Navigation semi-autonome d'un fauteuil roulant : franchissement de porte

Baptiste Brun

Master research Internship

Rapport de stage

Navigation semi-autonome d'un fauteuil roulant : franchissement de porte

Author :
Baptiste Brun

Supervisor :
Marie Babel
LAGADIC

Table des matières

Résumé

Dans ce rapport est abordé la navigation d'un véhicule non-holonome dans un environnement inconnu en utilisant l'asservissement visuel. L'environnement est analysé de manière géométrique, en utilisant principalement le point de fuite et la détection des portes. L'état de l'art portant sur la représentation de l'environnement, la localisation et la navigation sera présenté. Les méthodes décrites dans ce rapport permettent de contourner beaucoup de problèmes liés à la création de carte, et à la planification de trajectoire, aussi bien dans un environnement connu qu'inconnu. La navigation dans un couloir, et le positionnement devant une porte sont décrit en détail. Enfin les travaux restant à accomplir seront introduit.

Mots-clés

Asservissement visuel, suivie de ligne, fauteuil roulant, point de fuite, détection de porte, géométrie perspective, navigation, couloir, positionnement

Introduction

Durant mon stage j'ai travaillé sur un projet d'aide à la personne nommé APASH (Assitance au Pilotage pour l'Autonomie et la Sécurité des personnes Handicapées). La problématique qui m'a été confiée vise plus particulièrement le franchissement de portes, le but final est de pourvoir corriger la trajectoire de l'utilisateur afin que le passage se déroule sans encombre.

Dans la première partie de ce rapport je présenterai le projet APASH, puis dans une seconde partie nous verrons l'état de l'art lié à navigation et à la représentation de l'environnement. Dans la troisième partir nous verrons en détail les expériences auxquelles j'ai participé ainsi que les travaux que j'ai effectués durant mon stage, portant sur le suivie des lignes et l'asservissement visuel, j'exposerai à la fin de cette partie les travaux restant à accomplir. Enfin dans le dernier chapitre de ce rapport je conclurai.

Chapitre 1

Le projet APASH

L'objectif de ce projet est d'automatiser un fauteuil roulant motorisé afin d'une part, d'éviter les collisions, mais aussi d'assister la personne dans les manoeuvres pénibles qui demandent beaucoup d'attention. Ce type de fonctionnalité pourrait apporter un confort de vie non négligeable aux personnes souffrants de déficience motrice.

La navigation semi-autonome est une problématique importante qui a été abordée de nombreuse fois, car elle est intrinsèquement liée au développement de la robotique. Les solutions actuellement proposées utilisent des capteurs extrêmement chères et ne peuvent par conséquent qu'être employés dans des domaines restreints. Un défi supplémentaire de ce projet est de le rendre commercialement viable. Et par conséquent, les capteurs équipant le fauteuil roulant sont limités aux caméras et aux capteurs de proximités. Dès lors, la vision par ordinateur devient indispensable.

Ce projet a été monté par Marie Babel en 2012, en partenariat avec deux industriels situés en Bretagne. AdvanSEE, une entreprise spécialisée dans l'embarqué, qui nous fournis des caméras et qui s'occupera de matériel pour intégrer les logiciels que nous mettons au point sur le fauteuil. Ainsi que Ergovie une entreprise fournissant des fauteuils roulant, ce qui nous permet d'avoir un retour sur l'acceptabilité des solutions proposées. Ce projet devrait durer 2 ans.

1.1 Équipement du fauteuil

Le fauteuil roulant était équipé d'une seul caméra sur le côté gauche (figure 1.1) au début de mon stage. Une deuxième caméra a été installée à la fin de mon stage, mais elle n'est pas encore utilisée dans la partie logiciel. À terme, des capteurs de proximité seront ajoutés afin d'éviter toutes collisions.

Le fauteuil est équipé d'un système mis au point par François Pasteau, fonctionnant avec ROS (Robot Operating System), qui nous permet de commander le fauteuil, mais aussi d'obtenir les informations des capteurs par Wifi. Nous pouvons ainsi tester différent algorithmes facilement directement sur le fauteuil roulant.

1.2 Représentation du fauteuil

Les véhicules non-holonomes tel que les voitures ou les fauteuils roulant sont exposés à des contraintes dynamiques qui limitent leurs possibilités de mouvement. Par exemple, le fauteuil

FIGURE 1.1 – Fauteuil roulant utilisé dans le projet Apash. Nous pouvons distinguer la caméra sur l'avant gauche du fauteuil, ainsi que le boiter permettant de contrôler le fauteuil par Wifi à l'arrière.

ne peut pas se déplacer dans une direction correspondant à l'axe de ces roues.

Afin d'assurer la cohérence entre les calcules de trajectoire et les mouvements possibles du véhicule non-holonome, différents paramètres le concernant doivent être utilisés. Par exemple, dans l'article [17], le véhicule non-holonome est associé à son état $V = (x, y, \theta, \upsilon, \xi)$, où x et y correspondent à sa position dans le plan, θ à son angle, υ la vitesse et ξ l'orientation des roues (Figure 1.2).

Les valeurs de υ et ξ sont bornées non seulement pour couvrir les possibilités du véhicule, mais aussi pour éviter les manœuvres brutales, ces bornes fournissent ainsi un confort supplémentaire à l'utilisateur.

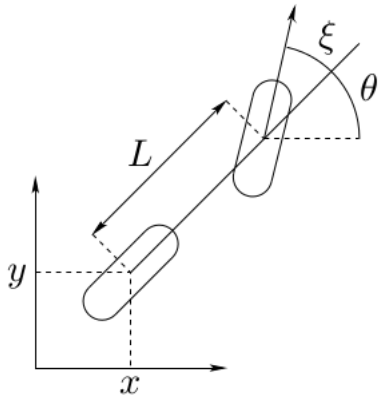


FIGURE 1.2 – Représentation des paramètres propre à un véhicule (Extrait de l'article [17])

Cette modélisation sera retenue pour le fauteuil du projet Apash. Ainsi le pilotage du fauteuil est possible en envoyant une commande $[\upsilon, \xi]$. Cette simplification nous permet de nous abstraire de la commande séparée des moteurs de chaque roues, tout en gardant les mêmes possibilités de

déplacement.

Chapitre 2

État de l'art

Dans cette partie, j'expliquerai les trois étapes nécessaires pour pouvoir franchir une porte. Nous verrons dans un premier temps plusieurs techniques de détection de porte, puis dans une seconde partie nous verrons différentes manières de représenter l'environnement afin de pouvoir se localiser et déterminer la position des portes. Ensuite dans une troisième partie, je présenterai des méthodes de planification de trajectoires. Enfin nous dresserons un bilan des éléments abordés.

2.1 Détection de portes

La détection des portes est une étape indispensable à la navigation autonome et semi-autonome en milieu intérieur inconnue. Pour ce type de navigation sans aprioris, elles peuvent être utilisées pour définir les zones navigables, mais aussi comme points de repère pour la localisation et l'orientation du système. Cette problématique a déjà fait l'objet de nombreuses recherches. Nous allons détailler dans cette partie deux grandes familles de méthodes proposées pour ce type de détection.

2.1.1 Détections par apprentissage

Comme pour la plupart des problématiques de détection liées à l'image, les méthodes d'apprentissage sont très utilisées. Elles permettent, à partir de plusieurs indices (classifieurs) de pouvoir prendre une décision. Dans ce cas précis cela nous permettrait de savoir si une hypothèse correspond ou non à une porte.

Nous pouvons citer par exemple l'approche de Hensler *et al.* [11], utilisant l'algorithme d'apprentissage *Ada-boost*. Les auteurs définissent une liste de classifieurs faibles pouvant être utilisés. Certains sont obtenues via l'utilisation d'un laser, à savoir :

- le renfoncement de la porte dans le murs,
- la largeur de la porte.

D'autres descripteurs sont obtenues en utilisant une caméra 2D, les principaux sont :

- Détection de la poignée de la porte
- La différence de couleur entre la porte et le mur
- L'espace sous la porte pour éviter la friction avec le sol

Une fois combinés, ces classifieurs permettent aux auteurs d'obtenir une taux de 72% de détection, avec un taux d'erreur de 0.008.

Une autre solution utilisant le même algorithme de *boosting*, mais uniquement basé sur la vision a été proposée en 2008 par Z. Chen *et al.* [6]. Ils obtiennent de meilleurs résultats en détectant 90% des portes, mais le taux de faux positifs s'élève à 0.05. Leur méthode se focalise sur

le bas de la porte pour la détecter. En plus des classifieurs définis précédemment, ils introduisent l'utilisation du point de fuite qui permet de tester la cohérence du positionnement de la porte dans l'environnement, en testant si les lignes de la porte sont bien orientées vers les points de fuite associés à l'environnement. Une amélioration de cet algorithme a été proposée en 2011 [7] par les mêmes auteurs. Une caméra calibrée leurs permet d'obtenir une estimation de la largeur de la porte. Ils réduisent ainsi le nombre de faux positifs. Le problème de ces algorithmes est qu'ils ne fonctionnent que sur des portes fermées, ils sont par conséquent peu adaptés aux tâches de navigation.

Une autre méthode proposée Anguelov *et al.* [1] se base sur le mouvement des portes pour les détecter. Si l'angle change conformément au mouvement d'une porte par rapport aux murs, entre deux balayages laser, celle-ci est alors détectée. Différentes caractéristiques associées à ces portes, telle que la couleur, sont alors enregistrées ou mises à jour via un algorithme de type EM (*Expectation Maximisation*). Ces paramètres sont ensuite utilisés pour détecter les portes qui ne sont pas en mouvement. Cette approche est capable de s'adapter à différents environnements, mais suppose la présence mouvement, ainsi qu'une certaine homogénéité de la coloration des portes.

