Aide à la navigation pour les personnes handicapées : reconnaissance de trajets

Régis Grasse

UNIVERSITÉ PAUL-VERLAINE DE METZ

IEAM Lorraine

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE METZ

Discipline : Automatique

par

Régis GRASSE

Aide à la navigation pour les personnes handicapées : reconnaissance de trajets.

Soutenue le 05 - 10 - 2007 devant le Jury :

M. Mohamed SLIMANE (professeur des universités) ..... Président
M. Patrick RIVES (directeur de recherche INRIA) ..... Rapporteur
M. Etienne COLLE (professeur des universités) ..... Rapporteur
M. Alain PRUSKI (professeur des universités) ..... Directeur de thèse
M. Yann MORERE (maître de conférences) ..... Co-Directeur de thèse
M. Guy BOURHIS (professeur des univesités) ..... Examinateur

Laboratoire d'Automatique des Systèmes Coopératifs – EA 3467
Université de Metz
Île du Saulcy
BP 80794
57012 METZ CEDEX 1 – FRANCE

A ma famille.

Remerciements

Je tiens à remercier Etienne COLLE et Patrick RIVES pour avoir accepter d'être mes rapporteurs. Je tiens également à remercier Mohamed SLIMANE pour avoir accepter de présider le jury de ma soutenance et pour ses remarques sur les modèles de Markov.

Je tiens aussi à remercier Alain PRUSKI pour m'avoir accueilli au LASC pendant les 4 années de ma thèse ainsi pour ses conseils éclairés lors des choix que j'ai eu à faire pendant tout ce temps.

Je voudrais également remercier Yann MORERE pour m'avoir soutenu et aidé à la fois dans mes travaux de recherche et pour les rédactions des diverses publications. Je le remercie également pour avoir toujours été présent et disponible pour répondre à mes questions sur la thèse, la rédaction des papiers ou encore sur le monde linuxien.

Je remercie également Olivier HABERT pour m'avoir permis de réaliser quelques heures de vacations qui m'ont permis d'avoir une rentrée pécuniaire mais également pour sa gentillesse et sa disponibilité tout au long de ces 4 années.

Je voudrais également remercier les autres permanents du LASC : Guy BOURHIS pour avoir accepté de faire parti du jury mais également pour sa convivialité et ses conseils avisés, Odile HORN pour ses conseils et sa bonne humeur, Abdellah NADIF et Choubeila MAAOUI pour leur amabilité et Pierre PINO pour sa bonne humeur et pour m'avoir permis de mettre un pied dans l'IUT GMP avec quelques encadrements de projets.

Je remercie aussi tous les thésard du que j'ai croisé dans le laboratoire pendant la durée de ma thèse. En premier Michel KRUTNER qui m'a donnée toutes les ficelles pour commencer correctement ma thèse pendant ma première année, Mhamed SAHNOUN pour sa bonne humeur permanente, Rachid MHAOUACHE, Dir SABER, Souir GHERDIRA, FAIZA, HAMEL pour tous les échanges intéressants que l'on a eut.

Je remercie également mes parents m'avoir toujours soutenus pendant toutes mes années d'études et pour m'avoir toujours encouragé à aller le plus loin possible. Pour terminer je remercie également Claudette KLESS

pour avoir bien voulu relire et rechercher les très nombreuses fautes qui se trouvaient dans ce rapport.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

LA mobilité est un besoin commun à toutes personnes. Certaines ont un handicap qui ne leur permet pas de se déplacer avec leurs bras sur des fauteuils manuels ou encore en utilisant un joystick pour contrôler un fauteuil électrique. Des interfaces spécialisées sont alors réalisées utilisant la parole, le souffle ou la contraction d'un muscle pour permettre à l'utilisateur d'émettre une commande perceptible par une machine. Cependant ces interfaces ne permettent pas de réaliser une commande précise de direction pour commander directement un fauteuil électrique. Une interface intelligente est alors nécessaire pour permettre, à partir d'ordres simples, de commander un fauteuil électrique. Mais la génération de ces commandes est fatigante pour ces personnes. Il est donc nécessaire de leur permettre de commander le fauteuil à partir d'un nombre réduit de commandes. La réduction du nombre de commandes nécessite une certaine intelligence de la part du fauteuil roulant afin de réaliser les tâches simples comme l'évitement d'obstacles mais également des tâches plus complexes comme l'aide à la navigation [93, 77].

