les lois de commande sont posées en terme de régulation à zéro d’une erreur entre la situation du robot par rapport à un référentiel lié à l’environnement, et une situation de référence définie a priori. Ces asservissements sont, de ce fait, généralement de type retour d’état continu et nécessitent alors la détermination de l’état du robot et donc sa reconstruction par une étape de localisation. Une autre approche consiste à exprimer
la tâche de navigation du robot comme une succession de tâches consistant à réguler des données capteurs locales : la commande référencée capteurs. L’originalité de cette méthode est d’essayer de se détacher le plus possible de la représentation métrique euclidienne du monde, au bénéfice d’une représentation définie en termes de données sensorielles.

Les travaux menés ici s’inscrivent dans le cadre de cette dernière démarche. Plus précisément, nous nous sommes intéressés à la réalisation de tâches de navigation guidées par la vision dans un environnement encombré. L’étude que nous nous proposons de développer se situe donc dans la problématique de l’implémentation de lois de commande référencées multi-capteurs. Les lois proposées doivent pouvoir s’intégrer à des applications existantes, plus particulièrement elles devront être insérées dans une application logicielle, GensM (voir § 1.2), embarquée sur un robot mobile réel : le SuperScout II.

Après avoir présenté le contexte d’étude, nous présenterons le formalisme de la commande référencée capteurs, à travers la réalisation d’une tâche de navigation guidée par la vision. Puis, nous nous intéresserons à la commande référencée multi-capteurs, qui s’avère nécessaire pour intégrer les fonctionnalités d’évitement d’obstacle requises pour une tâche de navigation en environnement encombré.
Chapitre I
Le contexte d’étude


Les recherches menées au sein du groupe RIA portent sur l’étude et la conception de machines autonomes, intégrant des capacités de perception, d’action et de raisonnement. Ces machines doivent interagir rationnellement avec un environnement variable et évolutif pour la réalisation de tâches diverses. Parmi les thématiques de recherche abordées par le groupe, une attention particulière est donnée à la commande des robots mobiles autonomes. En effet, cette dernière pose un problème de recherche très riche dont l’une des difficultés provient des contraintes non-holonomes de roulement sans glissement qui interdisent les déplacements latéraux instantanés. Cette problématique intervient sur bon nombre d’architectures de robot. À cet effet, pour l’intégration des lois de commande développées au cours de notre étude, nos expérimentations ont été menées sur un robot mobile de type char, le «Nomadic SuperScout II».

En outre, pour la mise en œuvre logicielle, nous utilisons un cadre logiciel particulier appelé GenM pour encapsuler nos lois de commande, afin de pouvoir les embarquer à bord du robot. De ce fait nous décrirons dans un premier temps la plateforme matérielle utilisée au cours de nos expérimentations, avant d’aborder la partie logicielle, en détaillant les principes de GenM.

[^1]: [http://www.laas.fr](http://www.laas.fr)
1.1 Le robot

Les lois de commande qui seront présentées ici ont été implémentées sur des SuperScout II, développés par la société Nomadic Technologies (cf figure I.1).

Le SuperScout II possède une base mobile de type char, et il est principalement destiné à une utilisation en intérieur. Son architecture est circulaire, de 35cm de diamètre, et il pèse 25kg environ. Il est également doté d’une ceinture de seize capteurs ultrasoniques, de contacteurs (bumpers) et d’odomètres. Il est constitué d’un système de contrôle multiprocesseur bas niveau qui commande à la fois les capteurs, le mouvement ainsi que la communication. À un niveau plus élevé, le scout peut être commandé soit par un PC-portable embarqué, soit par une station de travail communiquant via un modem radio.

En outre, la plate-forme dont nous disposons pour notre étude a été équipée d’une caméra permettant d’extraire les informations visuelles pertinentes. De nouvelles caméras viennent d’être installées. Il s’agit de caméras Sony DFW-VL 500, numériques couleur supportant la norme IEEE1394. Cette norme, aussi connue sous le nom de FireWire (Apple) ou i-LINK (Sony), a été créée à la fin des années 1980 pour satisfaire les besoins de plus en plus importants en largeur de bande de la communication vidéo. Elle s’annonce comme étant le futur protocole de communication vidéo, tant pour les applications domestiques (TV, DVR, camescopes numériques, Hi-fi, ordinateurs...) que pour des applications industrielles et scientifiques.

Comme nous le verrons plus loin, la mise en œuvre de commandes référencées vision nécessite que la caméra puisse se déplacer indépendamment de la base mobile. C’est pourquoi, cette dernière a été montée sur une platine orientable en site et azimut, commandable aussi
I.2. G\textsuperscript{en}oM : Generator of modules

G\textsuperscript{en}oM est un générateur de modules, élaboré au LAAS/CNRS par Sara Fleury \cite{Fleury 96, Fleury & Herrb 01}. Ainsi, il propose une interface standardisée de génération de modules, qui permet l’encapsulation de bibliothèques de fonctions (nos algorithmes). C’est à la fois un outil et un environnement de développement pour la conception de logiciels modulaires. Il est en particulier utilisé pour la mise en œuvre d’applications sur des systèmes embarqués complexes, tels que les robots mobiles, les satellites… ou tout système ayant des contraintes temps-réel distribuées.

La modularité permet à G\textsuperscript{en}oM de faciliter l’intégration de fonctions hétérogènes, comme la lecture de données capteurs, la commande des actionneurs, les boucles d’asservissement, le calcul de trajectoire… Elle permet aussi à l’ensemble d’avoir une homogénéité qui assure sa cohérence et sa prédictibilité, notamment pour le contrôle de processus (démarrage, interruption, reprise d’erreur…) ainsi que l’interaction avec le superviseur (contrôle, paramétrisation, communication de données…).

I.2.1 G\textsuperscript{en}oM : environnement de développement de module

G\textsuperscript{en}oM permet de définir un module générique et des protocoles de communication. Ce module générique peut intégrer à peu près n’importe quel type de traitement (périodique/apériodique, synchrone/asynchrone…). Ainsi, G\textsuperscript{en}oM génère le module automatiquement, en s’appuyant sur le modèle du module générique, et sur une description synthétique du module à générer. Cette description, incluse dans le fichier d’extension ‘.gen’, comprend la déclaration et la description des services offerts, des paramètres, des données importées et exportées. G\textsuperscript{en}oM produit alors un module exécutable\footnote{L’exécutable est produit pour le système d’exploitation spécifié : i386-linux, sparc-solaris, m68k-VxWorks…}, des bibliothèques d’interface pour l’accès aux services et aux posters, et des outils pour la supervision, le contrôle et la validation expérimentale du module.

De ce fait, le rôle du développeur se limite donc à la description du module en vue de sa génération et l’implémentation effective des services. Il intègre alors ses algorithmes à l’intérieur d’unités élémentaires de programme, appelés codels\footnote{Les codels sont des fonctions dont le prototype est uniformisé, produites lors de la génération du module que l’utilisateur devra compléter avec ses algorithmes.} (codes élémentaires). Le module est alors le résultat de l’édition de liens entre la partie générée par G\textsuperscript{en}oM (située dans le répertoire ’auto’) et les bibliothèques de codels (située dans le répertoire ’codels’) développées par l’utilisateur. La figure I.2 illustre les différentes étapes de la génération d’un module.
I.2.2 Fonctionnement des modules

L’accès aux différents services offerts par un module s’effectue par l’envoi de requêtes qui permettent de démarrer dynamiquement les fonctions, de les interrompre ou encore de les paramétrer. Les requêtes peuvent être émises par n’importe quel élément du système (un autre module, une autre application, un opérateur, un superviseur...). L’ensemble des services qui s’exécutent à bord du système est donc suivi en permanence par un flux de contrôle composé des requêtes et des réponses associées, ces dernières étant appelées répliques.

De ce fait, un module peut être vu comme un serveur qui propose un certain nombre de services ainsi que des accès à des données. Les données de chaque module sont exportées dans une base de données externes, nommées posters, gérée par un flux de données, indépendamment du flux de contrôle. Ces posters sont accessibles en lecture par tout élément de l’architecture de l’application : les autres modules, mais également l’opérateur ou un superviseur. Par ailleurs, les données partagées à l’intérieur d’un module sont stockées dans une structure de données interne, ou SDI, qui se décompose en deux bases de données distinctes :

1. Une Structure de Données Interne de Contrôle, ou SDI_C, qui regroupe l’ensemble des informations ayant trait au contrôle des activités et à l’état global du module (événements, états des activités, temps d’exécution, bilans...).

2. Une Structure de Données Interne Fonctionnelle, ou SDI_F, qui contient les données fonctionnelles dépendant des services offerts. Elle comporte les paramètres des requêtes, les résultats des traitements, et les données échangées entre les activités.

En outre, le module, entité logicielle indépendante mais communiquante, est constitué de tâches de contrôle et d’au moins une tâche d’exécution :

− La tâche de contrôle a un rôle pivot, elle doit réagir aux requêtes des clients et transmettre le bilan des traitements ainsi que les éventuels résultats.

− La tâche d’exécution, quant à elle, permet de traiter les algorithmes développés (à l’intérieur des codels), afin de mettre en œuvre les services offerts par le module. La figure I.3 détaille l’architecture interne d’un module.

De plus, comme nous l’avons déjà évoqué, nos algorithmes sont encapsulés dans des codels. Pour une bonne efficacité des traitements, il convient de décomposer les codels en différentes sous-fonctions correspondant aux étapes du traitement. Il est important de noter qu’un codel est la plus petite entité manipulable par le module. En particulier, un codel est
I.2. \textit{G^enO}_M : Generator of modules

... 

\textit{ininterruptible}. Ainsi pendant l’exécution d’un codel, le module ne peut rien faire d’autre. Pour déterminer la décomposition d’un traitement, il faut avoir à l’esprit le cycle de vie général d’une activité représenté sur la figure I.4.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{structure_module.png}
\caption{Structure d’un module \textit{G^enO}_M}
\end{figure}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
État & Description \\
\hline
\hline
START & phase de démarrage \\
EXEC & phase d’exécution principale \\
END & phase de terminaison \\
FAIL & phase de terminaison suite à une défaillance grave \\
INTER & phase de terminaison suite à une interruption \\
SLEEP & activité suspendue \\
ETHER & activité terminée \\
ZOMBIE & activité terminée et module suspendu \\
\hline
\end{tabular}
\caption{Description des différents états}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{etat_trame.png}
\caption{États et transitions d’une activité.}
\end{figure}

I.4.a : Graphe d’état.

I.4.b : Description des différents états
Le fonctionnement est alors le suivant. À chaque cycle du traitement, la tâche de contrôle se réveille, et fait évoluer l’état de chaque activité en cours de fonctionnement, comme indiqué sur la figure 1.4.a. Pour cela, chaque codel renvoie une variable d’état qui correspond à l’état du prochain cycle du traitement. Ainsi, lors de l’envoi d’une requête d’exécution pour démarrer une activité, la tâche de contrôle met à jour la SDI_C de manière à pointer sur la transition start/-. L’activité se trouve alors dans l’état START, et le codel associé sera lancé – s’il est défini – et renverra la variable d’état indiquant la prochaine transition, le plus souvent exec/- dans la mesure où l’état EXEC permet de traiter le codel d’exécution où sont écrits les algorithmes associés à la requête.

\[4\] Dans le cas contraire la SDI_C pointe sur la transition suivante.
Chapitre II
Commande référencée capteurs

II.1 Présentation

La capacité de navigation dans un environnement encombré constitue l’une des facultés indispensables à l’autonomie d’un robot mobile. Ce processus de navigation se décompose, traditionnellement, en plusieurs phases :

1. Perception : selon l’objectif fixé et les difficultés rencontrées, le système de navigation détermine les informations nécessaires pour l’accomplissement de la tâche ;
2. Modélisation : le robot procède à l’analyse de ces données perceptuelles afin de produire une représentation interprétable de l’environnement ;
3. Localisation : pour décider des déplacements, le robot doit déterminer sa situation (i.e. sa position et son orientation) par rapport au modèle produit ;
4. Planification : le robot décide de l’itinéraire, puis détermine la trajectoire et les conditions de son bon accomplissement ;

Il est nécessaire ensuite d’asservir le robot mobile sur les déplacements qu’il doit réaliser en vue de l’accomplissement de sa mission. La littérature regorge de méthodes concernant la commande des robots mobiles non-holonomes. Celle-ci se pose souvent en terme de régulation à zéro d’une erreur entre une situation de référence décrite par rapport à un repère global, et la situation courante du robot. Si les techniques classiques de commande non-linéaire offrent des solutions pour des problèmes de suivi de chemin ou de trajectoire, il est en revanche plus complexe d’asservir un robot en une configuration prédéfinie, du fait de la contrainte de roulement sans glissement des roues sur le sol.