Le principal problème des approches de Hensler *et al.* [11] et Anguelov *et al.* [1] réside dans le prix du dispositif laser, qui est un véritable frein pour le déploiement de cette technologie dans le domaine du handicap (fauteuil roulant), malgré de très bons résultats. La méthode de Z. Chen *et al.* [6] est très efficace, mais contraignante, d'une part sur la position de la caméra qui doit être très proche du sol pour détecter le bas de la porte, mais aussi lors de la détection qui s'effectue uniquement quand la porte est face à la caméra.

2.1.2 Détections par approche géométrique

En accord avec la volonté de mettre en place ce type de système à moindre coût, et permettant une détection sur une plus grande distance, des algorithmes de détection utilisant uniquement la vision ont été abordés.

Une méthode a notamment été proposée en 2007 par Murillo *et al.* [14] et est basée sur la détection de coin de *Harris*. L'utilisation de trois ou quatre coins permet de déterminer dans un premier temps un quadrilatère. Ensuite différents indices liés à la forme et à la couleur sont utilisés pour déterminer si la détection correspond à une porte ou non.

Une autre méthode, mise au point durant mon stage de première année de master [5] cherche à détecter des portes en utilisant principalement le point de fuite. Dans cette approche, une caméra possédant un grand angle de vu est utilisée. Cela permet de couvrir une zone large et d'acquérir plus d'informations. Cependant, l'image en résultant est distordue et doit être avant tout corrigée. Cet algorithme de détection se déroule en 3 étapes :

1. Extraction des lignes : On utilise l'algorithme Line Segment Detector (LSD) [19] qui procure une grande precision sur l'orientation de ces lignes.
2. Calcul des points de fuite : via une projection sur une sphère gaussienne similaire à celle proposée dans le projet ATIP [3] pour permettre que les points de fuite infinis sont détectés.
3. Détection des portes : les lignes verticales ne représente qu'un faible nombre de segments dans l'image et sont par conséquent utilisées pour les identifier. Le ratio $\frac{hauteur}{largeur}$ de la porte est ensuite calculé en utilisant la focale et les informations données par le point de fuite sur l'orientation du mur.

Les portes peuvent ainsi être repérées à des distances relativement grandes, comme le montre les exemples de la figure 2.1. De plus cette méthode présente l'avantage de détecter les portes ouvertes ou fermées. L'utilisation du point de fuite procure de forts a priori sur la position des portes dans l'image. Ces informations n'ont pas été suffisamment exploitées dans l'approche de Hensler [11].

FIGURE 2.1 – Exemples de détection géométrique basée sur la détection des lignes

Une récente amélioration de l'algorithme LSD appelé ELSD (Ellipse and Line Segment Detector) [18] qui permet de détecter aussi bien les lignes que les ellipses. Cette solution pourrait être adaptée pour identifier les portes directement sur une image distordue, de plus cela pourrait simplifier la détection du point de fuite, en évitant la phase projection sur la sphère. L'utilisation d'algorithmes d'apprentissage tels que ceux présentés en section 2.1.1 pourrait rendre les détections plus fiables. Ils devraient cependant s'adapter aux états ouverts ou fermés des portes, ou se baser uniquement sur les contours de ces dernières.

2.2 Représentation de l'environnement

La détection d'une porte n'est pas suffisante pour pouvoir se déplacer dans un environnement inconnu. Il est important de pouvoir d'abord se localiser. Or la localisation implique d'avoir une connaissance précise de l'environnement. Ces deux problèmes sont donc extrêmement liés et sont connues sous l'acronyme SLAM (*Simultaneous Localization And Mapping*) [22]. L'utilisation de points de repères ou de cartes devient alors d'une importance cruciale. Les environnements intérieurs, créés par l'homme, présente l'avantage d'avoir une structure géométrique particulière, qui peut être exploitée pour la création de carte. Dans cette partie, nous verrons dans un premier temps plusieurs modèles de représentation de cartes, puis différents moyens d'obtenir les informations nécessaires à la construction de celle-ci.

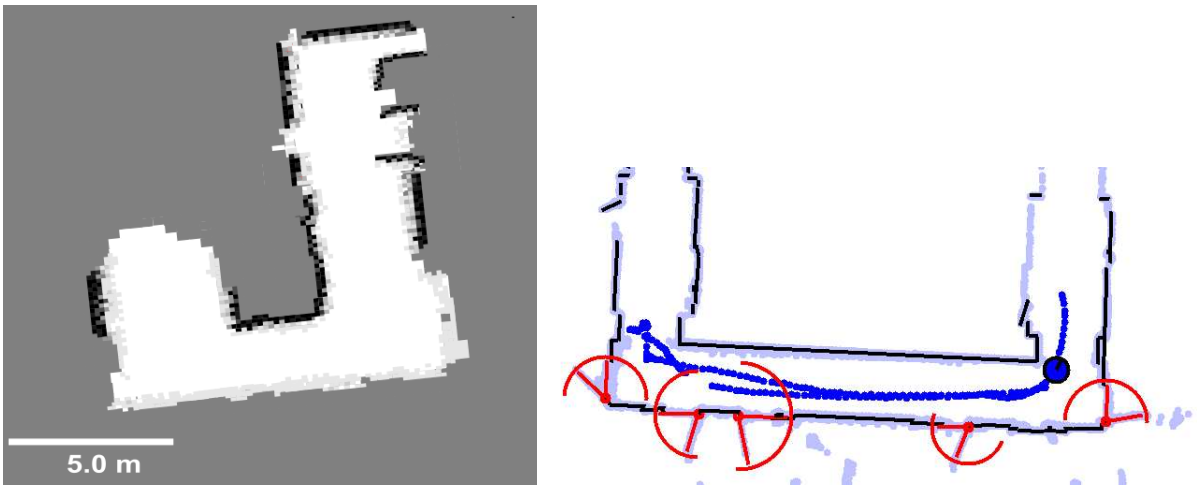
2.2.1 Modélisation d'une carte

Dans le but de pouvoir utiliser une carte dans d'autres algorithmes comme, la planification de trajectoire, mais aussi pour qu'elles puissent être misent à jour facilement. Il nous est alors nécessaire de définir un modèle de carte exploitable. Nous verrons dans cette section plusieurs modèles qui ont été proposées.

Une représentation de carte intéressante a été introduit par Anguelov *et al.* [1]. Deux classes d'objets sont utilisées, les portes et les murs. Chaque détection de ce type instancie une de ces classes. Ce mode de fonctionnement permet d'enregistrer plusieurs paramètres, comme la position des objets, mais aussi certains détails comme le sens d'ouverture d'une porte (Figure

2.2(b)). L'utilisation de ce genre de carte pourrait permettre de modéliser certaine forme de mouvement dynamique, mais les auteurs n'ont pas exploré cette piste. Cette approche permet, à terme, de modéliser une représentation précise de l'environnement, mais l'espace navigable n'est accessible uniquement quand la carte est terminée. Elle est donc difficilement exploitable par des tâches de planification de trajectoire qui sont sous la contrainte du temps réel.

La grille d'occupation (Figure 2.2(a)) permet de représenter une discrétisation de l'espace. Ce type de carte est utilisé par l'approche de Fulgenzi [10] ou encore celle de Santana [20]. Chaque cellule est mise à jour en fonction des détections effectuées. Cette technique est très simple à mettre en oeuvre et permet d'avoir rapidement une idée de l'espace navigable. De plus sa création par itération rend ce modèle approprié pour un fonctionnement en temps réel.



(a) (b)

FIGURE 2.2 – Représentation possible de carte de l'environnement. 2.2(a) Grille d'occupation établit par la méthode de Santana [20]. 2.2(b) Carte générée par l'approche d'Angoulov [1]

2.2.2 Acquisition de données et localisation

Pour acquérir les données relatives à la carte nous devons utiliser un capteur permettant d'obtenir la notion de distance tel qu'un laser télémétrique, un capteur à ultrason ou encore une caméra calibrée. Le principal problème de la création de carte se situe dans l'estimation du bruit dû à la mesure de ces distances. En effet l'erreur effectuée pendant la mesure est accumulée avec le temps et les résultats peuvent alors devenir aberrants comme l'illustre la figure 2.3.

Afin de prendre en compte cette erreur, des modèles probabilistes sont nécessaires. Ils permettent d'assurer la cohérence entre les différentes mesures effectuées, en prenant en compte le bruit. Comme, par exemple :

- *Extended Kalman Filter* (EKF), utilisé par [15], [8]
- Filtre Bayésien, utilisé par [10]
- *Expectation–maximization* (EM), utilisé par [1]

Un seul capteur n'est généralement pas suffisant. L'utilisation de l'odométrie est commune à presque toutes les approches [22]. Elle permet d'obtenir les mouvements du véhicule en mesurant, par exemple, les rotations effectuées par ses roues. Elle est utilisée de manière complémentaire

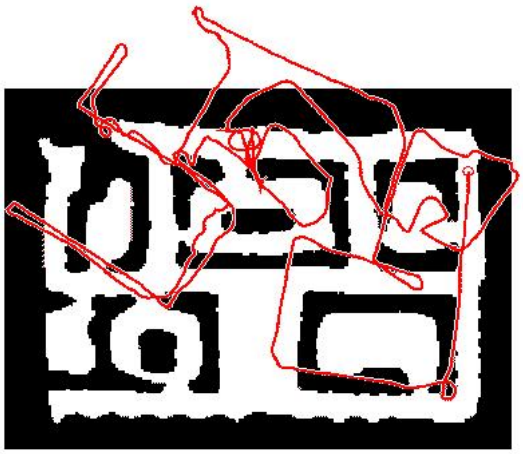


FIGURE 2.3 – Illustration de l'accumulation de l'erreur due au bruit dans la mesure. (Extrait de l'étude [22])

avec d'autres capteurs pour estimer la position du système.

Le laser télémétrique est ici encore très utilisé, car il procure des mesures très précises. L'approche de Fulgenzi *et al.* [10] et celle de Anguelov *et al.* [1] utilisent toutes deux un laser. La simple utilisation d'un modèle probabiliste, comme présenté précédemment, leurs permet de se localiser de manière presque immédiate grâce à sa grande precision de mesure. Bien que leurs modèles de représentation de carte diffèrent totalement.