L'utilisation des outils de la robotique mobile autonome permet de répondre à ces besoins d'autonomie. En effet, elle étudie, avec succès, depuis plusieurs décennies la possibilité de rendre un robot capable de se déplacer avec la plus grande autonomie possible. Et bien que les robots restent le plus souvent sous forme de prototypes certaines applications commerciales comme des tondeuses ou les aspirateurs autonomes sont aujourd'hui présents dans le commerce.

L'objectif du projet VAHM (Véhicule Autonome pour Handicapés Moteurs), conduit par le LASC (Laboratoire d'Automatique et de Systèmes Coopératifs) depuis le début des années 90, est de réaliser un fauteuil roulant intelligent. L'idée principale de ce projet est de pouvoir apporter une aide à la navigation aux personnes handicapées sans pour autant que celles-ci se sentent transportées par le fauteuil. L'utilisateur doit toujours être acteur de ses mouvements et doit pour cela toujours contrôler les déplacements du fauteuil. Le VAHM doit pouvoir apporter plusieurs degrés d'assistance en fonction du

handicap de l'utilisateur. Cela va du joystick à retour d'effort apportant une aide en repoussant le joystick avec une force contrôlée et proportionnelle émise dans la direction opposée aux obstacles jusqu'à un fonctionnement quasi autonome où l'utilisateur peut contrôler le mouvement du fauteuil à l'aide d'une entrée en tout ou rien (TOR) mais en laissant toujours à l'utilisateur la tâche de planification des trajets.

Dans le degré d'assistance le plus élevé du VAHM, l'utilisateur entre des commandes de direction à l'aide d'un capteur TOR. Cependant le fait d'actionner le capteur peut s'avérer difficile dans certains cas. Pour limiter le nombre d'actions, il est nécessaire que le fauteuil reconnaisse le trajet en cours pour proposer des changements de direction en fonction des habitudes de l'utilisateur. Le travail présenté dans ce rapport porte sur la reconnaissance de trajets. L'idée originale est basée sur l'utilisation de la structure multi-agent de système de contrôle du fauteuil dans laquelle les réactions du fauteuil sont guidées par des comportements (évitement d'obstacles, suivi de mur, suivi de direction...). L'enchaînement de ces comportements peut être vu comme une séquence aléatoire. Cette séquence peut alors être utilisée par des Modèles de Markov Cachés (MMC) pour modéliser les trajets effectués par le fauteuil. Cependant les résultats nous ont conduits à intégrer, dans les MMC, des données provenant des capteurs. Mais la difficulté de généralisation et la lenteur des temps d'apprentissage nous ont poussé à utiliser d'autres méthodes.

L'étude s'est alors portée sur la reconnaissance de courbes temporelles, les courbes étant l'évolution temporelle[1] des données provenant des capteurs. Le domaine dans lequel ce type d'entrées est utilisé couramment est celui du traitement d'images. Le système retenu est celui de l'algorithme CONDENSATION (CONditionnal DENSity propagATION), aussi appelé filtre particulaire ou algorithme de MONTE CARLO, pour sa capacité à déformer le modèle pour mieux se rapprocher des observations et également à intégrer naturellement plusieurs sources d'observations. L'utilisation de cet algorithme a également simplifié le problème puisqu'il est possible de travailler en prenant un trajet comme modèle. Un seul passage dans un environnement est nécessaire pour l'apprendre.

Pour lier les modèles, ils sont mis dans un réseau composé de noeuds et de transitions où les noeuds représentent les modèles de trajet et les transitions les liens qui existent entre ces trajets. Chaque transition possède

[1] Le terme "temporelle" ne signifiant pas ici la durée horaire du parcours mais plutôt de la longueur du parcours car l'échantillonnage est réalisé sur la distance parcourue. C'est le seul *tempo* disponible.

également une probabilité qui est utilisée pour proposer à l'utilisateur la direction la plus probable et de suivre cette transition lors de l'acquiescement par l'utilisateur[2].