En outre, les lois de commande sont généralement de type retour d’état et nécessitent la détermination de l’état du robot. Pour cela, il faut pouvoir reconstruire l’état du robot en se basant sur les informations sensorielles. On distingue deux types de capteurs sensoriels :
Chapitre II. Commande référencée capteurs

– les capteurs proprioceptifs, comme les odomètres ou gyroscopes qui permettent d’avoir une information sur l’état interne du robot ;
– les capteurs extéroceptifs comme les caméras, ultrasons, laser… qui permettent de connaître la situation du robot par rapport à son environnement.

Cependant, on ne peut espérer localiser précisément un robot en se basant uniquement sur les données proprioceptives, du fait, entre autres, des problèmes de dérives des capteurs. De plus ces données ne permettent pas de tenir compte des contraintes liées à l’environnement, ce qui peut poser problème dans l’exécution de certaines tâches, comme l’évitement d’obstacles non prévus lors de la planification. L’exploitation des données extéroceptives permet d’avoir une vision plus précise du véhicule dans son environnement.

La commande référencée capteurs permet, justement, de s’affranchir de cette étape de localisation. Elle consiste à exprimer les tâches robotiques à réaliser, non plus dans l’espace des configurations, mais directement dans l’espace du capteur sous la forme d’une relation locale entre le robot et son environnement. On cherche ainsi à définir des commandes en boucle fermée sur les informations provenant des capteurs et non plus sur la configuration du robot. De ce fait les lois de commande généralement synthétisées dans le cadre de cette approche sont de type retour de sortie et permettent de réaliser les tâches désirées de manière plus précise. Ainsi, toute tâche pouvant s’exprimer sous la forme d’une relation entre le robot et l’environnement sera réalisable par le biais de la commande référencée capteurs. Toutefois, cette approche requiert l’utilisation de capteurs performants, capables de fournir les mesures à une fréquence compatible avec la fréquence d’échantillonnage de l’asservissement. Parmi les capteurs extéroceptifs existants, la vision a plus particulièrement retenu l’attention des roboticiens du fait de la richesse des informations visuelles et de l’efficacité d’extraction de primitives simples. La commande référencée capteurs devient alors la commande référencée vision.

Il existe différentes approches de commande référencée vision. L’une d’elle repose sur la régulation de la situation de la caméra par rapport à son environnement. La tâche est alors décrête en termes d’une situation à atteindre entre la caméra et l’environnement. Une autre approche, considérée ici, est basée sur la régulation à zéro d’une erreur dans l’image. En d’autres termes, le but à atteindre est de faire coïncider des indices visuels relevés au cours de la tâche vers un motif de référence correspondant à la situation but. La commande est alors synthétisée sur la base des informations visuelles contenues dans l’image.

Rappelons que le robot considéré, le SuperScout II, est de type char. Il s’agit donc d’un véhicule non-holonome qui doit parfois, pour effectuer certaines tâches de navigation, réaliser des manœuvres. Or, dans le cas particulier de la commande référencée vision, les indices visuels doivent toujours rester dans le champ de vue de la caméra pour garantir une exécution sans échec de la tâche considérée. Cette condition peut ne pas être satisfaite si la caméra est rigidement liée à la base mobile lorsque celle-ci effectue des manoeuvres particulières. Aussi, pour pallier à ce problème, est-il nécessaire d’introduire au moins un degré de liberté supplémentaire permettant à la caméra de se déplacer indépendamment de la base. D’où l’introduction d’une platine commandable en lacet.

Ainsi, dans un premier temps, nous nous proposons d’établir la modélisation du robot muni d’une caméra montée sur une platine orientable en lacet, ainsi que le modèle pinhole de la caméra. Puis nous présenterons le formalisme des fonctions de tâche [Samson et al., 91]
II.2 Modélisation

II.2.1 Modèle du robot

Comme nous l’avons préalablement mentionné, nos expérimentations ont été effectuées sur un SuperScout II, un robot de type char, équipé d’une platine commandable en lacet.

![Diagramme du robot](image)

**Fig. II.1** – Modélisation du robot muni d’une caméra montée sur une platine.

En considérant la figure II.1, nous définissons :
- $O$ l’origine de la scène, et $R_0(O, X_0, Y_0, Z_0)$ le repère global lié à la scène ;
- $M$ le centre de la base mobile, et $R_M(M, X_M, Y_M, Z_M)$ le repère lié à la base mobile ;
- $P$ le centre de rotation de la platine, et $R_{pl}(P, X_{pl}, Y_{pl}, Z_{pl})$ le repère lié à la platine, et $D_x$ la longueur de l’entraîne $[MP]$ ;
- $C$ le centre optique de la caméra, et $R_C(C, X_C, Y_C, Z_C)$ le repère lié à la caméra ;

La configuration de la base mobile est décrite par $Q = \begin{pmatrix} x & y & \theta \end{pmatrix}^T$, où le couple $(x, y)$ désigne les coordonnées du centre $M$ de la base mobile dans le repère global lié à la scène $R_0$, et $\theta$ représente l’orientation du robot par rapport à l’axe $X_0$. Enfin, $\theta_{pl}$ définit l’orientation entre $R_{pl}$ (i.e. la platine) et $R_M$ (i.e. la base mobile). Nous définissons alors le vecteur $q$ suivant :

$$q = \begin{pmatrix} l & \theta & \theta_{pl} \end{pmatrix}^T \quad (II.1)$$

où $l$ représente l’absisse curviligne du point $M$. Si nous dérivons $q$ par rapport au temps, nous obtenons :

$$\dot{q} = \frac{dq}{dt} = \begin{pmatrix} v \\ \omega \\ \omega_{pl} \end{pmatrix} \quad (II.2)$$

où $v, \omega$ désignent respectivement les vitesses linéaire et angulaire de la base mobile, $\omega_{pl}$ correspondant à la vitesse de rotation de la platine. Notre robot étant commandable en vitesse, $\dot{q}$ sera notre vecteur de commande.
Modèle cinématique de la base mobile

Nous pouvons d’ores et déjà établir le modèle cinématique de la base mobile qui se définit classiquement par les relations suivantes :

\[
\begin{align*}
\dot{x} &= v \cos \theta \\
\dot{y} &= v \sin \theta \\
\dot{\theta} &= \omega
\end{align*}
\]

Le torseur cinématique de la base mobile se déduit simplement par :

\[
T_{M/R_0} = \left( \begin{array}{c} V_{M/R_0} \\ \Omega_{M/R_0} \end{array} \right)
\]

et s’exprime comme suit dans le repère \(R_M\) :

\[
T_{M/R_0}^{R_M} = \left( \begin{array}{c} V_{M/R_0} \\ \Omega_{M/R_0} \end{array} \right) = \left( \begin{array}{c} (v \ 0 \ 0)^T \\ (0 \ 0 \ \omega)^T \end{array} \right)
\]

Modèle cinématique de la caméra

Comme nous nous intéressons à la commande référencée vision, notre but est de pouvoir exprimer les mouvements effectués par la caméra, en fonction des commandes \(v, \omega\) et \(\omega_{pl}\). Pour cela, il est nécessaire de déterminer le torseur cinématique de la caméra par rapport au repère de la scène, exprimé dans le repère de la caméra. On dispose pour cela du résultat classique de mécanique :

\[
T_{C/R_0} = \left( \begin{array}{c} V_{C/R_0} \\ \Omega_{C/R_0} \end{array} \right) = \left( \begin{array}{c} V_{C/R_{pl}} + V_{P/R_{pl}} + \Omega_{R_{pl}/R_{M}} \wedge \overrightarrow{MC}_{R_{pl}} \\ \Omega_{C/R_{pl}} + \Omega_{R_{pl}/R_{M}} \wedge \overrightarrow{PC}_{R_{pl}} \end{array} \right) \quad (II.3)
\]

Tout d’abord, on se propose d’exprimer le torseur de cinématique de la caméra relativement au repère \(R_M\) :

\[
T_{C/R_{M}} = \left( \begin{array}{c} V_{C/R_{M}} \\ \Omega_{C/R_{M}} \end{array} \right) = \left( \begin{array}{c} V_{C/R_{pl}} + V_{P/R_{pl}} + \Omega_{R_{pl}/R_{M}} \wedge \overrightarrow{PC} \\ \Omega_{C/R_{pl}} + \Omega_{R_{pl}/R_{M}} \end{array} \right) \quad (II.4)
\]

sachant que \(V_{C/R_{pl}} = \frac{d\overrightarrow{PC}}{dt} \bigg|_{R_{pl}}\) et \(V_{P/R_{pl}} = \frac{d\overrightarrow{MP}}{dt} \bigg|_{R_{pl}}\) sont toutes deux nulles puisque les points \(C\) et \(P\) sont fixes respectivement dans les repères \(R_{pl}\) et \(R_M\). De même, comme la caméra est rigide ment liée à la platine, le repère \(R_C\) est également fixe dans le repère \(R_{pl}\). L’équation (II.4) se réduit donc à :

\[
T_{C/R_{M}} = \left( \begin{array}{c} \Omega_{R_{pl}/R_{M}} \wedge \overrightarrow{PC} \\ \Omega_{R_{pl}/R_{M}} \end{array} \right)
\]

Le moment de rotation de la caméra par rapport à la base mobile \((\Omega_{R_{pl}/R_{M}})\), s’exprime de la manière suivante :

\[
\Omega_{R_{pl}/R_{M}}^{R_{M}} = \left( \begin{array}{c} 0 \ 0 \ \omega_{pl} \end{array} \right)^T
\]

Le vecteur \(\overrightarrow{PC}\) définit les coordonnées du centre optique de la caméra\(^1\) dans le repère \(R_{pl}\) et s’écrit :

\[
\overrightarrow{PC}_{R_{pl}} = \left( \begin{array}{c} a \\ b \\ c \end{array} \right)^T
\]

\(^1\) Ces coordonnées sont obtenues par un calibrage *hand-eye.*
Dans un premier temps, pour des raisons de simplicité, nous nous proposons de travailler dans le repère $R_M$. Puis nous reviendrons au repère $R_C$ par le biais de la matrice de passage du repère $R_M$ au repère $R_C$ ($\mathcal{P}_{R_M R_C}$) qui s’exprime de la manière suivante :

$$\mathcal{P}_{R_M R_C} = \mathcal{P}_{R_M R_{pl}} \times \mathcal{P}_{R_{pl} R_C}$$

où $\mathcal{P}_{R_M R_{pl}}$ et $\mathcal{P}_{R_{pl} R_C}$ sont respectivement les matrices de passage de $R_M$ à $R_{pl}$ et $R_{pl}$ à $R_C$. Elles sont données par :

$$\mathcal{P}_{R_M R_{pl}} = \begin{pmatrix}
\cos \theta_{pl} & -\sin \theta_{pl} & 0 \\
\sin \theta_{pl} & \cos \theta_{pl} & 0 \\
0 & 0 & 1
\end{pmatrix}$$

$$\mathcal{P}_{R_{pl} R_C} = \begin{pmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
-1 & 0 & 0
\end{pmatrix}$$

(II.5)

Ainsi, le vecteur $\overrightarrow{PC}$ se projette dans le repère de la base mobile, comme indiqué ci-après :

$$\overrightarrow{PC}_{R_M} = \mathcal{P}_{R_M R_{pl}} \times \overrightarrow{PC}_{R_{pl}}$$

Nous pouvons alors déduire l’expression du torseur cinématique de la caméra relativement au repère $R_M$ :

$$\mathcal{T}_{C/R_M}^{R_M} = \begin{pmatrix}
V_{C/R_M}^{R_M} \\
\Omega_{RC/R_M}^{R_M}
\end{pmatrix} = \begin{pmatrix}
(-a \cdot \sin \theta_{pl} - b \cdot \cos \theta_{pl}) \cdot \omega_{pl} \\
(a \cdot \cos \theta_{pl} - b \cdot \sin \theta_{pl}) \cdot \omega_{pl} \\
0 \\
0 \\
0 \\
\omega_{pl}
\end{pmatrix}$$