Une méthode originale basée sur la détection des lignes verticales a été proposée en 1997 par J. Neira [15]. Les lignes verticales fournissent de précieuses informations sur les angles muraux et sur les diverses concavités présentes dans l'environnement. Un avantage important de cette méthode est qu'elle n'utilise qu'une seule caméra. Ce concept pourrait être amélioré en utilisant une caméra calibrée afin de pouvoir, en suivant ces lignes, déterminer à quelle distance elles se trouvent.

Mise au point dans le but d'estimer rapidement des mouvements totalement aléatoire sur les six degrés de liberté d'une caméra, la méthode monoSLAM [8] proposée en 2007 permet de repositionner des points d'intérêt dans un espace 3D. Un point d'intérêt et une région caractéristique facilement identifiable, comme un angle ou une texture particulière. L'algorithme s'exécute à une vitesse de 30 images par secondes, ce qui pourrait lui permettre d'être combiner à d'autres techniques dans le but d'affiner les résultats.

Une dernière méthode proposée par Santana *et al.* [20] basée elle aussi sur l'utilisation d'une seule caméra, mais cette fois-ci calibrée et limitée au niveau des degrés de liberté. Elle permet d'acquérir une carte de l'environnement sous forme de grille d'occupation. Bien que les détections ne soient effectuées qu'à une distance faible de la camera (un peu plus de 1,2 mètre, en accord avec l'orientation de la camera), cet algorithme présente l'avantage de pouvoir détecter des obstacles sous n'importe quelle forme. En isolant dans un premier temps le sol, puis en mesurant la distance à laquelle les objets ne correspondants pas au sol se situent.

Les méthodes proposées pour résoudre ce problème sont extrêmement nombreuses et variées.

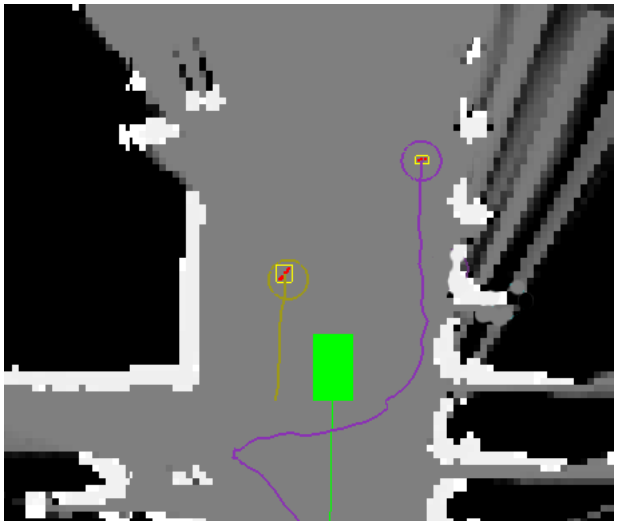


FIGURE 2.4 – Illustration de la grille d'occupation. Les objets statiques sont blancs et les dynamiques représentés en couleur par leurs cercles associés. Le système autonome au centre est en vert. (Extrait de l'article [10])

Il n'y a malheureusement pas de solutions universelles, ce qui implique que la méthode choisie doit être adaptée à nos besoins. Une fusion de différentes méthodes pourraient aussi être envisageable. Toutefois, la méthode de Santana [20] semble être la plus adaptée à notre problématique.

2.3 Planification de la trajectoire

Afin de pouvoir planifier une trajectoire sûre pour l'utilisateur, mais aussi pour les personnes qui l'entourent, nous devons prendre en compte tous les objets fixes ou mobiles présents dans l'environnement. Nous supposerons ici que nous avons connaissance de la carte de l'environnement ainsi que la localisation d'un endroit cible à atteindre. Nous verrons dans cette partie l'estimation de collisions et ensuite comment est planifiée une trajectoire.

2.3.1 Estimation des collisions

L'estimation des collisions est une étape indispensable à la sécurité de l'utilisateur. On distingue deux types de collisions, celles avec des objets statiques et celles avec des objets dynamiques.

Collision avec un objet statique

Dans un premier temps nous devons être capable de prendre en compte les objets statiques. La solution proposée par l'approche de Fulgenzi [10] utilise une grille d'occupation, comme nous l'avons vu dans la section précédente. À chaque case de cette grille sont associés différents attributs correspondant aux risques associés à l'espace couvert par cette dernière. Typiquement le risque de collision avec un object statique, comme un mur, est directement lié à la présence ou non de ce type d'obstacle sur la carte (Figure 2.4).

Collision avec un objet dynamique

L'estimation du risque de collision avec un objet dynamique implique la détection puis le suivi de chaque objet. Ainsi plusieurs paramètres liés à cet objet, peuvent être évalués suivant :

- la position (x, y) sur le plan,
- l'orientation θ,
- la vitesse de l'objet.

Une fois ces paramètres estimés, des algorithmes probabilistes permettent de calculer la trajectoire la plus vraisemblable que suivront ces objets. Dans l'approche de Fulgenzi [10] Ces paramètres permettent de définir une chaîne de Markov. La prédiction des mouvements de ces objets est ensuite approchée par l'utilisation de Modèles Cachées de Markov.
L'approche de Bennewitz *et al.* [2] utilise l'algorithme EM sur l'ensemble des trajectoires possibles afin d'obtenir la direction dominante.

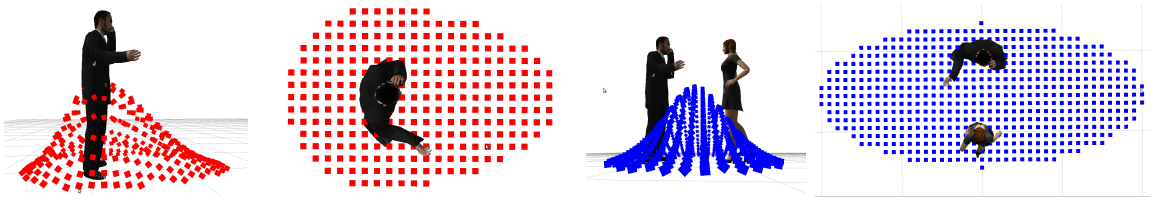
2.3.2 Environnement peuplé d'humains

Comme l'application visée est la navigation d'un fauteuil roulant celui-ci devra avoir un comportement socialement correct afin de ne pas mettre mal à l'aise l'utilisateur. Vasquez *et al.* [23] suggèrent de prendre en compte cette contrainte supplémentaire et proposent un filtre social. Les auteurs utilisent ainsi deux notions définit par des sociologues :

- *personal space*, correspondant à l'espace de confort social d'une personne seule (Figures 2.5(b), 2.5(a)),
- *o-space*, qui s'applique cette fois-ci à des personnes en conversation, l'espace définit est réservé uniquement aux nouveaux participants (Figures 2.5(d), 2.5(c)).

Les objets dynamiques correspondant à des humains sont alors approchés en considérant ces deux espaces, ainsi les comportements inconfortables pour l'utilisateur sont évités.

Cette approche, jugée encore expérimentale par les auteurs, correspond aux objectifs du projet APASH et pourrait apporter un confort supplémentaire à l'utilisateur.



(a) (b) (c) (d)

FIGURE 2.5 – Représentation des filtres sociaux associés aux personnes en communication 2.5(d), 2.5(c) ou seuls 2.5(b) 2.5(a) [23]

2.3.3 Calcul de la trajectoire

Nous avons défini et calculé au préalable les différentes probabilités de collision, les éventuelles espaces sociaux correspondant aux personnes présentes. Nous avons donc une connaissance très précise de l'environnement. Nous pouvons alors calculer la trajectoire à suivre.

La planification de trajectoire s'effectue très bien en utilisant un algorithme appelé *Rapidly-exploring Random Tree* (RRT) [12]. Il permet de calculer rapidement un grand nombre de tra-

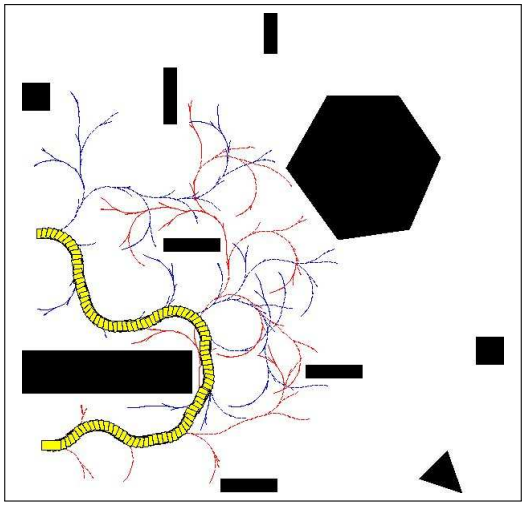


FIGURE 2.6 – Représentation d'un arbre des trajectoires construit par RRT. Figure extraite de l'article [9]

jectoires possibles, de manière incrémentale. L'ensemble des trajectoires forme un arbre, comme le montre la figure 2.6.

Initialement l'algorithme RRT ne calcule que des trajectoires globales, autrement dit, le chemin complet depuis l'état initial jusqu'à l'état final. L'approche globale n'est pas adaptée pour la planification de trajectoire en environnement dynamique, et ce pour deux raisons. La première est que le temps de calcul de la planification totale de la trajectoire serait trop long, et ne respecterait donc pas les contraintes de temps réel liées au système visé. La deuxième raison est liée à l'environnement dynamique, qui évolue au fil du temps : la détection des collisions probables avec les objets en mouvement sur une longue durée devient alors extrêmement difficile voir impossible.

Dans l'article [17], l'algorithme RRT a été adapté pour la planification partielle de trajectoire. Pour y parvenir, une contrainte de temps δ a été ajoutée, limitant le temps de création de l'arbre. Une fois le temps δ écoulé, Le nœud arrivant au plus proche de l'état cible, avec la meilleure probabilité est sélectionnée, puis la trajectoire est calculée en remontant de ce nœud jusqu'à la racine de l'arbre.