La première partie de ce rapport présente la problématique qui entoure le projet VAHM en décrivant les besoins des utilisateurs en fonction de leurs types d'handicap. La suite introduit les technologies et méthodes utilisées pour l'acquisition de données sur le fauteuil et la commande de ce dernier. Nous nous attacherons ensuite à présenter différents projets de fauteuils intelligents issus de laboratoires de recherche. Enfin nous présenterons les différentes méthodes de localisation utilisées en robotique mobile.

La seconde partie présente le projet VAHM en le détaillant dans son ensemble. Nous y décrivons l'architecture matérielle et logicielle des fauteuils intelligents sur lesquels cette étude est appliquée. Nous présentons également le simulateur développé pour remplacer l'environnement réel tout en utilisant les agents du VAHM. Puis nous détaillerons le contexte de modélisation de l'environnement et de la localisation utilisé dans la suite de ce mémoire.

Afin de réaliser une reconnaissance de trajet, il faut pouvoir comparer les données provenant du fauteuil avec des modèles. La suite de ce mémoire présente les méthodes utilisées pour localiser un robot dans un environnement intérieur. La plupart des méthodes existantes utilisent des modèles contenant des données géographiques. Cependant notre approche initiale est de ne pas construire de carte de l'environnement et de n'utiliser que les enchaînements des comportements pour reconnaître le trajet en cours. C'est pourquoi la troisième partie contient une explication détaillée des méthodes de reconnaissance de séquences de valeurs que sont les Chaînes de Markov et les Modèles de Markov Cachés (MMC). Cette partie contient également les résultats des essais de modélisation et de reconnaissance de ces algorithmes. Cependant cette partie nous montre que les MMC peuvent difficilement réussir à classifier correctement les trajets en utilisant uniquement les séquences de comportements. L'intégration de données provenant des capteurs et l'utilisation des MMC-MD [17] sont alors introduits mais bien que plus efficaces, ils s'avèrent peu adaptés pour répondre à notre problème.

La quatrième partie utilise uniquement les courbes formées par l'évolution

[2]En fait c'est l'absence de réaction de l'utilisateur qui est considérée comme un acquiescement.

des données provenant des capteurs. Nous introduisons alors l'algorithme CONDENSATION. Il est utilisé pour vérifier la possibilité de reconnaître le trajet en cours à partir d'un trajet précédemment réalisé dans le même environnement. Dans la suite de cette partie nous présentons les résultats de cette méthode qui a été appliquée avec succès.

Ceci nous amène donc à la cinquième section de ce mémoire qui est l'utilisation de cette méthode dans le système VAHM. Nous y détaillons l'agent « suivi de trajet » qui utilise l'algorithme CONDENSATION. Puis nous présentons les résultats de la reconnaissance et de l'utilisation du réseau de modèles obtenus à l'aide du simulateur présenté dans la première partie de ce mémoire. Pour finir nous présentons les résultats de tests réels réalisés dans le bâtiment du laboratoire.

2 Robotique mobile et handicap

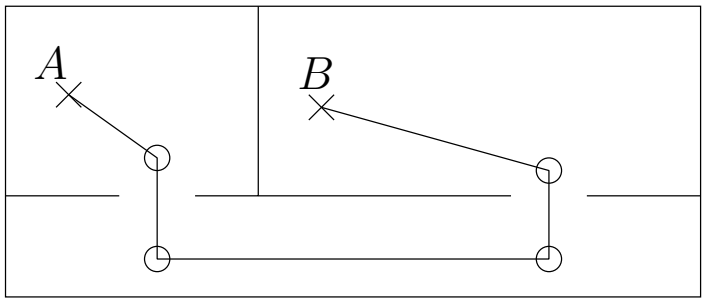
2.1 Introduction

L'évolution rapide de la robotique mobile ces dernières décennies a conduit de nombreux laboratoires de recherche à tenter d'adapter les techniques provenant de la robotique mobile à des fauteuils roulants électriques. L'objectif était et demeure toujours d'aider un peu plus les personnes handicapées au quotidien en leur rendant une autonomie de mouvement. Cependant, pour arriver à rendre autonome ces personnes, il est nécessaire qu'elles puissent donner des informations de directions compréhensibles par le fauteuil. Ces informations passent par des capteurs et plus les informations données par l'utilisateur sont pauvres en informations (utilisation d'un bouton TOR, commande vocale ...), plus le fauteuil doit pouvoir être autonome. Les différentes études sur ce sujet ont conduit à ce que l'on appelle les fauteuils intelligents (*Smart Wheelchair*). Ces fauteuils sont équipés d'une interface utilisateur répondant aux problèmes du handicap ainsi que, à l'instar des robots mobiles autonomes, de capteurs pour percevoir leur environnement et d'un calculateur.