(II.6)

Ainsi, nous pouvons à présent déterminer le moment de rotation de la caméra par rapport à la scène, $\Omega_{RC/R_0}$, en projetant $\Omega_{RC/R_M}$ dans le repère de la caméra : $\Omega_{RC/R_M}^{R_C} = \begin{pmatrix} -\omega_{pl} & 0 & 0 \end{pmatrix}^T$, et par le biais des équations (II.3) et (II.6) nous pouvons déduire :

$$\Omega_{RC}^{R_C} = \begin{pmatrix}
-(\omega + \omega_{pl}) \\
0 \\
0
\end{pmatrix}$$

Il nous reste à présent à déterminer entièrement le vecteur vitesse $V_{C/R_0}$. Si nous reprenons la relation (II.3), nous obtenons en projetant dans le repère de la caméra :

$$V_{C/R_0}^{RC} = V_{M/R_0}^{RC} + \Omega_{RC}^{R_C} \wedge \left( \mathcal{M}_{PM}^{RC} + \overrightarrow{PC}_{RC} \right)$$

avec $\mathcal{M}_{PM}^{R_M} = \begin{pmatrix} D_x & 0 & 0 \end{pmatrix}^T$. Il vient :

$$V_{C/R_0}^{RC} = V_{M/R_0}^{RC} + \Omega_{RC}^{R_C} \wedge \left( \mathcal{P}_{RC/R_M} \times \left( \mathcal{M}_{PM}^{R_M} + \overrightarrow{PC}_{R_M} \right) \right)$$

Cette matrice de passage est aussi appelée matrice des paramètres extrinsèques de la caméra.
Finalement, tous calculs faits, le torseur cinématique de la caméra par rapport au repère de la scène $R_0$ exprimé dans le repère $R_C$, s’écrit comme suit :

$$\mathbf{T}^{R_C}_{C/R} = \begin{pmatrix} 0 & 0 & 0 \\ -\sin \theta_{pl} & a + D_x \cdot \cos \theta_{pl} & a \\ -\cos \theta_{pl} & -b + D_x \cdot \sin \theta_{pl} & -b \\ 0 & -1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} v \\ \omega \\ \omega_{pl} \end{pmatrix} = J \times \dot{q} \quad \text{(II.7)}$$

où $J$ représente la jacobienne qui lie le mouvement de la caméra aux vitesses de la base mobile et de la platine. Notons que cette jacobienne comporte trois lignes de zéro, correspondant aux mouvements non admissibles de la caméra. En effet, celle-ci ne peut se déplacer en translation sur l’axe $X_C$ et ne peut pivoter autour des axes $Y_C$ et $Z_C$. Ces résultats sont donc conformes à la géométrie du robot. Cela nous permet alors de réduire le torseur cinématique de la caméra, en ne considérant que les degrés de liberté commandables de cette dernière :

$$\mathbf{T}_{\text{red}} = \begin{pmatrix} -\sin \theta_{pl} & a + D_x \cdot \cos \theta_{pl} & a \\ -\cos \theta_{pl} & -b + D_x \cdot \sin \theta_{pl} & -b \\ 0 & -1 & -1 \end{pmatrix} \times \begin{pmatrix} v \\ \omega \\ \omega_{pl} \end{pmatrix} = J_{\text{robot}} \times \dot{q} \quad \text{(II.8)}$$

où $J_{\text{robot}}$ représente la jacobienne réduite. En outre, comme son déterminant est non nul, $\det(J_{\text{robot}}) = D_x \neq 0$, $J_{\text{robot}}$ est une matrice régulière. La contrainte de non-holonomie n’intervient donc plus dans le mouvement de la caméra.

### II.2.2 Modèle de la caméra

Avant de présenter plus précisément la commande référencée vision, il est nécessaire de s’attarder sur le modèle de la caméra et sur les interactions entre la caméra et l’environnement.

Nous considérons dans l’ensemble de notre étude le modèle de caméra dit « sténopé » (pinhole camera model), présenté sur la figure [II.2]. Nous définissons alors $C$ comme étant le centre optique de la caméra, et $f$ la distance focale. Le modèle sténopé prend pour hypothèse que tous les rayons passent par un seul point : le centre optique $C$. De ce fait, les points sont projetés sur le plan image par une projection perspective. Ainsi, un point $A$ de la scène, de coordonnées $(x_c, y_c, z_c)^T$, exprimé dans le repère caméra $R_C$, est projeté en $A_i$, $(x_i, y_i)^T$ en coordonnées métriques, sur le plan image selon l’expression suivante en coordonnées homogènes :

$$\begin{pmatrix} z_c x_i \\ z_c y_i \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}$$

3On suppose que la focale, ainsi que tous les paramètres intrinsèques, sont constants au cours de l’exécution de la tâche de navigation.

4Ce modèle est acceptable si on utilise des focales faibles.
II.2 Modélisation

De plus, nous avons la relation de transformation caméra/image qui nous permet d’exprimer en coordonnées pixel la position d’un point de l’image :

\[
\begin{align*}
    u &= k_u \cdot x_i + u_0 \\
    v &= k_v \cdot y_i + v_0
\end{align*}
\]

où \( k_u \) et \( k_v \) représentent respectivement la taille des pixels selon les lignes et les colonnes, \( u_0 \) et \( v_0 \) correspondent aux coordonnées de la projection orthogonale du centre optique \( C \) dans le plan image. En introduisant la matrice des paramètres intrinsèques, nous pouvons exprimer les coordonnées pixel \((u, v)\) du point \( A_i \) de l’image à partir du point \( A \) dans \( R_C \), selon la relation suivante :

\[
\begin{pmatrix}
    z_c \alpha_u & 0 & u_0 & 0 \\
    0 & z_c \alpha_v & v_0 & 0 \\
    0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
    x_c \\
    y_c \\
    z_c \\
    1
\end{pmatrix}
\]

(II.9)

où \( \alpha_u = k_u f \) et \( \alpha_v = k_v f \). De plus, \( (u_0, v_0, \alpha_u, \alpha_v) \) sont appelés paramètres intrinsèques de la caméra.

A présent, nous sommes en mesure de lier la position des points de la scène exprimés dans le repère caméra et les mesures effectuées dans l’image. De plus, en utilisant les matrices de passages (II.5) définies précédemment nous pouvons projeter les points exprimés directement dans le repère de la scène dans le plan image.

II.2.3 Interaction caméra/environnement

Comme nous l’avons déjà mentionné, la commande référencée vision repose sur la régulation à zéro d’une erreur dans l’image. Nous posons néanmoins l’hypothèse que seuls les déplacements de la caméra sont susceptibles de faire évoluer les indices visuels mesurés \( i.e. \) la cible est fixe). Cependant, comme le robot muni de sa caméra se déplace dans son environnement, il est nécessaire de déterminer la manière dont l’information visuelle évolue.
lorsque la caméra se déplace. C’est pourquoi nous introduisons la notion de matrice d’inter-
action. Celle-ci permet de relier les variations des informations sensorielles, notées ici $s$, 
aux torseur cinématique de la caméra $T_{C/R_0}^{RC}$ donné par (II.7). Ainsi, la variation de ce signal $\dot{s}$ est définie de la façon suivante :

$$\dot{s} = L \times T_{C/R_0}^{RC}$$

(II.10)
où $L$ désigne la matrice d’interaction. Elle dépend non seulement de la géométrie des indices visuels considérés mais aussi de l’information de profondeur, c’est-à-dire la distance entre la caméra et les indices.

Pour nos travaux, nous considérerons une cible carrée composée de quatre points. Le vecteur des indices visuels $s$ comprend donc de 8 composantes :

$$s = \left( x_1 \ y_1 \ x_2 \ y_2 \ x_3 \ y_3 \ x_4 \ y_4 \right)^T$$

(II.11)
où le couple $(x_j, y_j)$ représente les coordonnées du $j^{\text{ième}}$ point de la cible projeté dans le plan image de la caméra. Dans ce cas, la matrice d’interaction est définie de manière générale par Chaumette 90 comme suit :

$$L = \begin{pmatrix} L_1 \\ \vdots \\ L_j \\ \vdots \\ L_n \end{pmatrix}$$

où $n = 8$ correspond aux nombres de composantes de $s$, et $L_j$ est définie pour chaque couple $(x_j, y_j)$ de la manière suivante :

$$L_j = \begin{pmatrix} -\frac{1}{z_j} & 0 & \frac{x_j}{z_j} & x_jy_j & -\left(1 + x_j^2\right) & y_j \\ \frac{y_j}{z_j} & 1 + y_j^2 & -x_jy_j & -x_j \end{pmatrix}$$

Cette expression de la matrice d’interaction $L$ suppose que le torseur cinématique soit de dimension $(6 \times 1)$. Or, nous avons défini un torseur cinématique réduit $T_{\text{red}}$ de dimension $(3 \times 1)$ ne contenant que les degrés de liberté réellement commandables. De ce fait, il est possible de rendre cette matrice compatible en ne conservant que les colonnes correspondant à ces degrés de liberté. Nous obtenons alors la matrice d’interaction réduite suivante :

$$L_{\text{robot},j} = \begin{pmatrix} 0 & \frac{x_j}{z_j} & x_jy_j \\ -\frac{1}{z_j} & \frac{y_j}{z_j} & 1 + y_j^2 \end{pmatrix}$$

(II.12)

Remarque 1 Notons que le calcul de la matrice d’interaction nécessite l’estimation de la profondeur $z_j$ à chaque pas de la boucle de commande. Dans la mesure où cette information n’est pas toujours disponible, nous considérons la matrice d’interaction à l’équilibre, notée $L^\ast_{\text{pl}}$, c’est-à-dire lorsque $s = s^\ast$, où $s^\ast$ est le signal de référence Chaumette 90.
II.3 Synthèse d’une loi de commande référencée vision

Ainsi à l’aide des résultats obtenus précédemment (II.8,II.10) nous pouvons exprimer directement la variation des indices visuels en fonction des vitesses du robot, c’est-à-dire en fonction de notre vecteur de commande :

\[
\dot{s} = L_{\text{robot}} \times J_{\text{robot}} \times \dot{q}
\]

(II.13)

II.3 Synthèse d’une loi de commande référencée vision

Rappelons que notre objectif est de synthétiser une loi de commande permettant de piloter les mouvements de la caméra de manière à assurer la convergence des indices visuels s vers un motif de référence s*. Pour cela, nous nous basons sur le formalisme des fonctions de tâche.

II.3.1 Formalisme des fonctions de tâches

Le formalisme des fonctions de tâches a été introduit par [Samson et al. 91] pour définir un cadre unificateur englobant la plupart des schémas de commande existant pour les robots manipulateurs holonomes. Il fut ensuite étendu au cas des robots mobiles non-holonomes [Pissard-Gibollet 93]. Il repose sur le fait qu’une tâche robotique peut généralement s’exprimer en termes de la régulation à zéro d’une fonction de sortie particulière, la fonction de tâche, notée e(q, t), où t est le temps pris en tant que paramètre indépendant. La dépendance explicite au temps de la fonction de tâche permet de prendre en compte des variations du motif visuel de référence, dues par exemple à un déplacement de la cible. Or, dans le cadre de nos travaux, nous ne considérons qu’une cible fixe dans un environnement statique. La fonction de tâche ne dépendra donc que de q seul. En outre, les fonctions de tâches peuvent s’exprimer en termes de données sensorielles, proprioceptives ou extéroceptives. En particulier, dans le cadre de notre étude, elles seront définies à partir des indices visuels s fournis par la caméra. De ce fait, le robot est contrôlé par une loi de commande exprimée non plus dans l’espace d’état, mais dans l’espace de sortie, plus précisément l’espace capteur, d’où le terme de commande référencée capteur. Cependant il est nécessaire pour la mise en œuvre de ce formalisme, d’assurer un lien régulier entre l’espace d’état et l’espace capteur, afin que le problème soit bien posé.