Le but est alors de construire un arbre cohérent avec les risques de collision précédemment calculés. Afin de ne pas se retrouver dans une situation où le fauteuil serait bloqué, sans possibilité d'évitement de l'obstacle, les auteurs des articles [10, 17] utilisent un concept appelé *Inevitable Collision States* (ICS) [9]. Un état est dit ICS, si tous les chemins possibles à partir de cet état correspondent à une collision. Ceci nous amène à la définition d'un chemin ICS-*free* suivante : si pour tous les états s associés à un temps $t \in [t_{initial}; t_{final}]$ d'un chemin ϕ, $s(t)$ n'est pas ICS. En accord avec ces définitions, seuls les nœuds non ICS sont gardés sur l'arbre. Ainsi tous les chemins décrits par cet arbre sont ICS-*free*.

Ensuite, le système commence à se déplacer vers la cible, en mettant à jour la carte de l'environnement (objets statiques et dynamiques) et réitère l'opération de calcul de trajectoire. Ces calculs s'effectuent pendant le déplacement du système afin de fournir une certaine fluidité lors de l'enchainement des mouvements. L'opération est répétée jusqu'à l'arrivée à l'état final.

2.4 Bilan

Nous avons vu dans ce chapitre toute la chaîne d'action à accomplir, afin de franchir une porte. Pour la détection des portes, l'algorithme proposé reste à améliorer afin de le rendre plus robuste, en utilisant par exemple un algorithme d'apprentissage, mais aussi à optimiser afin de minimiser le temps de calcul. Typiquement le calcul des points de fuite pourrait être grandement accéléré en utilisant par exemple un a priori sur sa position dans l'image, qui est intrinsèquement lié position de la caméra, mais cela le rendrait plus instable. D'autres solutions devront être étudiées pour le calculer.

La représentation de l'environnement est un processus extrêmement lourd qui gagnerai à être simplifié. Dans le cas du projet APASH, nous devrons nous concentrer uniquement sur les éléments essentiels afin de minimiser le temps de calcul, et de pouvoir s'adapter à des environnements intérieurs divers et variés

La planification de trajectoire est aussi problématique, car elle peut difficilement être appliquée à la navigation semi-autonome. De plus, sans une connaissance de l'environnement précise ce type de méthode peut s'avérer chaotique.

Enfin le calcul de la trajectoire en prenant en compte les filtres sociaux [23], qui viennent à peine d'être introduits, dans une solution "grand public" serait une véritable innovation dans le domaine.

Chapitre 3

Planification de trajectoire par asservissement visuel

Dans cette partie, nous verrons les bases de l'asservissement visuel, essentielle à la compréhension de notre approche de planification de trajectoire. Dans un premier temps nous verrons le fonctionnement de base d'un algorithme d'asservissement, puis nous aborderons la gestion de l'erreur et de la commande, et enfin nous verrons les différents avantages et inconvénient de ces techniques.

3.1 Introduction à l'asservissement

L'asservissement est une méthode utilisée dans de nombreux domaines. Le but est de faire correspondre une valeur mesurée à l'aide d'un capteur, à une valeur désirée. L'erreur entre la valeur mesurée et la valeur désirée permet de calculer une commande à appliquer à un actionneur pour se rapprocher de la valeur voulue. Dans un four par exemple, la régulation de la température s'effectue par ce type de méthode, un capteur de température est utilisé pour déterminer l'erreur par rapport à la consigne de l'utilisateur, et les résistances sont commandées de manière à minimiser cette dernière. La figure 3.1 illustre le fonctionnement d'un algorithme d'asservissement simple.

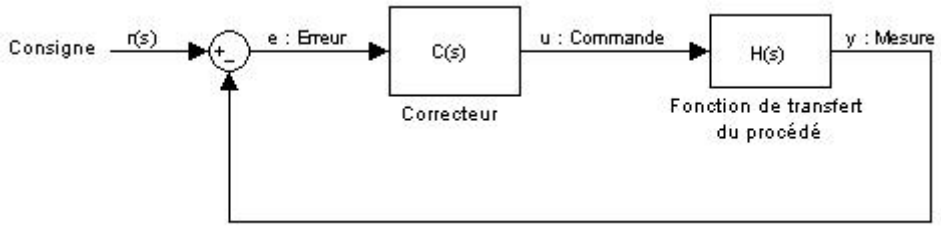


FIGURE 3.1 – Schéma d'une boucle d'asservissement

3.2 Gestion de l'erreur en asservissement visuel

Dans le cas de l'asservissement visuel, le retour s'effectue par une analyse de l'image. L'utilisation d'une caméra permet de calculer un grand nombre de mesures, l'erreur est alors représentée

sous forme de vecteur $\mathbf{e}$,

$$\mathbf{e} = \mathbf{s} - \mathbf{s}_d, \tag{3.1}$$

où $\mathbf{s}_d$ est le vecteur associé aux caractéristiques visuelles désirées, et $\mathbf{s}$ celles mesurées.

Tout l'art de l'asservissement se situe donc dans la gestion de l'erreur. La matrice d'interaction $\mathbf{L}_s$ définit la relation entre l'erreur et la commande. Elle définit les actions possibles, associés à chaque erreur. Ainsi la commande $\mathbf{c}$ est obtenue suivant,

$$\mathbf{c} = -\lambda \mathbf{L}_s^+ \times \mathbf{e} \tag{3.2}$$

où λ est une constante permettant de faire varier la sensibilité aux erreurs, et $\mathbf{L}_s^+$ la pseudo inverse de la matrice d'interaction. Ce cas simple permettrai de contrôler tout les degrés de liberté possible.

Afin de limiter les degrés de liberté à ceux que nous pouvons utiliser, nous définissons une matrice ${}^r\mathbf{J}_r$ de dimensions $n \times 6$ où n est la dimension du vecteur commande c (définit dans la section 1.2). Dans le cas du fauteuil roulant ${}^r\mathbf{J}_r$ est défini suivant,

$${}^r\mathbf{J}_r = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{3.3}$$

où la première colonne associe à la commande de vitesse v au degré de liberté lié à la translation sur l'axe x, et la deuxième lie la rotation sur l'axe z à ξ.

De plus, différent repères doivent donc être pris en compte pour calculer la commande. En effet la repère de la camera n'est pas le mêmes que celui du fauteuil situé entre les deux roues motrice (Figure 3.2). Les vecteurs de translation ${}^c\mathbf{t}_r$ et de rotation ${}^c\mathbf{r}_r$ sont défini suivant,

$${}^c\mathbf{r}_r = \begin{bmatrix} \pi/2 \\ 0 \\ -\pi/2 \end{bmatrix} {}^c\mathbf{t}_r = \begin{bmatrix} w \\ 0 \\ -l \end{bmatrix} \tag{3.4}$$

Ces vecteurs permettent de calculer une matrice 6×6 qui associe les vitesse détectées dans le plan caméra à des vitesse dans le plan robot,

$${}^c\mathbf{W}_r = \begin{bmatrix} {}^c\mathbf{R}_r & [{}^c\mathbf{t}_r]_\times^c \mathbf{R}_r \\ 0_{3\times3} & {}^c\mathbf{R}_r \end{bmatrix} \tag{3.5}$$

L'équation permettant de calculer une commande pour le fauteuil roulant devient alors,

$$\mathbf{c} = -\lambda * (\mathbf{L}_s \times^c \mathbf{W}_r \times^r \mathbf{J}_r)^+ \times \mathbf{e} \tag{3.6}$$

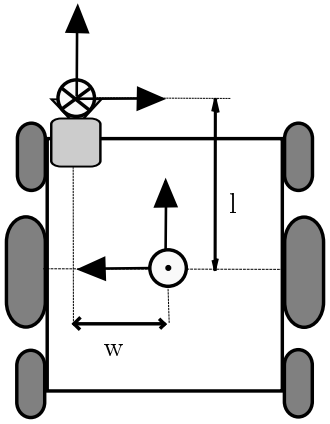


FIGURE 3.2 – Représentation des repères du fauteuil roulant et de la caméra

3.3 Avantages et inconvénient de l'asservissement

L'asservissement visuel présente l'avantage d'être simple à mettre en place, et demande peu de puissance de calcule. Autre avantage considérable, dans une application au déplacement d'un robot, est que ce type d'algorithme ne nécessite aucune carte complète de l'environnement, seule certaines caractéristiques visuelles dans l'image sont utilisées.

Cependant, plusieurs problèmes sont liés à cette technique. Le premier est lié à la fréquence d'acquisition des mesures. En effet lorsqu'une commande est calculée elle est appliquée jusqu'à l'arrivée de la commande suivante. Ainsi si le temps d'acquisition de l'image et/ou le temps de calcule des caractéristiques visuelles est trop long, cela pourrait être d'une part dangereux pour l'utilisateur, mais aussi nous éloigner de l'objectif initial. Il est donc extrêmement important de garder une fréquence d'acquisition élevée.

Un autre problème concerne la précision des actionneurs. Les moteurs du fauteuil roulant par exemple n'ont pas une précision infinie. Par exemple, s'il manque une rotation de 1° pour atteindre un objectif, mais que le fauteuil ne peut pas effectuer une rotation inférieure à 2°, il y aura une oscillation autour de l'objectif à atteindre. Ces oscillations peuvent être minimisées grâce à la constante λ définit dans l'équation 3.2.

Chapitre 4

Déplacement du fauteuil

Comme nous l'avons vu l'asservissement visuel est un outil puissant, pouvant être adapté à beaucoup de problématique. Nous avons opté pour une solution basée sur l'asservissement visuel pour la planification de trajectoire. Cela nous permet de garder un maximum de possibilité de déplacement. Les approches telles que celle de S. Petti [17] utilisant un arbre des trajectoires possibles ne sont pas robuste à la déviation de trajectoire. Avec ces méthodes, si une erreur est induite lors du déplacement, l'arbre des trajectoires doit-être recalculé, ce qui est très coûteux en temps et en puissance de calcul. L'approche que nous proposons est beaucoup plus flexible sur ce point, ce qui nous permet d'agir rapidement sur fauteuil quelle que soit sa position.