Dans une première partie, nous nous intéresserons aux besoins liés aux différents types d'handicap. Puis, dans une seconde partie, nous étudierons les différents capteurs présents sur les fauteuils intelligents ainsi que les modes de fonctionnements les plus courants sur les fauteuils intelligents et finalement nous présenterons quelques projets de fauteuils intelligents. Dans une troisième partie nous présentons les différents modes de localisation utilisés en robotique mobile et donc aussi dans les fauteuils intelligents.

2.2 Problématique du handicap moteur : les besoins

La mobilité englobe en fait plusieurs notions que sont les trajets et la planification de trajet. Ainsi, comme le montre la figure 2.1, pour se déplacer d'un point A à un point B on effectue un enchaînement de trajets



× *Point de départ/arrivée*
○ *Changments de direction*

FIG. 2.1: Exemple d'un chemin constitué de trajets et de changements de direction. Le fait d'aller d'un point A à un point B demande de planifier ces changements de direction.

et à chaque nouveau trajet on définit une nouvelle direction. C'est justement ces changements de direction qui posent problèmes aux personnes handicapées.

Mais le besoin de mobilité est un besoin commun à toute personne même à celles qui souffrent d'un handicap ne leur permettant pas de conserver cette mobilité. Elles sont donc contraintes à être assistées et ne sont pas maîtresses de leurs déplacements. Le but des fauteuils intelligents est de leur rendre tout ou partie de cette mobilité perdue, de leur indépendance et de la maîtrise de leurs mouvements.

Les problèmes du handicap moteur sont variés et dépendent de la pathologie. Il n'est, bien sûr, pas question ici de rentrer dans les détails des différentes maladies ou accidents qui ont causé ces handicaps mais uniquement de leurs symptômes physiques : ceux qu'un fauteuil intelligent peut aider à contrer. Ces symptômes sont variés tant dans les parties du corps qu'ils affectent que dans le degré d'handicap qu'ils entraînent. Ils peuvent être placés dans deux catégories :

- une difficulté à réaliser des mouvements précis
- une incapacité à commander une partie des muscles.

De plus certaines lésions cérébrales impliquent, en plus des problèmes moteurs, des problèmes cognitifs altérant la capacité de la personne à planifier les trajets.

2.2.1 Imprécision de la commande musculaire

Dans cette catégorie on peut citer des causes telle que la maladie de Parkinson qui entraine un mouvement non désiré et qui bruite le mouvement initial. On y trouve aussi des lésions cérébrales qui entraînent une difficulté à commander efficacement les muscles (manque de force, de précision ou d'endurance dans le maintien d'une commande continu) ou encore des maladies empêchant progressivement la commande correcte des muscles.

Le besoin principal de ces types d'handicap se situe au niveau de l'interprétation de la commande et de l'aide à la définition de trajectoires.

2.2.2 Défaillance de la commande musculaire

Cette catégorie concerne principalement les cas où il est impossible de commander certains muscles. Les besoins diffèrent en fonction des muscles atteints. En effet pour les personnes hémiplégiques aucune aide intelligente n'est nécessaire. Lorsque les invalidités concernent les parties inférieures, seule une adaptation des systèmes de commandes est nécessaire. Cependant lorsque des difficultés dans la réalisation d'une commande correcte surviennent, une aide à la commande devient alors nécessaire tant dans les outils permettant de capter des signaux de commandes que dans les aides apportées pour la définition des trajets.

Les besoins dans ces cas d'handicap sévères sont liés à la définition d'une commande de direction. En effet dans ces cas il n'est pas possible d'obtenir une commande riche en information car les personnes ne peuvent plus commander leurs membres avec précision. Les informations de commande de direction proviennent alors principalement d'un capteur TOR (Tout ou Rien) ou encore de commandes vocales. Les capteurs TOR utilisés sont des micro-rupteurs (figure 2.2(a)), qui peuvent être très sensibles, actionnés par une partie mobile du corps (doigt, tête ...), des capteurs sensibles aux mouvements des muscles (figure 2.2(b)), des capteurs de souffle (figure 2.2(c)) ou encore des capteurs de clignement de paupière (figure 2.2(d)).