Considérons e(q, t) une fonction de tâche décrivant l’application robotique à réaliser. On considère que la tâche est correctement effectuée sur l’intervalle [0, T] si:

\[
e(q, t) = 0 \quad \forall t \in [0, T]
\]

(II.14)

Le bon conditionnement de la commande requiert l’unicité de la trajectoire, noté q_r(t), solution de l’équation (II.14). Or, la fonction de tâche étant non-linéaire en q, la résolution de (II.14) peut conduire à la définition de plusieurs trajectoires distinctes. Pour éviter cela, une condition initiale q_0 est introduite, telle que q_r(0) = q_0 vérifie e(q_0, 0) = 0. Notons qu’une trajectoire vérifiant cette propriété est dite trajectoire idéale [Samson et al. 91]. Cependant, il se peut qu’aucune solution n’existe, exceptée la condition initiale ou au contraire qu’il y en ait une infinité, on parle alors dans ce dernier cas de tâche redondante [Samson et al. 91, Cadenat 99]. Cela se produit lorsque la jacobienne \( \frac{\partial e}{\partial q}(q_r(t), t) \) n’est pas régulière au voisinage de la trajectoire solution q_r(t). Outre l’unicité et l’existence de la trajectoire, il est nécessaire d’une part que la régulation de la fonction de tâche e(q, t) à
zéro impose la convergence de la trajectoire du robot vers la trajectoire solution désirée, et d’autre part qu’une petite variation de $e$ n’induise pas une grande variation de la configuration $q$ du robot. La notion de $\rho$-admissibilité permet de répondre à l’ensemble de ces questions.

**$\rho$-admissibilité**

La $\rho$-admissibilité permet de prouver l’existence d’une application bijective de classe $C^1$ dont la réciproque est aussi de classe $C^1$, c’est-à-dire l’existence d’un difféomorphisme entre l’ensemble des couples $(q,t)$ et $(e,t)$. Ce difféomorphisme permet alors d’établir un lien régulier entre l’espace d’état et l’espace capteur dans lequel s’exprime la commande. La $\rho$-admissibilité est définie de la façon suivante [Samson et al. 91] :

**Définition 1** $e(q,t)$ est une fonction de tâche $\rho$-admissible sur l’ensemble $C_{\rho,T}$ des couples $(q,t)$ défini sur l’intervalle de temps $[0,T]$ si et seulement si la fonction $F(q,t) = (e(q,t),t)$ admet un difféomorphisme de classe $C^k$ de $C_{\rho,T}$ sur la boule fermée $B_{\rho}$ centrée en 0 et de rayon $\rho$.

Le lecteur intéressé pourra trouver une définition plus formelle dans [Samson et al. 91]. Sur cette base, [Samson et al. 91] propose le théorème suivant :

**Théorème 1** Soit $\{e(q,t) ; q_0\}$ une tâche admettant une trajectoire idéale $q_r(t)$ définie sur l’intervalle de temps $[0,T]$. Alors :
- si $e(q,t)$ est de classe $C^2$,
- si la jacobienne de la tâche $J_e = \frac{\partial e}{\partial q}$ est inversible au voisinage de la trajectoire $q_r(t)$ :
  \[ \det \left| \frac{\partial e}{\partial q}(q_r(t),t) \right| \neq 0, \]
- si l’horizon de temps reste fini : $T < \infty$, il existe un $\rho > 0$ tel que la tâche considérée soit $\rho$-admissible.

Ainsi, la condition essentielle permettant d’établir la $\rho$-admissibilité est de définir la fonction de tâche de telle sorte que sa jacobienne soit inversible au voisinage de la trajectoire solution. Il nous suffit donc de contraindre autant de degrés de liberté que d’actionneurs disponibles sur le robot. L’obtention de cette propriété se réduit alors à un problème de modélisation de la tâche à effectuer. Finalement, la notion de $\rho$-admissibilité permet de montrer qu’il y a équivalence entre la synthèse de la loi de commande dans l’espace d’état et la synthèse dans l’espace capteur.

**II.3.2 La commande référencée vision**

Notre but est de synthétiser une loi de commande pour la réalisation d’une tâche référencée vision de positionnement face à une cible. Pour ce faire, nous exploitons le formalisme des fonctions de tâches que nous venons brièvement de présenter. Rappelons que dans l’ensemble de nos travaux nous considérons une cible composée de quatre points. Le signal visuel $s$ est alors défini par un vecteur de dimension 8, comprenant les coordonnées $(x_j,y_j)$ [II.11, p.110]. Dans ce cadre particulier, la fonction de tâche que nous cherchons à réguler, notée $e_1(q(t))$, de dimension $(8 \times 1)$, s’exprime comme suit :

\[ e_1(q(t)) = s(q(t)) - s^* \]

\footnote{Les fonctions de classe $C^k$ sont des fonctions continues différentiables, ainsi que leurs dérivées jusqu’à l’ordre $k$.}
où $s^*$ sont les indices visuels de référence. La jacobienne associée à la tâche référencée vision précédente est définie alors par :

$$J_{e_1} = \frac{\partial e_1}{\partial q} = L_{\text{robot}} \times J_{\text{robot}} \quad (\text{II.15})$$

où l’on retrouve la matrice d’interaction $L_{\text{robot}}$ (II.12, p.16) associée à $s$, et la matrice jacobienne du robot $J_{\text{robot}}$ (II.8, p.14). Cependant, la jacobienne $J_{e_1}$ étant de dimension $(8 \times 3)$, et non inversible, la fonction de tâche $e_1(q(t))$ n’est pas $\rho$-admissible. Une solution consiste alors à projeter $e_1(q(t))$ dans $\mathbb{R}^3$ en introduisant une matrice de combinaison $C$ [Chaumette 90]. La fonction de tâche obtenue, notée $e_{\text{pos}}$, s’écrit de la manière suivante :

$$e_{\text{pos}}(q(t)) = C e_1(q(t)) = C (s(q(t)) - s^*) \quad (\text{II.16})$$

La satisfaction de la condition de régularité de la jacobienne associée à la fonction de tâche $e_{\text{pos}}$, nécessaire pour la $\rho$-admissibilité, s’obtient en choisissant la matrice de combinaison $C$ de plein rang 3 et telle que la matrice $(C L_{\text{robot}})$ soit aussi de plein rang 3. Un choix judicieux consiste à fixer $C$ égale à la pseudo-inverse à gauche de la matrice d’interaction $L_{\text{robot}}$. Cependant, ce choix implique le calcul de la matrice $C$ à chaque pas de la boucle de commande, et donc la détermination en-ligne de la profondeur $z_j$ de la cible. De plus $C$ devient fonction de $q$, ce qui complexifie grandement la synthèse de la commande. Ainsi, tout comme dans [Chaumette 90], nous choisissons de fixer la matrice de combinaison égale à la pseudo-inverse de la matrice d’interaction à l’équilibre $L^*_\text{pl}$ (i.e. lorsque $s = s^*$). $C$ s’exprime alors comme suit :

$$C = L^*_\text{robot} = (L^*_\text{robot} L^*_\text{robot})^{-1} L^*_\text{robot} \quad \text{où} \quad L^*_\text{robot} = L_{\text{robot}}|_{s=s^*} \quad (\text{II.17})$$

Toutefois, le fait d’introduire la matrice de combinaison $C$ peut perturber la convergence des indices visuels $s$ vers leur consigne $s^*$. En effet, la régulation de $e_{\text{pos}}$ n’assure que la convergence de $Cs$ vers $Cs^*$, il n’y a donc pas équivalence entre la régulation de $e_{\text{pos}}$ et celle de $(s - s^*)$. Néanmoins, la matrice de combinaison permet de pouvoir prendre en compte un nombre d’informations capteur supérieur au nombre de degrés de liberté de la caméra contraints par la tâche référencée vision. La propriété de $\rho$-admissibilité ne dépend plus alors que de la structure mécanique du robot.

Afin d’assurer la convergence de $e$ vers 0 nous imposons à la fonction de tâche de décroître de manière exponentielle vers zéro [Chaumette 90]. Le comportement désiré en boucle fermée est alors décrit de manière générale par la relation suivante :

$$\dot{e} = J_T \times \dot{q} = -\lambda e \quad (\text{II.18})$$

où $\lambda > 0$ est un gain scalaire définissant la vitesse de décroissance exponentielle de la fonction de tâche, et $J_T$ désigne la matrice jacobienne de la tâche. Le correcteur permettant de satisfaire la dynamique (II.18) est alors donné par :

$$\dot{q} = -\lambda J^{-1}_{\text{pos}} e_{\text{pos}}$$

où $J_{\text{pos}}$ la matrice jacobienne de la fonction de tâche $e_{\text{pos}}$ qui est définie à partir des équations (II.15) et (II.16) comme suit :

$$J_{\text{pos}} = \frac{\partial e_{\text{pos}}}{\partial q} = CL_{\text{robot}} J_{\text{robot}} \quad (\text{II.19})$$
En remplaçant $C$ par $L^*_{\text{robot}}$ \[\text{(II.17)}\] l’équation \[\text{(II.19)}\] ci-dessus se simplifie de la façon suivante :

$$\hat{J}_{\text{pos}} = L^*_{\text{robot}} L_{\text{robot}} J_{\text{robot}} = (L^*_{\text{robot}} T L^*_{\text{robot}})^{-1} L^*_{\text{robot}} T L^*_{\text{robot}} J_{\text{robot}} = J_{\text{robot}}$$

Notons que cette matrice est carrée inversible, en rappelant que le déterminant de $J_{\text{robot}}$ est non nul \[\text{(II.8, p.14)}\].

Le correcteur à implémenter s’écrit donc sous la forme suivante :

$$\dot{q} = -\lambda J^{-1}_{\text{robot}} e_{\text{pos}}$$

soit finalement :

$$\dot{q} = -\lambda J^{-1}_{\text{robot}} C (s - s^*) \quad \text{(II.20)}$$

Dans ce chapitre, nous avons décrit une tâche référencée capteur à travers la synthèse d’une commande référencée vision. Finalement, nous avons établi les notions sur lesquelles sont basés nos travaux, en rappelant les principaux fondements de la commande référencée vision d’un robot mobile non-holonome muni d’une caméra. Nous montrerons dans le chapitre \[\text{III}\] suivant comment les techniques de commandes référencée vision que nous venons de présenter peuvent être étendues pour développer des lois de commande référencée multicapteurs, permettant à un robot mobile de naviguer dans un environnement contraint.
Chapitre III
Commande référencée multi-capteurs

La commande référencée capteur présentée dans le chapitre précédent, ou plus précisé-
ment la commande référencée vision, permet au robot de se positionner face à une cible,
grâce à l’utilisation des indices visuels dans la boucle de commande. Cependant, à elle seule,
elle ne permet pas de décrire des tâches robotiques plus élaborées comme la navigation en
environnement contraint. Cet asservissement ne se réalise donc sans encombre que si le robot
se déplace dans un environnement libre. Or, afin de se rapprocher des situations réelles, il
parait nécessaire de s’attacher à réaliser des tâches de navigation en environnement encom-
bré. Dans un premier temps, nous présenterons dans ce chapitre une méthode d’évitement
d’obstacle basée sur la définition d’un champ de potentiel. Puis nous montrerons comment
réaliser conjointement l’évitement et la tâche référencée vision en définissant une loi de com-
mande référencée multi-capteurs. Nous ne considérerons dans le cadre de notre étude que
des obstacles statiques et d’une hauteur inférieure à la caméra, afin d’éviter les problèmes
d’occultation.

III.1 Évitement d’obstacles par champ de potentiel

Dans le domaine de l’évitement d’obstacle, la recherche se divise traditionnellement en
deux approches : l’approche globale et l’approche locale. Dans les méthodes globales, la
tâche robotique est définie hors-ligne par un planificateur. La configuration des obstacles
est alors supposée connue et disponible pour le planificateur de trajectoires qui produit alors
un chemin reliant le point initial au point final en évitant les collisions avec les obstacles
connus. Les méthodes locales, que nous considérerons dans notre étude, utilisent en temps
réel les informations sensorielles qui donnent une information sur l’environnement proche du
robot. Sur la base de ces informations, l’objectif est alors de synthétiser une commande qui
guide le robot vers son but tout en tenant compte de la présence des obstacles. Les capteurs de perception d'environnement couramment employés sont les lasers, capteurs infrarouge ou à ultrasons qui permettent de caractériser localement l'obstacle le plus proche du robot, c'est-à-dire en fait le risque de collision.

La méthode que nous présentons ici s’appuie sur le formalisme des potentiels. Nous présentons dans un premier temps le formalisme des potentiels classiques proposé par [Khatib 86], puis décrivons le potentiel-rotation développé par [Khatib 96], avant d’introduire l’approche originale des potentiels-rotatifs proposée par [Cadenat et al. 99]. Enfin, nous détaillons la synthèse de la loi de commande garantissant l’évitement d’obstacle.