Nous allons présenter dans ce chapitre les différentes méthodes d'asservissement qui ont été appliquées au fauteuil roulant. Dans la première partie nous verrons une méthode permettant le déplacement du fauteuil dans un couloir. Puis dans une seconde partie celle permettant de positionner le fauteuil devant une porte. Enfin dans la dernière partie de ce chapitre nous passerons en revu les différentes améliorations et idées qui pourraient permettre au projet d'être plus polyvalent et efficace.

4.1 Déplacement dans un couloir

Ce déplacer dans un couloir sans utiliser de carte n'est pas une tâche facile. Nous avons défini plusieurs caractéristiques visuelles pouvant être calculé à partir de l'image, et permettant de nous positionner facilement dans tout type de couloir, que nous détaillerons dans les parties suivantes. Les travaux initié par R. Vassallo en 1998 [24] ont été réutilisées dans notre approche [16].

4.1.1 Utilisation du point de fuite

Le point de fuite est un élément essentiel, il apporte une information précise sur l'orientation du couloir. Il va nous permettre de calculer l'angle entre l'orientation du fauteuil, et les murs. En effet le point de fuite est le résultat d'une projection en perspective de lignes parallèles. Ainsi, comme l'illustre la figure 4.1 la ligne formée par le point de fuite et le centre optique de la camera permet de calculer un angle ϕ correspondant à l'orientation du fauteuil par rapport au couloir (où tout autre objet ayant contribué à ce même point de fuite).

Cette information pourrai donc être utilisée pour orienter le fauteuil roulant dans la bonne direction. Cependant, elle n'est pas suffisante, car elle ne permet pas de savoir où se trouve le fauteuil roulant dans le couloir. Plus précisément, tant que l'angle entre le fauteuil et les murs reste identique, le point de fuite apparaîtra au même endroit dans l'image, peu importe

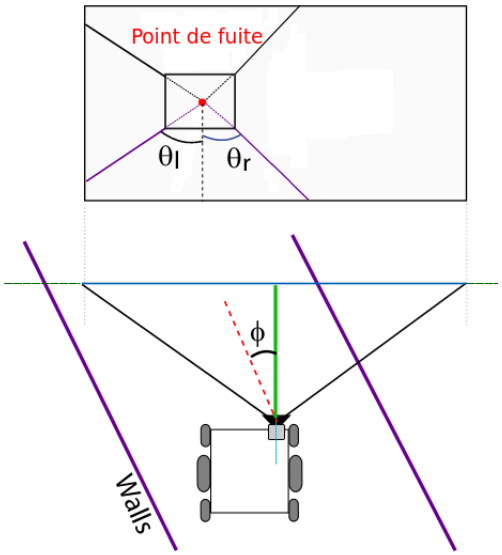


FIGURE 4.1 – Utilisation du point de fuite et des lignes du couloir pour le positionnement

l'emplacement du fauteuil. De ce fait une information supplémentaire doit être calculée pour éviter toutes collisions où fortement avec les murs.

4.1.2 Utilisation des lignes du couloir

Afin de pouvoir se positionner dans un couloir, nous utiliserons certaines lignes caractéristiques. Ces lignes séparant le mur du sol nous permettent d'avoir une idée de la position du fauteuil. Les angles θ_l et θ_r présent sur la figure 4.1 nous permettent de calculer l'angle de la médiane θ_m. Cette valeur nous indique la distance entre les murs avec la position du fauteuil, calculé de la manière suivante :

$$\theta_m = \arctan(\frac{\theta_l + \theta_r}{2}). \tag{4.1}$$

Si l'angle θ_m correspond à $\frac{\pi}{2}$ (la médiane est verticale) cela signifie que la camera se situe au milieu du couloir.

La détection des lignes du couloir est effectuée lors du calcul du point de fuite. Chaque ligne votant pour un point de fuite vote aussi pour un angle associé à leurs directions. Les angles ayant reçu le plus de vote en dessous du point de fuite à gauche et à droite sont la plupart du temps associés aux lignes séparatrices sol/murs d'un couloir. Cependant, nous ne pouvons pas nous permettre de calculer le point de fuite à chaque image, notamment à cause de la détection des lignes. L'algorithme LSD [19], bien qu'il soit rapide dans sa catégorie ne nous permet pas d'effectuer un asservissement à une fréquence correct.

4.1.3 Suivie des lignes du couloir

Afin d'améliorer la rapidité d'acquisition des données indispensables pour évaluer notre position dans le couloir, nous avons utilisé un algorithme de suivi de ligne appelé *Moving Edge* (ME) [4]. Une implémentation de cet algorithme est présente dans la bibliothèque libre ViSP [13] développée par l'équipe projet LAGADIC.

Toutes les lignes ayant votées pour un angle sur le point de fuite sont alors *trackées*. Les angle θ_l et θ_d sont alors obtenue en faisant la moyenne des angles des lignes suivies de chaque coté du couloir.

Fonctionnement du *tracker* de ligne

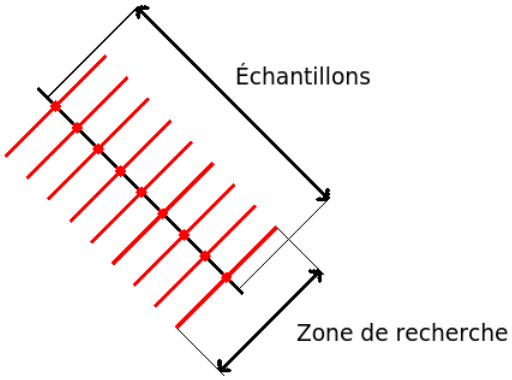


FIGURE 4.2 – Représentation des échantillons et de leurs zone de recherche associée

Le suivie d'une ligne s'effectue en deux étapes.

La première est l'initialisation, une ligne déjà identifiée est fournie au *tracker*. Cette ligne va être échantillonnée en n point. Pour chacun de ces points le contraste, l'orientation du gradient sur un masque vont être calculés.

La deuxième phase consiste à suivre ces points d'une image I à une image $I+1$. Pour chacun des points mémorisés, une zone de recherche est défini perpendiculairement à la ligne de l'image I. Un produit de convolution, entre les points de la zone de recherche de l'image $I+1$ et le point mémorisé de I, nous permet de tester la vraisemblance des deux points. Ainsi le point sera identifié dans l'image, ou déclaré comme perdu si aucun résultat n'est convenable.

Une fois tous les points identifiés, la méthode des moindre carrée est utilisée pour retrouver la ligne. Les extrémités sont mises à jour s'il y a eu une perte de point. La deuxième phase peut alors être réitéré pour l'image suivante. Si le nombre dde point restant est trop faible la ligne peux être re-échantillonnée.

Remarques sur le suivie des lignes

L'algorithme de suivie de lignes basé sur ME est très efficace lorsque la ligne est définie par un fort contraste. Mais ce n'est pas toujours le cas des lignes détectées avec l'algorithme LSD ce qui entraîne parfois des erreurs lors du suivie, car si la ligne est initialisée sur une zone presque uniforme, où le contraste est dû en partie au bruit dans l'image, il sera alors très difficile de la suivre. De plus cet algorithme est sensible au changement d'illumination, ce qui peut entraîner la perte de certaines lignes lors d'un déplacement.

Remarques sur l'utilisation des lignes suivies pour l'asservissement visuelles

L'utilisation des lignes suivies comme caractéristiques visuelles implique certaine problématique supplémentaires. Concernant la précisions des lignes du couloir, mais aussi celle du point de fuite qui sera dès lors calculé uniquement à partir de deux lignes. Nous devons alors nous assurer que ces lignes ne divergent pas. Cela impliquerai une erreur importante au niveau du positionnement du fauteuil dans le couloir. Nous vérifions alors la cohérence des lignes suivies pour chaque angle. Si une ligne diverge, nous réinitialisons l'algorithme de suivie en recalculant le point de fuite et les lignes du couloir. Une autre manière de déterminer si l'algorithme de suivie diverge est de vérifier la hauteur du point de fuite. En effet comme l'angle entre le sol et l'axe optique de la camera est invariant, le point de fuite devrait rester à une hauteur constante. Si ce n'est pas le cas, la ré-initialisation du suivie est lancée. Enfin pour éviter qu'une erreur éventuel ait trop d'impact, l'algorithme de suivie est réinitialisé périodiquement après un certain nombre d'images traitées (autour de 10 lors des expérimentations).

4.1.4 Asservissement visuel dans un couloir

Nous avons ici les deux caractéristiques visuelles nécessaires au positionnement du fauteuil dans le couloir. Nous pouvons dès lors commencer la phase d'asservissement visuel, avec θ_m pour la position du fauteuil et l'abscisse du point de fuite pour sont angle. Nous allons utiliser la ligne d'angle θ_m passant par le point de fuite. L'angle de la ligne désirée passe par le centre de l'image, afin de s'assurer que le fauteuil soit droit dans le couloir (figure 4.3. L'angle θ_d de cette ligne nous permet de définir la position du fauteuil dans le couloir. Si cette ligne correspond à une ligne verticale, alors la caméra viendra se positionner au milieu du couloir.