2.2.3 Problèmes cognitifs

Les problèmes cognitifs altèrent les capacités de la personne à planifier un trajet ou à se repérer dans l'environnement dans lequel elle évolue. Il leur est donc impossible de définir quels trajets ils doivent prendre pour aller d'un point A à un point B et dans certains cas ils ne savent pas où

(a) Exemple de capteur de grand taille ne demandant pas une grande précision pour être actionné.

(b) Exemple de capteur musculaire. Ce capteur réagit aux contractions, même faibles, d'un muscle.

(c) Exemple de capteur de souffle.

(d) Exemple de capteur de paupière monté sur lunettes. Ce capteur réagit au passage de la paupière dans le faisceau IR normalement réfléchi par l'oeil.

FIG. 2.2: Différents types de capteurs adaptés au handicap.

placer le point B sur une carte.

Les besoins sont donc l'aide à la planification de trajet et l'aide à la localisation.

Il arrive que des personnes cumulent plusieurs types d'handicap ce qui entraîne une augmentation des besoins d'assistances et donc, pour les concepteurs des systèmes d'aide, un besoin important de flexibilité dans les outils d'assistances.

Comme nous venons de le voir les besoins se trouvent à la fois dans la réalisation de commandes et dans l'aide à la navigation. Cependant de nombreux systèmes permettent aux personnes handicapées de générer des informations. Ces systèmes sont grandement utilisés dans l'aide à la communication et peuvent également servir de source pour l'aide à la navigation.

La partie suivante présente différents systèmes d'aide à la mobilité qui sont constitués essentiellement de fauteuils intelligents.

2.3 Les fauteuils intelligents (*Smart Wheelchair)*

Les robots mobiles ont grandement évolué ces trois dernières décennies, ils ont été constament améliorés et ont pleinement profités de l'évolution de la microélectronique pour devenir plus "intelligents", plus réactifs et plus autonomes. Cette évolution a naturellement été transposée dans le domaine du handicap et c'est vers la fin des années 80 que sont apparus les premiers

fauteuils intelligents dans les laboratoires de recherche. Les premiers fauteuils intelligents étaient composés d'une base robotique sur laquelle était monté un fauteuil (par exemple Mister Ed [24], le premier prototype du VAHM [14]) mais les fauteuils intelligents plus récents sont généralement des fauteuils électriques du commerce, sur lesquels ont été ajoutés des capteurs et un calculateur (par exemple NavChair [53], OMNI [12], MAid[73], SENARIO[8], VAHM [13]). D'autres projets (par exemple SWCS[92], SPAM [89], Hephaestus[90], TinMan [61], Siamo[58]) ont été développés comme des ajouts qui peuvent être mis et enlevés du fauteuil électrique standard. Le but de ces assemblages est de pallier aux déficiences des utilisateurs en leur apportant une aide à la définition d'une commande de direction et une aide à la navigation ajustées à leur handicap.

Cependant, la réalisation d'un fauteuil intelligent est une tâche longue et fastidieuse. C'est la raison pour laquelle la plupart des projets sont encore en cours de développement et qu'il n'existe que peu de fauteuils intelligents commercialisés. Seul le CALL[1] Center propose, par le biais le la société Smile Rehab (Berkshire, Grande Bretagne), des systèmes permettant à des enfants d'apprendre à utiliser un fauteuil électrique. Les difficultés dans la réalisation d'un fauteuil intelligent se trouvent à tous les stades de leur élaboration [93].

Une première difficulté vient de la fabrication du fauteuil. Bien que des sociétés comme ActivMedia Robotics[2] et Applied AI Sytems, Inc.[3] commercialisent des fauteuils du commerce déjà équipés de capteurs, la plupart des fauteuils intelligents sont réalisés par les équipes des laboratoires à partir de fauteuils électriques standard.

La sécurité est une deuxième source de problèmes qui doit être incluse dans le projet. En effet le but étant de fournir ces fauteuils à des personnes handicapées, il est nécessaire d'en sécuriser l'utilisation. Le fauteuil doit ainsi par exemple éviter de tomber dans des escaliers ou encore de se trouver dans une situation bloquante comme, par exemple, se retrouver dans un couloir étroit sans possibilité de faire demi-tour ou marche arrière.