### III.1.1 Méthode des potentiels

L’idée du formalisme des potentiels est de créer un champ de potentiel artificiel, qui va guider les déplacements du robot vers l’objectif. Classiquement, le but à atteindre joue le rôle d’un potentiel attractif, a contrario les obstacles sont modélisés par des potentiels répulsifs. Le robot se comporte alors comme une particule chargée qui glisse le long du gradient de potentiels résultant de la somme des différents potentiels en cherchant à atteindre le minimum (e.g. le but). Cette approche semble bien adaptée à la problématique du contrôle du mouvement des robots en milieu encombré, dans la mesure où la force dérivant du potentiel permet de calculer une direction de déplacement et une accélération proportionnelle à son intensité.

Considérons un obstacle et $X$ un vecteur décrivant la configuration du robot. La fonction de potentiel artificiel $U(X)$ appliquée au robot s’écrit naturellement comme suit :

$$U(X) = U_g(X) + U_{\text{Obst}}(X)$$

où

- $U_g(X)$ définit le potentiel attractif généré par le but, qui peut s’exprimer de la manière suivante :

$$U_g(X) = \frac{1}{2} k_g (X - X_g)^T (X - X_g)$$

où $k_g > 0$ est un gain scalaire, et $X_g$ définit la configuration du but. $U_g(X)$ est alors une fonction positive dont la dérivée au premier ordre est continue, et dont le minimum est unique et de valeur nulle à $X = X_g$.

- $U_{\text{obst}}(X)$ caractérise le potentiel répulsif produit par l’obstacle. Il doit être positif, continu, dérivable et sa valeur doit tendre vers l’infini sur la surface de l’obstacle. De plus, afin d’éviter le déplacement du minimum de la fonction de potentiel attractif, nous définissons une zone de sécurité $\xi_0$ située à une distance $d_0$ autour de l’obstacle, qui délimite la région d’influence de l’obstacle. Nous supposons que le but est défini à l’extérieur de cette région afin que le problème soit bien posé. La fonction de potentiel $U_{\text{Obst}}(X)$ sera donc définie sur deux intervalles de distance en assurant toujours sa continuité de deuxième ordre, et une expression de ce potentiel est la suivante :

$$U_{\text{obst}}(X) = \begin{cases} 
\frac{1}{2} k_{\text{obst}} \left( \frac{1}{d_\text{O}} - \frac{1}{d_0} \right)^2 & \text{si } d_\text{O} < d_0 \\
0 & \text{sinon}
\end{cases}$$

où $d_\text{O}$ définit la plus courte distance entre le robot et l’obstacle.

\(^1\)Toutes les distances de cet ouvrage sont des distances euclidiennes.
Dans l’approche des potentiels classiques [Khatib 80, Khatib 86], le potentiel généré dépend de la distance à l’obstacle qui varie de façon continue; cette force produit donc une trajectoire continue. Cependant, il est important de noter que, dans cette représentation, la fonction de potentiel possède un gain constant quelle que soit la situation de l’obstacle par rapport au robot. En conséquence, le robot subit des forces produites par un obstacle même si sa trajectoire n’entraîne pas de risque de collision immédiate. Se basant sur la méthode des potentiels classiques, [Khatib 96] propose une extension de cette approche dans laquelle la fonction de potentiel dépend de la position et de l’orientation du robot par rapport à l’obstacle, appelé potentiel rotation. L’idée est de produire deux fonctions de potentiels autour de l’obstacle, l’une assurant la non-collision, et l’autre corrigeant les mouvements du robot autour de l’obstacle en agissant à une distance plus élevée.

En considérant la figure III.1 nous définissons
– $O$ le point de l’obstacle le plus proche du robot,
– $T_{obst}$ un vecteur unitaire tangent à l’obstacle en $O$ ;
– $\alpha$ l’angle entre la direction $X_M$ du robot et la tangente $T_{obst}$ de l’obstacle.

En imposant $\alpha \in [\frac{-\pi}{2}, \frac{\pi}{2}]$, la tangente $T_{obst}$ est alors unique et caractérise le sens de contournement de l’obstacle. Le paramètre $\alpha$ permet ainsi de caractériser la force produite par l’obstacle en fonction de l’orientation du robot. Nous définissons alors les deux potentiels répulsifs suivants :

$$
 p_1(X) = \frac{1}{2} k_1 \left( \frac{1}{d_O} - \frac{1}{d_0} \right)^2 
$$

$$
 p_2(X) = \frac{1}{2} k_2 \alpha^2 (d_O - d_0)^2
$$

avec $k_2 \gg k_1 > 0$ deux gains scalaires permettant de définir l’importance des différents potentiels répulsifs. Nous proposons maintenant la fonction de potentiel répulsif suivante :

$$
 U_{obst}^\alpha(X) = \begin{cases} 
 p_1(X) + p_2(X) & \text{si } d_O < d_0 \\
 0 & \text{sinon}
\end{cases}
$$

Ainsi, la fonction $p_1(X)$ prédomine près de la surface de l’obstacle et tend vers l’infini pour assurer la non-collision. Au contraire la fonction de potentiel $p_2(X)$ permet de pondérer
$U_{\text{Obst}}^\alpha$ en fonction de l’angle $\alpha$ et d’éviter des mouvements de contournement importants lorsque cela n’est pas nécessaire. De ce fait, lorsque le robot pénètre dans la zone délimitée par $\xi_0$ parallèlement à l’obstacle, l’angle $\alpha$ est petit et le mouvement du robot n’est que très peu modifié. Inversement, lorsque le robot se dirige droit vers l’obstacle, l’intensité de la force s’accroît et le mouvement de contournement est amplifié. La figure III.2 présente brièvement les différentes trajectoires obtenues selon les différentes approches.

Les approches présentées restent néanmoins fortement sensibles aux minima locaux, appelés *puits de potentiels*, qui se produisent lorsque les forces attractive et répulsive s’annulent. Le robot se trouve alors immobilisé dans une configuration dont il ne peut pas sortir. De petits mouvements aléatoires doivent alors être mis en œuvre pour permettre au robot de quitter le puits de potentiel et terminer la tâche désirée.

Remarque 2 Notons que nous nous intéressons ici qu’au cas d’obstacle unique. Dans le cas où plusieurs obstacles sont considérés la force répulsive appliquée au robot est donnée par la somme de toutes les forces générées par chacun des obstacles [Khatib 96]. Cependant, dans la mesure où nous avons défini une enveloppe $\xi_0$ qui délimite la zone d’influence d’un obstacle, en posant que la distance entre deux obstacles doit être supérieure à $2d_0$ nous pouvons considérer à tout instant uniquement l’obstacle le plus proche.

III.1.2 Méthode des potentiels rotatifs

Les limitations des approches décrites précédemment naissent de la présence de deux forces antagonistes pendant la phase d’évitement. Afin d’y remédier, nous proposons d’utiliser la méthode des potentiels rotatifs qui repose sur la séparation des effets attractif et répulsif des différentes forces [Cadenat et al. 99]. Elle s’inspire des méthodes dites de champ de vortex [De Luca & Oriolo 94]. L’idée est de générer un champ de potentiel autour de l’obstacle qui impose au robot de suivre une enveloppe de sécurité. Or, dans la méthode des potentiels classiques la force répulsive repousse le robot jusqu’à l’enveloppe de sécurité $\xi_0$ qui délimite le domaine de répulsion. La force correspondante tend alors vers zéro, et le robot s’immobilise en l’absence de toute force attractive. C’est pourquoi, nous nous proposons de définir un potentiel répulsif capable de maintenir la commande du robot au voisinage de $\xi_0$. Pour cela, nous définissons trois enveloppes autour de l’obstacle (cf. figure III.3) :

1. la première, notée $\xi_-$ située à une distance $d_-$, définit la région où l’obstacle est considéré comme réellement dangereux pour le robot ;
2. la seconde, notée $\xi_0$, se trouve à une distance $d_0 > d_-$ et représente l’enveloppe de sécurité que le robot doit emprunter pour contourner l’obstacle ;

En fait, d’un point de vue expérimental, la force générée oscille en intensité et orientation autour de 0, il s’avère donc difficile de détecter les puits de potentiels.
III.1. Évitement d’obstacles par champ de potentiel

3. la troisième, notée $\xi_+$, se trouve à une distance $d_+ > d_0$ et caractérise la zone à partir de laquelle l’obstacle est pris en considération, c’est-à-dire la limite de la zone d’influence de l’obstacle. D’autre part, pour que le problème soit bien posé, nous supposons que la distance entre deux obstacles reste supérieure à $2d_+$. 

![Diagramme III.3 – Évitement d’obstacle avec potentiel rotatif.](image)

Notre objectif est de déterminer un potentiel répulsif permettant au robot mobile de contourner l’obstacle sans l’aide de potentiel attractif. Nous définissons l’orientation de la force répulsive de telle sorte qu’elle soit :
- orthogonale à la surface de l’obstacle lorsque le robot en est très proche ;
- tangente à l’obstacle lorsque le robot se trouve sur l’enveloppe $\xi_0$ ;
- légèrement orientée vers l’obstacle, lorsque le robot se trouve entre les enveloppes $\xi_0$ et $\xi_+$ de manière à d’imposer au robot de suivre l’enveloppe $\xi_0$.

Afin de déterminer une telle force, nous considérons la fonction de potentiel suivante :

$$
U_{obst}(d_O) = \begin{cases} 
\frac{1}{2}k_1 \left( \frac{1}{d_O} - \frac{1}{d_+} \right)^2 + \frac{1}{2}k_2 (d_O - d_+)^2 & \text{si } d_O < d_+ \\
0 & \text{sinon}
\end{cases}
$$  \hspace{1cm} (III.2)

où $k_1$ et $k_2$ sont deux gains scalaires positifs. Comme dans le cas du formalisme du potentiel rotation, nous retrouvons un premier terme prépondérant à proximité de l’obstacle qui tend vers l’infini pour garantir la non-collision. Le second terme quant à lui, dominant au voisinage de $\xi_+$, permet au potentiel de conserver une valeur suffisamment importante pour pouvoir commander le robot sur l’enveloppe $\xi_0$. Sur la base de la fonction de potentiel (III.2) nous définissons séparément le module et l’orientation de la force répulsive, notés respectivement $F$ et $\beta$ :

- Le module se déduit par la dérivée de la fonction de potentiel $U_{obst}$ par rapport à $d_O$, soit :

$$
F = -\frac{\partial U_{obst}}{\partial d_O}(d_O) = k_1 \frac{1}{d_O^2} \left( \frac{1}{d_O} - \frac{1}{d_+} \right) - k_2 (d_O - d_+) 
$$  \hspace{1cm} (III.3)
L’orientation $\beta$, exprimée dans le repère lié au robot, se définie en fonction de la distance à l’obstacle $d_0$ et de l’orientation relative du robot par rapport à l’obstacle, $\alpha$, de la manière suivante :

$$\beta = \alpha - \frac{\pi}{2} \frac{d_0}{d_0} + \frac{\pi}{2} \quad \text{(III.4)}$$

Ainsi, à proximité de l’obstacle l’angle $\beta$ tend vers $\alpha + \frac{\pi}{2}$ produisant une force répulsive normale à l’obstacle. De même, sur l’enveloppe $\xi_0$, $d_0 = d_O$ et $\beta = \alpha$, le robot subit une force tangentielle non nulle qui permet au robot suivre l’enveloppe de sécurité $\xi_0$. Si le robot s’écarte de cette enveloppe et pénètre dans le couloir délimité par $\xi_0$ et $\xi_+$ l’angle $\beta$ diminue et la force est alors légèrement dirigée vers l’obstacle, ramenant le véhicule vers $\xi_0$.