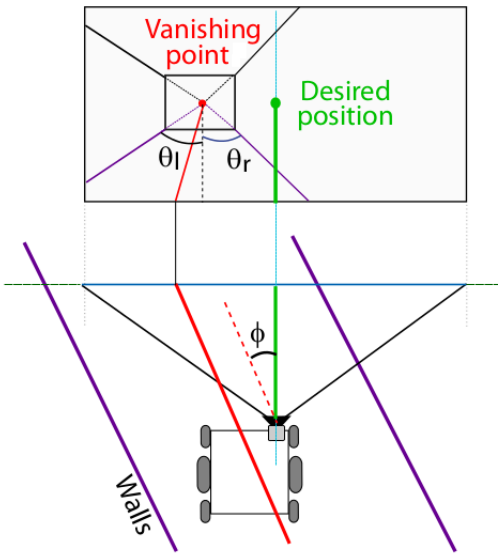


FIGURE 4.3 – Utilisation du point de fuite et des lignes du couloir pour le positionnement

Les lignes sont représentées sous la forme (ρ, θ) où ρ est la distance de la perpendiculaire à la ligne passant par le point en haut à gauche de l'image, et θ son angle (figure 4.4).

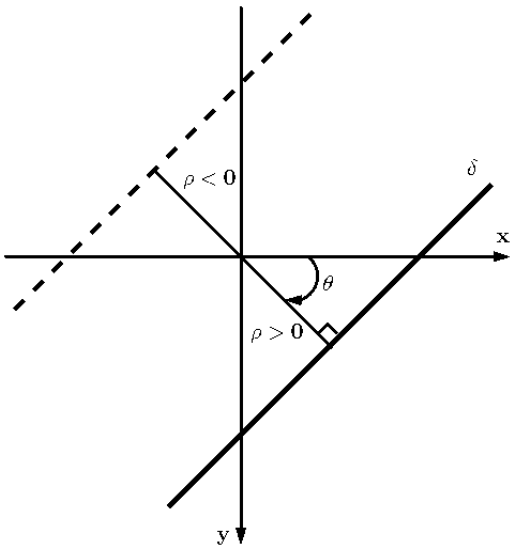


FIGURE 4.4 – Representation d'un ligne en (ρ, θ)

Comme nous l'avons vu dans la partie 3.2, nous devons mettre au point la matrice d'interaction Ls de dimension 2×6 pour pouvoir associer les deux erreurs aux 6 degrés de liberté possible.

Pour calculer cette matrice nous devons définir le plan dans lequel se trouve les caractéristiques visuelles. Dans notre cas la ligne formée par le point de fuite et l'angle θ_m se situe au niveau du sol, et est donc simplement définie par la hauteur de la caméra h. L'équation générale d'un plan 3D est sous la forme suivante,

$$AX + BY + CZ + D = 0. \tag{4.2}$$

Notre plan est alors défini par :

$$BY = h \tag{4.3}$$

À partir de cette équation du plan nous pouvons définir la matrice d'interaction, suivant

$$L_s = \begin{bmatrix} L_\rho \\ L_\theta \end{bmatrix} = \begin{bmatrix} \lambda_\rho \cos(\theta) & \lambda_\rho \sin(\theta) & -\lambda_\rho \rho & (1+\rho^2)\sin(\theta) & -(1+\rho^2)\cos(\theta) & 0 \\ \lambda_\theta \cos(\theta) & \lambda_\theta \sin(\theta) & -\lambda_\theta \rho & -\rho\cos(\theta) & -\rho\sin(\theta) & -1 \end{bmatrix}, \tag{4.4}$$

où

$$\lambda_\rho = \frac{\rho \sin(\theta)}{-h}, \tag{4.5}$$

et

$$\lambda_\theta = \frac{cos(\theta_m)}{h}, \tag{4.6}$$

Grâce à cette matrice nous allons pouvoir générer la commande $\mathbf{c} = [v, \omega]$. Cependant, nous n'utiliserons que la vitesse angulaire ω et nous fixerons v à une vitesse fixe, pour éviter que le fauteuil s'arrête une fois bien positionné dans le couloir. De cette manière, nous pouvons en parcourir la totalité.

4.1.5 Résultat

Grâce à l'ajout d'un laser télémétrique sur le fauteuil roulant, nous tester la précision du positionnement dans la couloir. La figure 4.5 nous montre la position angulaire du fauteuil dans le couloir. Le premier pic sur la courbe est dû à la position de départ du fauteuil, qui était droit, mais proche d'un des murs, une rotation a dû être effectuée pour qu'il puisse se déplacer vers le centre.

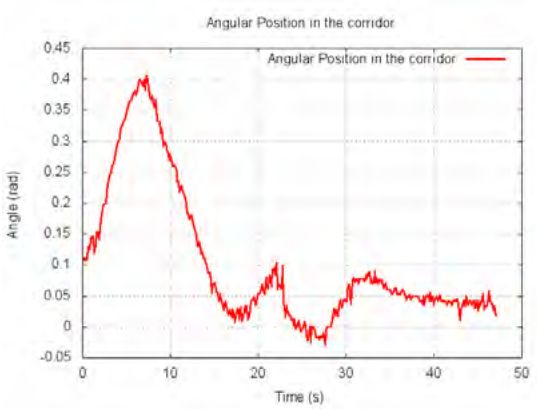


FIGURE 4.5 – Angle du fauteuil roulant dans le couloir

La figure 4.6 nous montre quant à elle la position du fauteuil dans le couloir 0 étant le centre du couloir, on peut constater que la courbe décrois plus rapidement que la précédente, ceci est dû au mouvement non-holonome du fauteuil. Pour que les deux caractéristiques visuelles puissent être asservie, le bon positionnement du fauteuil dans le couloir doit-être prioritaire.

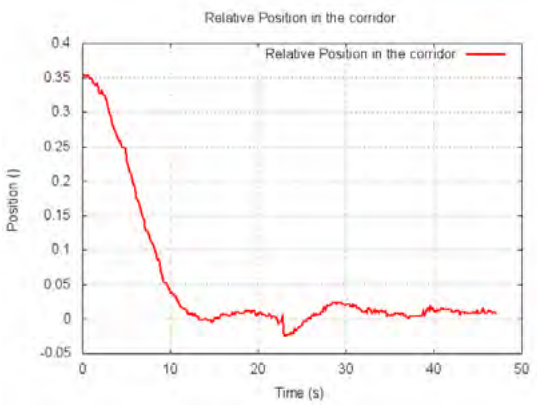


FIGURE 4.6 – position dans le couloir

4.2 Positionnement devant un porte

La trajectoire qui doit être suivie pour positionner le fauteuil devant une porte n'est pas des plus intuitive. En effet, le fauteuil étant un véhicule non-holonome, ces possibilités de déplace-

ment sont limitées. Il faut parfois avoir un angle d'approche suffisamment grand afin d'éviter de se retrouver bloqué devant la porte sans possibilité de la franchir. La figure 4.7 illustre ce problème.

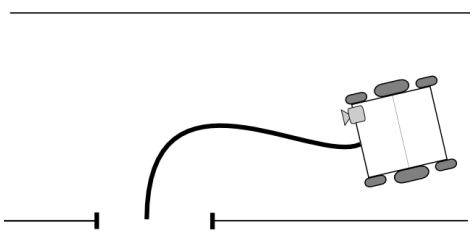


FIGURE 4.7 – Exemple de trajectoire possible

L'utilisation d'un algorithme de planification de trajectoire, tel que RRT, est certainement la manière la plus intuitive pour surmonter ce type de difficulté. Cependant, l'utilisation de l'asservissement visuel se montre là aussi très efficace.

4.2.1 Détection des portes

Nous avons dans un premier temps besoin de détecter les portes dans l'image. Pour y arriver nous utiliserons l'algorithme de Rafik Sekkal [21] se basant sur celui élaboré lors de mon stage de master 1 [5] décrit dans la section 2.1.2. La différence réside principalement sur les lignes utilisées pour la détection, la ligne correspondant au haut de la porte n'est plus prise en compte. Cette ligne était problématique, car si elle était occultée, ou mal détectée, la porte n'était pas reconnue. Ainsi dans l'approche de Rafik [21] vérifie uniquement la cohérence entre la hauteur d'un segment et la distance qui le sépare d'un autre. Si le ratio $\frac{hauteur}{distance}$ correspond à un ratio de porte, elle est mémorisée. Cette méthode permet de détecter un nombre plus important de porte, mais le taux de faux positifs est aussi plus élevé.

4.2.2 Suivie des portes

Dans un premier temps la méthode de suivie de ligne utilisée était basée sur l'algorithme EM [4]. Mais les lignes des montants de porte sont souvent sujet à des changements d'illumination, notamment lorsque la porte est ouverte et que nous découvrons la pièce en avançant. Ce phénomène provoque un changement d'orientation du gradient. De plus la légère distortion présente dans l'image déforme les lignes verticales ce qui les rend plus difficiles à suivre. Sous ces conditions l'algorithme EM était très instable.

Nous avons alors proposé une autre approche, exploitant au mieux les particularités de ces lignes, à savoir :

- Elles sont toujours verticales.
- Elles sont toujours visible à la hauteur du point de fuite.
- La couleur du montant côté couloir est moins exposé aux variations.

Fonctionnement de l'algorithme de suivi des montant de porte

Nous travaillerons ici dans une bande de l'image située au niveau du point de fuite, comme le montre la figure 4.8. Cela permet d'une part de réduite le temps de calcul, mais cela permet

aussi de ne pas encombrer l'algorithme de suivie avec la gestion des extrémités des segments. De plus nous considérerons toutes les lignes suivies comme verticale. Cet algorithme utilise principalement le gradient de l'image, plus particulièrement sur la dérivée partielle en x de l'image.

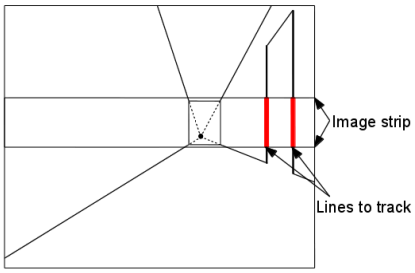


FIGURE 4.8 – Bande utilisée pour le suivie des montants.

Lors de la phase d'initialisation, nous calculons un vecteur g, correspondant au gradient le plus fort sur sur chaque ligne de la bande. Les meilleurs gradients sont sélectionnés sur une largeur $2 \times nbcol$ centrée sur la ligne à suivre, afin de supporter un minimum de variation d'angle (figure 4.9). Ce vecteur de gradient sera utilisé comme un descripteur de ligne. En plus de ce vecteur, la couleur moyenne du côté extérieur de la porte est mémorisée. Cela nous permettra d'avoir un suivie plus robuste, notamment lors d'un changement d'orientation du gradient.

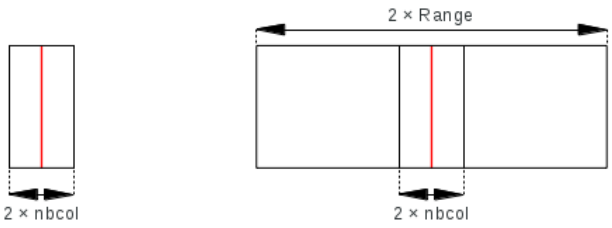


FIGURE 4.9 – Largeur utiliser pour le calcule du vecteur $\mathbf{g}$ et largeur de la bande de recherche.

Lors de la phase de suivie, nous définissons une zone de recherche centré sur l'emplacement de la ligne dans l'image précédente (I) (figure 4.9). Une fenêtre de largeur $2 \times nbcol$ va parcourir cette zone, ce qui nous permettra de calculer, sur chaque emplacement de la fenêtre, un vecteur g et la couleur moyenne C qui lui est associée. La colonne gardée est celle qui minimisera l'erreur e tel que,

$$e = \alpha \left(\mid C_I - C_{I+1} \mid \right) + (1 - \alpha) \left(\sum_{j=0}^{h} \mid \mathbf{g}_I^j \mid - \mid \mathbf{g}_{I+1}^j \mid \right). \tag{4.7}$$

Où h est la hauteur de la bande et α une constante permettant de pondérer l'erreur de la couleur et celle du gradient. La valeur absolue sur les deux vecteurs $\mathbf{g}$ permet de prendre en compte l'inversion du gradient. Une fois la colonne trouvée, son vecteur de gradient $\mathbf{g}$ et sa couleur C sont mémorisés et utilisés comme référence sur l'image suivante. Si la somme des gradients du vecteur $\mathbf{g}$ se situe sous un certain seuil en rapport avec la hauteur h, la ligne est considérée comme perdue. La figure 4.10 nous montre les montants des portes qui sont suivies

ainsi que les lignes du couloir.

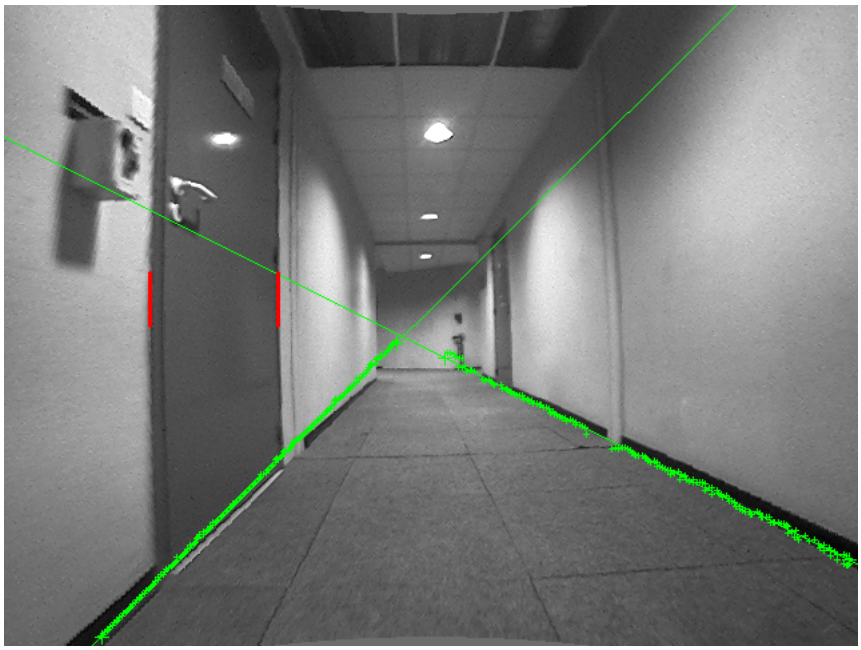


FIGURE 4.10 – Suivie des lignes de la porte, et suivie des lignes du couloir

Remarque sur le suivie des portes

L'algorithme présenté ci-dessus, se montre plus efficace que celui basée sur ME pour le suivie de ces lignes particulière. Cependant, la prise en compte de l'inversion du gradient entraîne parfois des sauts sur une autre ligne. De plus l'utilisation de la couleur rend l'algorithme sensible au changement d'illumination, ce qui est parfois problématique.

4.2.3 Asservissement sur un montant

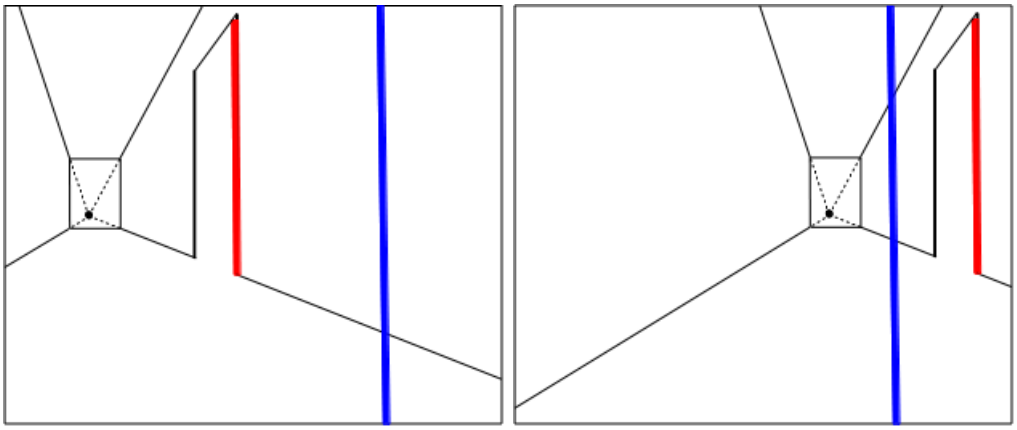
L'idée pour se positionner devant une porte consiste à asservir sur une ligne fixe dans le plan image située à environ 1/4 de l'image du côté correspondant au mur de la porte, comme le montre la figure 4.11. Avec cette stratégie le fauteuil effectuera automatiquement un détour s'il est trop proche du mur, comme l'illustre la figure 4.7. Plus la ligne désirée sera éloigné de l'axe de déplacement du fauteuil, plus le détour sera important.

Nous nous asservirons donc uniquement sur le montant le plus proche du fauteuil. L'erreur sera simplement la différence entre l'abscisse du montant de la porte, et celui de la position désirée.

Afin de pouvoir calculer la matrice d'interaction, nous devons définir le plan dans lequel se trouve la ligne. Dans un premier temps nous calculons la distance de la porte suivant,

$$d_l = \left(\frac{h \times f_v}{\mid y_p - y_{vp} \mid} \right). \tag{4.8}$$

Où h est la hauteur de la caméra, f_v la focal verticale en pixel, y_p la hauteur du point d'intersection entre le montant le plus proche de la caméra, et la ligne du couloir qui lui correspond ; et y_{vp} la hauteur du point de fuite dans l'image.



(a) (b)

FIGURE 4.11 – Dans le cas de la figure 4.11(a) le fauteuil est trop près du mur pour pouvoir la franchir, il doit donc s'en éloigner. Dans la figure 4.11(b) la distance du fauteuil au mur et suffisante, il peut commencer à se diriger vers la porte.

Cette équation exploite une propriété intéressante du point de fuite, comme toutes les lignes qui composent le point de fuite sont parallèles, y compris l'axe entre le centre optique de la caméra et le point de fuite, alors la distance verticale entre les points d'une des lignes du couloir et la hauteur du point de fuite est constante et égale à la hauteur h de la caméra sur le fauteuil.

Nous pouvons ensuite calculer la distance du plan parallèle à la caméra par lequel passe la ligne du montant qui nous intéresse selon,

$$CZ = D = \cos\left(\arctan\left(\frac{\left|\frac{w_I}{2} - x_p\right|}{f_h}\right)\right) \times d_l. \tag{4.9}$$

À ce point, nous possédons toutes les informations nécessaires pour effectuer l'asservissement visuel.

Nous n'utilisons qu'une seul mesure pour effectuer cet asservissement, en effet l'angle de la ligne de la porte restera constant, et aucun mouvement du fauteuil ne peut modifier son inclinaison. La matrice d'interaction se retrouve alors simplifié, suivant,

$$L_s = L_\rho = \begin{bmatrix} \lambda_\rho \cos(\theta) & \lambda_\rho \sin(\theta) & -\lambda_\rho \rho & (1+\rho^2)\sin(\theta) & -(1+\rho^2)\cos(\theta) & 0 \end{bmatrix}. \tag{4.10}$$

La commande généré à partir de cette matrice, suivant la formule 3.2 ne concernera que la vitesse de rotation ω du fauteuil. La vitesse vitesse d'avance du fauteuil sera ici encore fixe.

Remarques sur l'asservissement sur un montant de porte

Nous n'avons pas de résultats à présenter sur cet asservissement, dut à certains problèmes liés aux caméras fournies par advanSEE. Certaines images acquises comportaient une bande de l'image précédente créant aussi d'important artefacts qui faisaient échouer le suivie des lignes des portes. Cependant, les quelques expérimentations qui n'ont pas été affectées par ce problème semblent très prometteuses.

Un autre problème concerne l'angle de vue qui n'est pas assez grand pour effectuer cette asservissement, en effet en nous rapprochant de la porte, nous perdons la ligne du couloir, et nous sommes alors dans l'incapacité d'évaluer la distance du plan. Mais ce problème pourra être réglé avec l'utilisation de la deuxième caméra.