À présent nous sommes en mesure de déterminer, sur la base des équations (III.3) et (III.4), la commande de la base mobile, c’est-à-dire les vitesses linéaire et angulaire assurant l’évitement d’obstacle. En effet, les lois de la mécanique classique permettent de lier les forces appliquées à un système à sa dynamique, et donc d’établir la commande cinématique de ce système. Pour ce faire, il est nécessaire de définir un point ($P_F$) d’application de cette force. Le choix le plus simple pour un système holonome consiste à choisir son centre d’inertie. En revanche, dans le cas d’un système non-holonome, c’est-à-dire ayant des contraintes cinématiques non intégrables, le problème est plus complexe. En particulier, [Khatib 96] montre qu’un tel système ne peut être commandé en plaçant le $d$’application $P_F$ sur l’axe des roues ($Y_M$). Un choix judicieux consiste à prendre $P_F$ sur l’axe $X_M$ du robot, et plus précisément de le faire coïncider avec le centre $P$ de la platine. La commande d’évitement s’exprime alors comme suit :

$$v_{evit} = k_{evit}^w F \cos \beta \quad \text{(III.5)}$$

$$\omega_{evit} = k_{evit}^w \frac{F}{D_F} \sin \beta \quad \text{(III.6)}$$

où $k_{evit}^w$ et $k_{evit}^w$ sont deux gains scalaires positifs qui permettent de fixer la dynamique de la commande d’évitement. $D_F$ quant à lui permet de prendre en compte le bras de levier défini par le centre $M$ du robot et le point d’application $P_F$ de la force, c’est-à-dire $P$ le centre de la platine, en d’autres termes $D_F = D_x$ (cf figure II.1, p. II).

**Remarque 3** Selon le **principe fondamental de la dynamique** nous savons qu’une force est proportionnelle à une accélération. Cependant, quand un robot peut être commandé en vitesse et que seul son modèle cinématique est pris en compte, la force virtuelle produite peut être considérée comme directement proportionnelle à la vitesse.

L’approche des potentiels rotatifs permet de synthétiser une loi de commande contraignant le robot à se déplacer le long d’une enveloppe de sécurité $\xi_0$. En ce sens, elle s’apparente aux méthodes de suivi de chemin. Ainsi, en définissant un chemin de contournement autour de chaque obstacle les problèmes de puits de potentiels inhérents aux méthodes classiques sont limités. Il est alors possible de considérer des obstacles concaves, si néanmoins leur concavité n’est pas trop importante, c’est-à-dire si l’enveloppe de sécurité $\xi_0$ n’est pas suffisante pour combler la concavité. En effet, dans le dernier cas, le robot ne peut pas définir de manière unique le point $O$ le plus proche de l’obstacle (cf. figure III.4).
III.2 Commande référencée multi-capteurs en environnement encombré

Notre objectif est de synthétiser une loi de commande permettant à un robot mobile non-holonome de naviguer dans un environnement contraint. Nous considérons deux tâches de navigation spécifiques :

1. La première, conçue pour valider la méthode d’évitement, consiste à positionner le robot en une position désirée, et s’appuie sur le formalisme des potentiels décrits précédemment.

2. La seconde consiste à positionner la caméra embarquée face à une cible fixe et repose sur le formalisme des fonctions de tâches introduit au chapitre II.

Afin de pouvoir réaliser ces deux tâches dans un environnement encombré, il est nécessaire d’intégrer les fonctionnalités d’évitement d’obstacles. La stratégie envisagée ici consiste à synthétiser séparément la commande d’évitement et celle guidant le robot vers son objectif, puis à fusionner les deux correcteurs ainsi obtenus par combinaison convexe. Il est ainsi possible de définir une loi de commande globale permettant de réaliser la tâche de navigation désirée. Celle-ci s’écrit [Cadenat 99] :

\[
\dot{q} = (1 - \mu)\dot{q}_g + \mu\dot{q}_{\text{evit}} \tag{III.7}
\]

où

- \( \dot{q} = (v \ \omega \ \omega_{pl})^T \) définit la commande en vitesse globale appliquée au robot. Nous rappelons que \( v \) et \( \omega \) représentent les vitesses linéaires et angulaires de la base mobile, et \( \omega_{pl} \) la vitesse angulaire de la platine ;

- \( \dot{q}_g = (v_g \ \omega_g \ \omega_{pl_g})^T \)

représente le correcteur qui permet de réaliser la tâche robotique ;

- \( \dot{q}_{\text{evit}} = (v_{\text{evit}} \ \omega_{\text{evit}} \ \omega_{pl_{\text{evit}}})^T \) définit le correcteur assurant l’évitement d’obstacle ;

- \( \mu \) est une fonction de la distance à l’obstacle \( (d_O) \) qui permet de lisser le basculement d’un correcteur à l’autre.

En effet, cette fonction \( \mu \) permet d’activer l’un ou l’autre correcteur en fonction de l’environnement. Ainsi lorsqu’il n’y a pas de risque de collision (\( e.g. \) le robot est hors de la zone d’influence \( \xi_0 \) de l’obstacle) \( \mu \) est fixé à 0 et seul \( \dot{q}_g \) commande le robot. Inversement si le robot est à proximité d’un obstacle, \( \mu \) passe à 1 de manière à ce que le robot assure l’évitement par le correcteur \( \dot{q}_{\text{evit}} \). Une évolution de \( \mu \) purement binaire n’est pas envisageable,\(^3\) Les indices \( g \) et \( \text{evit} \) caractérisent simplement le type de correcteur considéré.

\(^3\) Les indices \( g \) et \( \text{evit} \) caractérisent simplement le type de correcteur considéré.
car cela produirait une discontinuité de la loi de commande (III.7). \( \mu \) est donc défini comme une fonction continue de la distance, et elle dépend de la stratégie de commande souhaitée. Nous le définissons selon l’algorithme de commande suivant :

\[
\begin{align*}
\mu &= 0 \quad \text{si} \quad d_O \geq d_0 \quad \text{ET} \quad AVOID = 0 \\
\mu &= \frac{d_O - d_0}{d_0 - d_0} \quad \text{si} \quad d_O \in [d_-, d_0] \quad \text{ET} \quad AVOID = 0 \\
\mu &= \frac{d_O - d_0}{d_S - d_+} \quad \text{si} \quad d_O \in [d_S, d_] \quad \text{ET} \quad LEAVE = 1 \\
\mu &= 1 \quad \text{sinon}
\end{align*}
\]

où

- \( AVOID \) est un indicateur positionné à 1 lorsque le robot franchit l’enveloppe de sécurité \( \xi_\). Il permet de maintenir \( \mu \) à 1 tant qu’il y a un risque de collision.
- \( LEAVE \) est un indicateur de sortie qui passe à 1 lorsque la condition de sortie est satisfaite. Cet événement se produit lorsque la base mobile se trouve parallèle à l’obstacle, c’est-à-dire lorsque l’angle \( \alpha \) devient égal à zéro. Il est ainsi possible d’imposer au robot de suivre l’enveloppe de contournement \( \xi_0 \) tant que cela s’avère nécessaire, \( \text{i.e.} \) tant que l’obstacle présente un danger.
- \( d_S \) est la distance du robot à l’obstacle relevé par le capteur quand la condition de sortie intervient (\( \text{i.e.} \) lorsque \( LEAVE \) passe de 0 à 1). Elle permet de garantir la continuité de l’évolution de \( \mu \) au moment où le robot est autorisé à quitter l’enveloppe de sécurité autour de l’obstacle.

Nous décrivons à présent la mise en œuvre des différents correcteurs proposés.

### III.2.1 Tâche de positionnement dans l’espace cartésien

Notre objectif est ici d’asservir le robot en position. Pour cela, nous disposons d’odomètres qui fournissent la position du robot relativement au but assigné \( \text{[..]} \). Posons : \((x, y)\) la position du robot, et \((x_g, y_g)\) la position de l’objectif, dans un repère global \( R_0 \).

La méthode que nous présentons ici se base sur le formalisme des potentiels présenté précédemment puisque la définition d’un but peut se modéliser par un potentiel attractif. De ce fait, nous proposons la fonction de potentiel suivante, inspirée de l’équation (III.1) :

\[
U_g(d_g) = k_g d_g
\]

où

- \( k_g \) un gain scalaire positif,
- \( d_g \) est la distance euclidienne au but, définie par :

\[
d_g = \sqrt{(x - x_g)^2 + (y - y_g)^2}
\]

La force, \( F_g = (F_X \ F_Y)^T \), dérivée de ce potentiel, se déduit de la manière suivante :

\[
F_g = -\frac{\partial U_g}{\partial d_g}(d_g) = -k_g \begin{pmatrix}
x - x_g \\
y - y_g \\
\frac{d_g}{d_g}
\end{pmatrix}
\]

\( \text{[..]} \) La position du robot étant mesurée par odométrie, cette tâche peut également être vue comme une tâche référencée capteur.
Remarque 4 Dans la mesure où \( x - x_g \) et \( d_g \) converge de même manière identique vers 0, il n'y a donc pas de problème de singularité dans la détermination de la force \( F_g \). De plus, la norme de cette force attractive est égale à 1. En effet, nous avons :

\[
||F_g|| = \sqrt{F_x^2 + F_y^2} = \sqrt{\left(\frac{x - x_g}{d_g}\right)^2 + \left(\frac{y - y_g}{d_g}\right)^2} = \sqrt{\frac{(x - x_g)^2 + (y - y_g)^2}{d_g^2}} = 1
\]

Ainsi, lorsque le robot converge vers l'objectif assigné, la force produite est suffisante pour amener le robot à la position souhaitée.

Remarque 5 Il est difficile de faire parvenir le robot mobile exactement à la position souhaitée. En effet, le robot oscille autour de la position but en fonction de sa dynamique. De ce fait, nous définissons une distance minimale, \( d_\varepsilon \), autour du point objectif à partir de laquelle le robot peut s'arrêter.

Pour définir la loi de commande \( \dot{q}_g \) associée, nous considérons toujours \( P \) le centre de la platine comme point d’application de la force \( F_g \). Par ailleurs, pour tenir compte de la contrainte de non-holonomie, nous choisissons ici de projeter la force en fonction de l’orientation du robot, afin de définir une commande dans le sens de mouvement du robot. En considérant la force \( F_g = (F_x \ F_y)^T \), nous définissons les vitesses linéaire et angulaire comme suit :

\[
v_g = k_v^g (F_x \cos \theta + F_y \sin \theta) \tag{III.8}
\]
\[
\omega_g = \frac{k_\omega^g}{D_x} (-F_x \sin \theta + F_y \cos \theta) \tag{III.9}
\]

où \( k_v^g \) et \( k_\omega^g \) sont deux gains scalaires positifs qui permettent de fixer la dynamique de la commande. Finalement, nous déduisons des équations (III.8) et (III.9) la commande suivante :

\[
\dot{q}_g = (v_g \quad \omega_g \quad 0)^T \tag{III.10}
\]

Nous avons donc défini un correcteur permettant au robot d’atteindre la position désirée en environnement libre. Il est maintenant possible de réaliser cette même tâche en environnement encombré en fusionnant cette commande avec la loi d’évitement définie par les expressions (III.5) et (III.6), à l’aide de la relation (III.7).

Remarque 6 Dans la mesure où le robot ici n’est pas guidé par la vision, la platine n’est pas commandée et \( \omega_{pl} \) est donc fixé à zéro.
III.2.2 Tâche référencée vision

Nous nous intéressons à présent à l’exécution d’une tâche référencée vision de positionnement. Cette tâche est définie, en environnement libre, par la fonction de tâche $e_{pos}(q(t)) = C(s(q(t)) - s^*)$ donnée par la relation (II.16). Nous rappelons aussi que le correcteur permettant de réguler $e_{pos}$ à zéro est défini par l’équation (II.20), comme suit :

$$\dot{q} = -\lambda J^{-1}_{robot} C (s - s^*)$$

Ce correcteur correspond donc à notre commande $\dot{q}_g$ de la stratégie envisagée.

Intéressons-nous maintenant au cas de l’évitement d’obstacle. La stratégie de commande envisagée jusqu’ici consiste à suivre un chemin de référence défini par une enveloppe de sécurité autour de l’obstacle. Dans ce cas, nous avons montré que la commande de la base mobile était donnée par la vitesse linéaire, $v_{evit}$, et angulaire, $\omega_{evit}$, définies respectivement par les expressions (III.5) et (III.6). Ainsi pendant la phase d’évitement, la base mobile sera commandée par le vecteur :

$$\vec{q}^{base}_{evit} = (v_{evit} \ \omega_{evit})^T \quad (III.11)$$

Il reste alors à commander la platine de manière à maintenir la cible dans le champ de vision de la caméra.

Commande de la platine

Nous nous appuyons ici sur les travaux de [Tsakiris et al. 96] et [Cadenat 99]. La méthode employée repose sur la désolidarisation de la base mobile et de la platine. Ainsi, pendant la phase de transition où le robot cherche à atteindre sa configuration finale, la platine est commandée indépendamment de la base de telle sorte que les indices visuels ne soient jamais perdus.

Nous rappelons que la cible considérée est constituée de quatre points, le vecteur des informations visuelles $s$ comprend les coordonnées $(x_j, y_j)$ de ces points projetés sur le plan image (cf. l’équation (II.11)). Pendant la phase d’évitement, si le mouvement de la base n’est pas compensé, la valeur de la fonction de tâche $e_{pos}(q(t))$ augmente et les indices visuels peuvent être perdus. Dans la mesure où le mouvement de la caméra n’est commandé que dans le plan horizontal, cette erreur se traduit par un déplacement latéral des indices visuels dans l’image, et affectera donc les ordonnées $y_j$ des points projetés. Pour cette raison, nous considérons que la tâche référencée vision à réaliser en phase d’évitement est de réguler à zéro, une nouvelle fonction de tâche $e_Y$ dépendant des ordonnées $y_j$ et de leurs consignes $y^*_j$.

En outre, puisque nous n’utilisons qu’une seule commande pour déplacer la caméra indépendamment de la base, nous considérons uniquement une erreur scalaire afin de satisfaire la propriété de $\rho$-admissibilité. Pour cela, nous nous proposons de centrer la cible dans l’image et définissons la fonction de tâche comme suit :

$$e_Y = y_0$$

où $y_0$ représente l’ordonnée du centre de la cible dans le plan image. Nous imposons alors la dynamique suivante à la fonction de tâche :

$$\dot{e}_Y = L_0 T_{red} = -\lambda_{pl}e_Y \quad \lambda_{pl} > 0 \quad (III.12)$$
où $\lambda_{pl}$ est un gain positif scalaire, et $L_0$ désigne la matrice d’interaction réduite définie uniquement pour l’ordonnée $y_0$ soit :

$$
L_0 = \left( \begin{array}{ccc}
-\frac{1}{z_0} & \frac{y_0}{z_0} & 1 + y_0^2
\end{array} \right)
$$

Il reste désormais à déterminer la commande de la platine satis faisant cette dynamique. Dans ce but, nous partitionnons la jacobienne du robot, $J_{robot}$, de manière à séparer les termes relatifs à la base, $J_{base}$, de ceux de la platine, $J_{pl}$, soit en reprenant la définition du torseur cinématique réduit de la caméra (II.8) :

$$
T_{red} = (J_{base} J_{pl}) \times (\dot{q}_{base}^\text{evit} \dot{q}_{pl}^\text{evit})
$$  (III.13)

où
- $J_{base}$ désigne la matrice constituée des deux premières colonnes de la jacobienne $J_{robot}$ ;
- $J_{pl}$ est donnée par la troisième colonne de $J_{robot}$ ;
- $\dot{q}_{evit}$ représente les vitesses linéaire et angulaire en phase d’évitement, définie par la relation (III.11) ;
- $\dot{q}_{pl}^\text{evit}$ désigne la commande de la platine, c’est-à-dire la vitesse de rotation $\omega_{pl}$.

En considérant les équations (III.12) et (III.13) nous déduisons :

$$
L_0 J_{base} \dot{q}_{base}^\text{evit} + L_0 J_{pl} \dot{q}_{pl}^\text{evit} = -\lambda_{pl} e_Y
$$

D’où nous déduisons l’expression de la commande de la platine permettant de centrer la cible dans l’image en phase d’évitement :

$$
\dot{q}_{pl}^\text{evit} = -\frac{1}{L_0 J_{pl}} (\lambda_{pl} e_Y + L_0 J_{base} \dot{q}_{base}^\text{evit})
$$

**Remarque 7** Le terme $L_0 J_{pl}$ est un scalaire non nul du fait de la définition de la fonction de tâche $e_Y$. La commande $\dot{q}_{pl}^\text{evit}$ peut donc être déterminée de manière unique et exacte.

Ainsi, la commande de la platine intègre l’erreur de suivi de la cible, tout en compensant les déplacements de la base en phase d’évitement. De cette manière, la conservation de la cible dans l’image est garantie pendant la phase d’évitement. Nous disposons ainsi de la commande d’évitement suivante :

$$
\dot{q}_{evit} = \begin{pmatrix}
\dot{q}_{base}^\text{evit} \\
\dot{q}_{pl}^\text{evit}
\end{pmatrix} = \begin{pmatrix}
v_{evit} \\
\omega_{evit} \\
\omega_{pl}^\text{evit}
\end{pmatrix}
$$

Finalement, nous avons deux correcteurs, l’un permettant d’exécuter une tâche de positionnement référencée vision dans l’espace libre, l’autre permettant l’évitement d’obstacle tout en assurant le suivi de la cible. Nous pouvons alors fusionner ces deux correcteurs à l’aide de la relation (III.7).
À présent, nous allons présenter les moyens mis en œuvre pour la validation des différentes lois de commande développées dans le cadre de nos travaux. Nous les avons tout d’abord validés en simulation Matlab®, avant de les expérimenter sur le robot SuperScout II. Nous présentons dans un premier temps les modules, développés sous Gm²M, avant de décrire les résultats expérimentaux obtenues.

IV.1 Implémentation des modules

Comme nous l’avons préalablement mentionné, nos algorithmes sont encapsulés à l’intérieur de Gm²M que nous avons déjà présenté au chapitre I. Afin de faciliter l’accès aux différentes lois de commande décrites dans cet ouvrage, nous les avons regroupées au sein d’un module unique, nommé **VSRP**, pour *Visual Servoing and Rotational Potential*, qui doit pouvoir accéder aux services offerts par les trois modules suivants :

1. le module **scout** qui permet d’accéder aux commandes en vitesse de la base mobile, ainsi qu’à la lecture des données sensorielles embarquées, notamment les capteurs ultrasonique et odométrique.
2. le module **platine** qui permet de commander la platine.
3. le module **extract** qui permet de lire la position des indices visuels, en faisant appel aux services du module **camera** qui gère l’accès aux images.

Les différents accès sont représenté sur la figure [IV.1]. Les deux premiers modules ont été mis en œuvre au LAAS par Sara FLEURY. Nous avons développé le dernier en partenariat
Chapitre IV. Implémentation et aspect expérimentaux

Fig. IV.1 – Organisation des modules.

avec Stéphane MAS, stagiaire chargé de mettre en place des travaux pratiques de robotique, traitant des problèmes de planification et de localisation sur des amers par vision stéréoscopique.

Tout d’abord, nous présentons brièvement le module extract, avant de décrire le module VSRP. Enfin, nous terminons en décrivant les résultats expérimentaux obtenus pour les différentes lois de commande.

IV.1.1 Le module extract

Pour les besoins de l’asservissement visuel, il est nécessaire de déterminer la position des indices visuels dans l’image fournie par la caméra SONY. Celle-ci renvoie des images couleur, de 640 colonnes par 480 lignes, au format YUV compressé en 4 : 2 : 2. Ce format permet de restituer au final les composantes RGB et présente l’avantage de faciliter la compression des couleurs et de fiabiliser le transport du signal vidéo.

Il existe dans la littérature de nombreuses approches pour la détection et l’extraction d’informations visuelles dans une image. En ce qui nous concerne, les critères prépondérants sont les suivants :

- les performances temporelles : l’algorithme d’extraction doit pouvoir s’intégrer dans une boucle d’asservissement.
- la précision des points extraits : l’information visuelle obtenue doit être suffisamment précise pour limiter les incertitudes de modélisation.

Le module extract s’appuie sur un algorithme d’extraction de taches proposé par Frédéric LERASLE. Le formalisme associé à cet algorithme est décrit sur la figure [IV.2].

\[ Y \] représente la luminance, \( U \) et \( V \) les composantes Rouge et Bleu, appelées chrominances. La compression 4 : 2 : 2 signifie que le codage de la couleur d’un bloc est effectué avec quatre valeurs pour la luminance suivies de quatre valeurs pour les chrominances (deux pour \( U \) et deux pour \( V \)), soit finalement le codage...
IV.1. Implémentation des modules

Cependant, l’algorithme original fonctionne uniquement sur des images en niveaux de gris. Dans la mesure où les nouvelles caméras numériques installées sur le robot renvoient des images en couleur, il a été nécessaire d’adapter l’algorithme en conséquence. De plus, l’étiquetage des régions s’effectuant de manière récursive sur tous les pixels marqués de l’image, la fonction est appelée de manière très fréquente, générant un trop grand nombre d’appel à cette fonction et un débordement de la pile. De ce fait, il est nécessaire de limiter le nombre d’appels en utilisant un algorithme de fusion pour les régions adjacentes étiquetées différemment.

L’algorithme d’extraction est basé sur les cinq étapes suivante :

1. **Seuillage** : les images YUV issues de la caméra sont d’abord converties en composantes RGB. Puis un seuillage est effectué sur les trois plans R, G et B.


3. **Étiquetage** : toujours pour chacun des plans, chaque pixel marqué de l’image est parcouru et, en considérant ses 4-voisins, une étiquette lui est attribuée. L’étiquetage est alors propagé aux voisins marqués, puis les régions étiquetées connexes sont fusionnées.

4. **Extraction des attributs** : en se basant sur cette dernière image, les attributs (surface, centre de gravité...) des régions étiquetées sont alors extraits.

5. **Filtrage** : enfin, les données obtenues sont filtrées à partir des caractéristiques connues des taches à extraire.

Ainsi, après le filtrage, nous disposons de la position des indices visuels définies par les coordonnées des centres de gravité des taches.

**Remarque 8** Selon la focale utilisée, il peut s’avérer nécessaire d’effectuer une correction de la distorsion afin de limiter l’importance des déformations des bords de l’image.

Suivant : YYYYYUUVV.

Les images produites sont de 640 colonnes par 480 lignes, soit des images de 307200 pixels.
Dans un premier temps, nous avons développé et validé cet algorithme sous Calife⁴ avant de l’implanter au sein d’un module GœM, nommé extract. Le tableau IV.1 suivant présente les requêtes disponibles, ainsi que leurs données d’entrées/sorties :

<table>
<thead>
<tr>
<th>Requête</th>
<th>Description</th>
<th>Type</th>
<th>input/output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>Initialisation du module, et recherche des posters</td>
<td>init</td>
<td>imageRef, posterNames</td>
</tr>
<tr>
<td>Stop</td>
<td>Arrêt des activités du module</td>
<td>contrôle</td>
<td></td>
</tr>
<tr>
<td>GetExtractParams</td>
<td>Lecture des paramètres d’extraction</td>
<td>contrôle</td>
<td>extractParams</td>
</tr>
<tr>
<td>SetExtractParams</td>
<td>Écriture des paramètres d’extraction</td>
<td>contrôle</td>
<td>extractParams</td>
</tr>
<tr>
<td>ExtractTarget2D</td>
<td>Lance l’algorithme d’extraction sur l’image courante</td>
<td>exec</td>
<td>mode</td>
</tr>
</tbody>
</table>

Tab. IV.1 – Requêtes du module extract

La requête Init permet d’initialiser le module en spécifiant :
- l’image de référence pour une utilisation du module sur différentes sources d’images ;
- le nom des posters, qui est composé de :
  - images le poster où sont lues les images ;
  - extractParams le poster qui définit les paramètres d’extraction, ce qui permet de proposer une modification dynamique de ces paramètres par un autre module.

La requête ExtractTarget2D permet de lancer l’algorithme d’extraction sur une cible constituée de quatre tâches, et de placer la position des indices visuels dans un poster Target2D. Elle prend en paramètre le mode d’exécution de la requête, à savoir exécuter une seule extraction ou lancer l’activité d’extraction en boucles (la requête stop permettant alors de l’arrêter). Ainsi, l’opérateur/superviseur peut définir la séquence acquisition/extraction au sein d’une même boucle, ou bien dans deux boucles distinctes.

Remarque 9 Il est prévu une extension de ce module pour une utilisation avec des caméras installées sur un banc stéréoscopique. Il sera alors possible d’extraire des informations visuelles tridimensionnelles. Ce travail sera réalisé par Stéphane MAS.

IV.1.2 Le module VSRP

Le module VSRP, pour Visual Servoing and Rotational Potential, permet d’accéder aux différentes lois de commande présentées dans cet ouvrage. Il permet ainsi de réaliser une l’asservissement visuel seul (i.e. en environnement libre) ou bien combiner à la méthode des potentiels rotatifs pour l’évitement d’obstacle. Il est également possible d’effectuer une tâche de positionnement dans l’espace cartésien. Le tableau IV.1 détaille l’ensemble des requêtes disponibles.