4.3 Conclusion et future travaux

Les différentes méthodes d'asservissement présentées dans ce rapport ont permis de confirmer que l'asservissement visuel peut être utilisée pour la navigation, mais elles sont pour l'instant au stade d'expérimentation, et beaucoup de choses peuvent encore être améliorées.

La première concerne la limitation aux couloir, pour l'instant nos algorithme sont incapables de se positionner en dehors de ce type d'environnement. Ceci est principalement dû au fait que nous n'utilisons qu'un seul point de fuite. Hors chaque mur, meuble ou tout autre objet de forme rectangulaire nous fournit un point de fuite. Utiliser les principaux pourrait nous permettre, de se positionner dans toutes sortes de pièces tant que les murs ne sont pas anordis. En plus de la détection des portes, les lignes verticales, correspondant aux angles entre les murs ou à leurs fin, pourrait nous apporter de précieuses informations à la fois pour notre positionnement, mais aussi pour étendre les possibilités de déplacement par asservissement visuel (Figure 4.12). Le déplacement du fauteuil pourrait alors s'effectuer par une commande de type « Au fond du couloir à droite, troisième porte à gauche ».

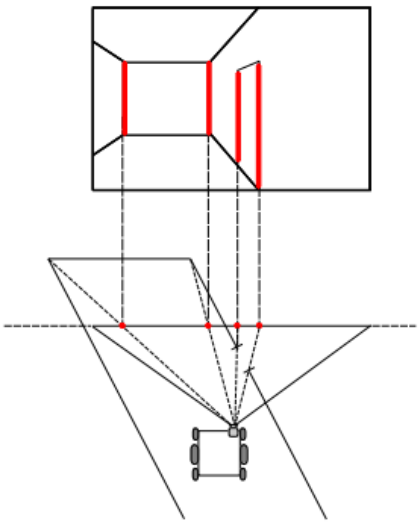


FIGURE 4.12 – Utilisation de plusieurs lignes, pour se positionner dans l'environnement

L'utilisation de la deuxième caméra devrait être intégré aux algorithmes prochainement, cela permettra de couvrir un angle de vu beaucoup plus important. Et ainsi rendre plus robuste la détection des points de fuite, des portes, mais aussi tous les algorithme d'asservissement visuel. L'utilisation de la stéréo-vision, même avec un recouvrement faible des deux caméras, pourrait éventuellement aider à l'évaluation des distances nécessaires pour l'asservissement, ainsi qu'à l'analyse de l'environnement.

La partie concernant concernant la détection des obstacles sera mis en pratique à l'aide de capteurs de proximité. Mais ces capteurs peuvent être gênant pour l'utilisateur du fauteuil, ou pourrait se trouver recouvert par les objets qu'ils transportent. Une solution basée vision pourrait être envisageable, mais la puissance de calcule embarquée sur le fauteuil est assez limitée. Mais l'ajout d'une carte dédié à la détection d'obstacle droit être envisageable. De plus la détection des obstacles pourrait exploiter les données acquises via l'utilisation des points de fuite ainsi que des lignes verticales. De cette manière il serait plus facile de dresser une grille d'occupation de l'environnement.

La prise en compte du facteur humain dans la détection des obstacle pourrait aussi être envisagé. Les filtres sociaux sont d'un intérêt évidant pour l'utilisateur. Mais la détection des humains uniquement par la vision reste une difficulté difficilement surmontable pour le moment, même sans aborder la prédiction de leurs mouvement.

L'intégration de l'utilisateur du fauteuil dans la boucle de contrôle est l'un des objectifs principaux du projet qui ne sera pas sans difficulté. Bien que cela touche à un tout autre domaine. La principale difficulté se situe au niveau de l'interprétation des mouvements du *joysick* qui devront être utilisés pour prédire la direction que l'utilisateur va emprunter. Des méthodes d'apprentissages devront être utilisée, ce qui viendra encore empiéter sur notre si précieuse et limitée puissance de calcule.

La partie la plus difficile sera finalement d'arriver à embarquer tous ces algorithmes sur le fauteuil roulant. L'augmentation importante de la puissance des processeurs ARM due au développement des *smartphones* sera certainement favorable au projet APASH, et à la robotique en générale, en permettant d'embarquer des algorithmes de plus en plus complexe.

Conclusion générale

Ce stage m'a permis de découvrir plus en détails le fonctionnement de l'asservissement visuel et du suivie de lignes. Cela m'a introduit aux problèmes de la robotique, notamment ceux liés à la navigation autonome. L'utilisation de caméras comme capteur est une chose absolument fantastique, et les utiliser pour analyser l'environnement nous amène à mieux comprendre la vision humaine, notamment la manière dont nous percevons les distances et les objets qui nous entourent. L'utilisation des points de fuite, et de l'asservissement visuel est encore peu exploité dans ce domaine, bien que se soit une technique des plus prometteuse pour les systèmes basés vision uniquement.

Malgré quelques difficultés lié à la réutilisation du code existant, ou encore aux caméras capricieuses, l'objectif du franchissement de porte semble être atteint. Certaines améliorations doivent encore être apportées sur le projet APASH, mais les résultats seront certainement à la hauteur des attentes des industriels. À terme le projet devrait être utilisé dans les hôpitaux ou les maisons de retraite, se sera un grand pas en avant pour la robotique dans le domaine de l'aide à la personne.

Bibliographie

[1] D. Anguelov, D. Koller, E. Parker, and S. Thrun. Detecting and modeling doors with mobile robots. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3777 – 3784 Vol.4, 26-may 1, 2004.

[2] M. Bennewitz, W. Burgard, and S. Thrun. Adapting navigation strategies using motion patterns of people. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[3] Kadi Boulanger, Kadi Bouatouch, and Sumant Pattanaik. ATIP : A Tool for 3D Navigation inside a Single Image with Automatic Camera Calibration. In EUROGRAPHICS, editor, *EG UK conference*, Middlesbrough, Royaume-Uni, June 2006. EUROGRAPHICS, WILEY.

[4] P. Bouthemy. A maximum likelihood framework for determining moving edges. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(5) :499–511, 1989.

[5] Baptiste Brun. Rapport de stage de master 1 informatique : Détection de portes, *Université de Rennes 1*, 2012.

[6] Zhichao Chen and S.T. Birchfield. Visual detection of lintel-occluded doors from a single image. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –8, june 2008.

[7] Zhichao Chen, Yinxiao Li, and Stanley T. Birchfield. Visual detection of lintel-occluded doors by integrating multiple cues using a data-driven markov chain monte carlo process. *Robotics and Autonomous Systems*, 59(11) :966 – 976, 2011.

[8] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam : Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6) :1052–1067, June 2007.

[9] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots ? *Advanced Robotics -Utrecht-*, 18(10) :1001–1024, 2004.

[10] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Probabilistic motion planning among moving obstacles following typical motion patterns. In *IEEE/RSJ International Conference on Intelligent RObots and Systems*, St. Louis, Missouri, United States, 2009.

[11] J. Hensler, M. Blaich, and O. Bittel. Real-Time Door Detection Based on AdaBoost Learning Algorithm. In A. Gottscheber, D. Obdržálek, and C. Schmidt, editors, *Research and Education in Robotics - EUROBOT 2009, Communications in Computer and Information Science, Volume 82. ISBN 978-3-642-16369-2. Springer Berlin Heidelberg, 2010, p. 61*, page 61, 2010.

[12] Steven M. Lavalle. Rapidly-exploring random trees : A new tool for path planning. Technical report, 1998.

[13] E. Marchand, F. Spindler, and François Chaumette. ViSP for visual servoing : a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation*

Magazine, 12(4) :40–52, 2005. Special issue on Software Packages for Vision-Based Control of Motion, P. Oh, D. Burschka (Eds.).

[14] A. C. Murillo, J. Košecká, J. J. Guerrero, and C. Sagüés. Visual door detection integrating appearance and shape cues. *Robot. Auton. Syst.*, 56(6) :512–521, June 2008.

[15] Jose Neira, Mar A Isabel Ribeiro, Juan Domingo Tardos, and Centro Politecnico Superior (cps. Mobile robot localization and map building using monocular vision. In *In The 5th Symposium for Intelligent Robotics Systems*, pages 275–284, 1997.

[16] F. Pasteau, M. Babel, and R. Sekkal. Corridor following wheelchair by visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'2013*, Tokyo, Japan, November 2013.

[17] S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2210 – 2215, aug. 2005.

[18] Viorica Pătrăucean. *Detection and identification of Elliptical Structure Arrangements in Images : Theory and Algorithms*. PhD thesis, Université de Toulouse, 2012.

[19] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, Gregory Randall. LSD : a Line Segment Detector. *Image Processing On Line*, 2012.

[20] André M. Santana, Kelson R.T. Aires, Rodrigo M.S. Veras, and Adelardo A.D. Medeiros. An approach for 2d visual occupancy grid map using monocular vision. *Electronic Notes in Theoretical Computer Science*, 281(0) :175 – 191, 2011. Proceedings of the 2011 Latin American Conference in Informatics (CLEI).

[21] Rafik Sekkal, François Pasteau, Marie Babel, Baptiste Brun, and Ivan Leplumey. Simple Monocular door detection and tracking. In *IEEE Int. Conf. on Image Processing, ICIP'14*, Melbourne, Australie, September 2013.

[22] S. Thrun. Robotic mapping : A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.

[23] Dizan Vasquez, Procópio Stein, Jorge Rios-Martinez, Arturo Escobedo, Anne Spalanzani, and Christian Laugier. Human Aware Navigation for Assistive Robotics. In *ISER - 13th International Symposium on Experimental Robotics - 2012*, Québec, Canada, June 2012. The original publication is available at www.springerlink.com.

[24] Raquel Frizera Vassallo, Hans Jörg Schneebeli, and José Santos-Victor. Visual navigation : Combining visual servoing and appearance based methods, 1998.