Tout comme pour le module extract, la requête Init permet d’initialiser le module et de rechercher les posters définis par les requêtes associées. Cette requête prend en paramètre le mode d’exécution du module : l’asservissement seul ou en présence d’obstacle, et tâche de positionnement. Le choix du mode, permet d’offrir des services adaptés selon les besoins de l’opérateur/superviseur, et permet de filtrer en conséquence les éventuelles erreurs. Ensuite,

---

⁴Calife est un environnement de développement pour le traitement d’image développé au LAAS-CNRS.
IV.2. Résultats expérimentaux

Nous présentons à présent quelques-uns des nombreux résultats que nous avons obtenus pour nos différentes lois de commande.

IV.2.1 La tâche de positionnement dans l’espace cartésien

Dans le cadre de nos travaux nous avons validé expérimentalement la tâche de positionnement dans l’espace cartésien. Nous présentons ici principalement deux résultats obtenus, l’un en environnement libre, le second en présence d’un obstacle.

Tâche de positionnement en environnement libre

Nous avons relevé au cours de nos expérimentations les trajectoires ainsi que les vitesses linéaire et angulaire du robot au cours de sa tâche de positionnement en l’absence d’obs-
Chapitre IV. Implémentation et aspect expérimentaux

tacle. Les résultats sont donnés sur la figure IV.3. Nous avons également représenté sur la tracjetoire du robot la force attractive que celui-ci subit

\[ -1 \quad 0 \quad 1 \quad 2 \quad 3 \]
\[ -0.5 \quad 0 \quad 0.5 \]

Le but

Position du robot et de la force attractive

X  (m)
Y  (m)
0 200 400 600 ...

Vitesse lineaire −− global −−

\[ 0 \quad 200 \quad 400 \quad 600 \quad 800 \]
\[ -50 \quad 0 \quad 50 \]

Vitesse angulaire −− global −−

\[ 0 \quad 200 \quad 400 \quad 600 \quad 800 \]
\[ -50 \quad 0 \quad 50 \]

\[ \xi_0 \]
\[ d_0 = 50 \text{cm} \]
\[ \xi_- \quad \xi_+ \]

Fig. IV.3 – Tâche de positionnement dans l’espace cartésien seul.

Comme nous pouvons le constater la tâche de positionnement est parfaitement réalisée. Par ailleurs, il semble intéressant de remarquer que dans la mesure où la force attractive, projetée dans le sens de déplacement, fait “basculer” le robot avant d’atteindre l’objectif. Celle-ci génère alors un point de rebroussement sur la trajectoire et permet au robot de réaliser sa mission.

Tâche de positionnement en présence d’obstacle

Un obstacle circulaire est maintenant placé sur la trajectoire du robot. Nous avons fixé ici l’enveloppe de sécurité $\xi_0$ à une distance $d_0 = 50\text{cm}$ de l’obstacle, et les enveloppes $\xi_-$ et $\xi_+$ à 10cm de part et d’autre de $\xi_0$. Comme précédemment, nous avons relevé au cours de nos expérimentations les trajectoires, les vitesses linéaire et angulaire du robot au cours de sa tâche de positionnement, ainsi que la distance, $d_0$, et l’orientation, $\alpha$, entre le robot et l’obstacle. Les résultats sont regroupés sur la figure IV.4.

Au début de l’exécution le robot se trouve loin de l’obstacle et $\mu$ est fixé à 0. Le robot n’est alors piloté que par le correcteur III.10 permettant de réaliser la tâche de positionnement
IV.2. Résultats expérimentaux

IV.4.a : Évolution de la position du robot.

IV.4.b : Commande du robot.

IV.4.c : Information relative à l’obstacle.

Fig. IV.4 – Tache de positionnement en présence d’obstacle.
seule. Il commence donc par se diriger vers l’objectif. Une fois qu’il franchit l’enveloppe $\xi_0$, le robot mobile pénètre dans la zone d’influence de l’obstacle, et $\mu$ augmente progressivement.

Le robot est alors commandé par une combinaison linéaire des correcteurs défini par [II.7], réalisant la tâche de positionnement. Ici, la force répulsive n’est pas suffisante pour repousser le robot avant le franchissement de l’enveloppe $\xi_-$. $\mu$ passe donc à 1 et le véhicule n’est alors dirigé que par la commande d’évitement qui repousse le robot jusqu’à l’enveloppe de contournement $\xi_0$. Il est ensuite contraint à la suivre jusqu’à ce que l’obstacle ne présente plus de danger. Il peut alors naturellement quitter la zone d’influence de l’obstacle et converger vers l’objectif.

La tâche robotique imposée a donc été parfaitement réalisé, malgré la présence d’un obstacle sur sa trajectoire.

IV.2.2 La commande référencée vision

Nous présentons maintenant quelques résultats de concernant de la commande référencée vision. Bien que le module VSRP ait été complètement développé et soit prêt pour la validation, les expérimentations ont été retardées par des problèmes techniques survenues lors de la mise en place des nouvelles caméras couleur sur le robot. C’est pourquoi, nous ne présentons dans cette partie que les résultats de simulation réalisé sous Matlab.

Asservissement visuel seul

L’objectif maintenant est de positionner la caméra face à une cible composée de quatre points dans un environnement libre. Les figures IV.5 et IV.6 représentent respectivement la trajectoire du robot $4$ et l’évolution des indices visuels dans l’image.

![Evolution du robot](image)

**Fig. IV.5** – Évolution du robot en asservissement seul.

Nous avons défini une cible de référence (le cadre dans la figure IV.6) vers laquelle la caméra embarquée sur le robot doit converger. Comme nous pouvons le constater les indices visuels atteignent leur valeur de consigne, démontrant que la tâche est parfaitement réalisé.

La figure IV.7 donne les commandes appliquées sur le robot.

En l’absence d’obstacle les commandes d’évitement restent nulles, le véhicule n’étant alors piloté que par la loi de commande référencée vision.

---

$4$ Le robot n’est pas ici modélisé à l’échelle.
IV.2. Résultats expérimentaux

Fig. IV.6 – Évolution des indices visuels en asservissement visuel seul.

Fig. IV.7 – Commandes du robot en asservissement visuel seul.

Asservissement visuel et évitement d’obstacle

L’objectif de la tâche robotique envisagée est de positionner la caméra face à une cible en environnement contraint. Un obstacle circulaire de 25cm de rayon est positionné de manière
à entraver le déplacement du robot. L’enveloppe de contournement \( \xi_0 \) est fixée à 1m de l’obstacle, et les enveloppes \( \xi_- \) et \( \xi_+ \) positionnées à 10cm de part et d’autre de \( \xi_0 \). Les figures IV.8, IV.9 et IV.10 représentent respectivement la trajectoire du robot, l’évolution des indices visuels, et l’évolution du paramètre \( \mu \).

![Évolution du robot](image)

**Fig. IV.8** – Évolution du robot en asservissement en environnement contraint.

![Évolution des indices visuels](image)

**Fig. IV.9** – Évolution des indices visuels en asservissement visuel en environnement contraint.

Au début de la tâche, le robot se trouve loin de tout obstacle. \( \mu \) est fixé à 0 et le véhicule est alors guidé uniquement par la commande référencée vision. Il converge vers la cible sans prendre en compte l’obstacle présent. Dès qu’il franchit l’enveloppe de sécurité \( \xi_0 \), \( \mu \) augmente, ainsi que la force répulsive correspondante. \( \mu \) passe à 1 lorsque le robot franchit l’enveloppe \( \xi_- \). À cet instant, la base mobile et la platine sont commandées indépendamment, la première pour éviter l’obstacle, et la seconde pour compenser les mouvements de contournement et garder les indices visuels dans l’image. Le mouvement du robot est alors modifié de telle sorte que l’obstacle soit évité en suivant l’enveloppe de contournement \( \xi_0 \). Enfin, lorsque la condition de sortie est vérifiée, le robot est autorisé à quitter l’enveloppe de sécurité. \( \mu \) diminue progressivement jusqu’à s’annuler sur \( \xi_+ \). Le robot n’est alors plus piloté que par l’asservissement visuel et continue de converger vers la cible.
IV.2. Résultats expérimentaux

La figure IV.10 regroupe les différentes commandes appliquées sur le robot.

Les informations suivantes sont représentées sur la figure IV.11 :
– l’orientation de la base mobile $\theta$ ;
– l’orientation de la platine $\theta_{pl}$ ;
– l’angle entre le robot/l’obstacle $\alpha$ ;
– l’orientation de la force répulsive $\beta$ ;
– la distance à l’obstacle $d_\text{obst}$.

**Fig. IV.10** – Évolution de $\mu$.

**Fig. IV.11** – Données du robot en asservissement seul.
Enfin, la figure IV.12 regroupe les différentes commandes appliquées sur le robot, nous y retrouvons :

– en première ligne la commande global appliqué à la base mobile \((\omega, v)\) et la platine \((\omega_{pl})\);
– en seconde ligne la commande référencée vision ;
– en troisième ligne la commande d’évitement par la méthode des potentiels rotatif.

Nous pouvons noter que la commande global définit par une combinaison linéaire des lois de commande référencée vision et d’évitement d’obstacle, est peu important, interdisant toute saturation des actions. Les courbes des vitesses de rotation de la base mobile et de la platine mettent bien en évidence les différentes phase opératoire. Au début de l’exécution de la tâche, le robot étant excentré par rapport à la cible, la base mobile cherche à converger vers la cible, alors que la commande de la platine compense ses mouvements pour conserver les indices visuels dans l’image. Puis, un obstacle est détecté entraînant une variation brutal de la commande d’évitement, pendant que la loi d’asservissement visuel tente toujours de converger vers la cible. La combinaison linéaire de ses deux correcteurs permet alors d’assurer la continuité de la commande. Enfin, le robot parvient à s’extraire de l’obstacle et converge alors simplement vers la cible.
Conclusion et perspective

Au cours de ces travaux, nous avons présenté une technique de commande référencée multi-capteurs pour la navigation d’un robot mobile non-holonome en milieu encombré. L’originalité de cette méthode réside en l’utilisation conjointe du formalisme des fonctions de tâches et du formalisme des potentiels pour définir une commande référencée multi-capteurs permettant au robot mobile de réaliser une tâche de positionnement référencée vision dans un environnement encombré d’obstacles. Ainsi, dans un premier temps, nous avons présenté le formalisme et les principaux fondements pour la synthèse d’une commande référencée vision. Nous avons ensuite décrit le formalisme des fonctions des potentiels classiques ainsi que ses extensions, et montré comment l’utiliser pour définir une loi de commande réalisant l’évitement d’obstacle. Enfin, nous avons fusionné ces deux correcteurs pour définir la loi de commande référencée multi-capteurs.

Les méthodes présentées dans ce mémoire ont tout d’abord fait l’objet d’une validation en simulation sous matlab, avant d’être mises en œuvre expérimentalement sur un SuperScout II, étape indispensable pour entériner les travaux effectués. Bien que certaines expérimentations soient encore en cours, les résultats obtenus jusqu’ici sont plus qu’encourageants et ouvrent des perspectives prometteuses. Nous aurions également souhaité tester davantage de scénarii possibles, en prenant en compte un nombre d’obstacles plus important, ou encore en considérant des obstacles concaves. Il aurait été également intéressant de reconstruire la géométrie des obstacles à partir des données proximétriques pour générer ensuite l’enveloppe de contournement.
Nos travaux ont été volontairement axés sur la réalisation d’une tâche référencée vision canonique consistant à positionner la caméra face à une cible fixe dans un environnement contraint statique. Une extension naturelle consisterait à considérer un ensemble d’obstacles mobiles, ou encore à étendre la navigation référencée vision au cas du suivi de cible. De plus, il serait également souhaitable de s’affranchir des hypothèses contraignantes concernant la hauteur de la cible et des obstacles considérés, et de pouvoir prendre en compte les problèmes d’occultations éventuelles de la caméra.

Finalement, ce stage m’a permis de mettre en œuvre non seulement les connaissances acquises au cours de ma formation antérieure, mais aussi d’appliquer les enseignements dispensés au sein du DEA Systèmes Informatiques. A présent, je souhaite pouvoir poursuivre dans le domaine de la recherche en robotique mobile en continuant en thèse dans cette thématique.
Références bibliographiques