Enfin le choix des aides apportées par le fauteuil (fonctionnement autonome ou semi-autonome) et des technologies utilisées pour rendre le fauteuil "intelligent" influent sur les difficultés de mise en oeuvre de fauteuil. Ainsi la mise en place d'un système de vision ou d'un système de locali-

[1] Communication Aids for Language and Learning (aide à la communication et à l'apprentissage)
[2] http ://www.activrobots.com/ROBOTS/RoboChariot.html
[3] http ://www.aai.ca/robots/tao_7.html

sation sont des tâches fastidieuses et les rendre sécuritaires et efficaces demande de nombreux tests.

Ces difficultés n'ont cependant pas empêché de nombreux laboratoires de part le monde à s'investir dans ce domaine comme le montre l'état de l'art de R.C. Simpson [93] qui recense 45 projets de fauteuils intelligents dans le monde. Les projets de fauteuils roulants intelligents n'ont pas tous été conçus avec les mêmes objectifs et ils n'intègrent donc pas les mêmes fonctionnalités. Globalement ces objectifs dépendent du niveau d'assistance voulu. Ces niveaux d'assistance vont d'un faible niveau d'assistance où le fauteuil réalise des évitements d'obstacles [71, 73] jusqu'à une assistance totale où l'utilisateur n'a qu'à indiquer sur un plan le lieu où il veut aller pour que le fauteuil s'y rende [97, 6].

Parmi les fauteuils semi-autonomes, la plupart utilisent un système à bases de plusieurs comportements. Ces comportements sont expliqués dans la suite de cette partie mais avant de présenter plus en avant les modes de fonctionnement et les comportements, il est intéressant de présenter les capteurs utilisés sur les fauteuils intelligents. L'intérêt de cette présentation vient du fait que l'utilisation de certains comportements est liée aux limitations des capteurs utilisés.

Une fois les capteurs présentés, nous nous intéresserons aux modes de fonctionnement et aux comportements utilisés dans les différents projets de fauteuils intelligents. Puis nous enchaînerons sur une présentation de quelques projets existants.

2.3.1 Présentation des capteurs

Les fauteuils intelligents ont besoin de percevoir leur environnement afin d'aider l'utilisateur. Cette aide est utile car, dans certains cas, la pathologie de l'utilisateur ne lui permet pas de regarder dans toutes les directions. C'est alors au fauteuil de percevoir l'environnement qui l'entoure. Cette perception doit être précise afin de mieux connaître l'environnement. Cependant les intérêts économiques pour une probable commercialisation doivent être pris en compte. C'est pourquoi le capteur le plus présent est, comme en robotique mobile classique, le capteur à onde Ultra-Sonore (US). Les capteurs Infra-Rouge (IR) sont également très présents tandis que les scanners laser sont moins présents car, bien que très précis, ils restent onéreux et souvent encombrants. L'utilisation d'une caméra est également courante mais c'est la complexité de la mise en oeuvre qui freine ici son

utilisation.

Les Capteurs peuvent être classés dans deux catégories, proprioceptifs ou extéroceptifs, selon qu'ils mesurent respectivement des données internes ou externes au robot. Afin de mieux comprendre les choix des capteurs présents sur les différents projets présentés plus avant mais également de mieux appréhender la problématique de la localisation, nous allons présenter les différents capteurs présents en robotique mobile d'intérieur.

2.3.1.1 Les capteurs proprioceptifs

Les capteurs proprioceptifs renseignent le robot sur ses mouvements. Ils mesurent, soit le déplacement de ses roues à l'aide de codeurs optiques, soit les données physiques d'accélération à l'aide d'accéléromètre et de gyroscopes ou encore son orientation par rapport au champ magnétique terrestre avec une boussole numérique.

Les codeurs optiques Il existe deux types de codeurs optiques qui sont les codeurs incrémentaux et les codeurs absolus. Les codeurs absolus donnent une information d'angle de rotation et sont très peu utilisés en robotique mobile contrairement aux codeurs incrémentaux qui donnent une information de vitesse de rotation qui sont présents sur la plupart des robots mobiles roulants. Ils servent à la fois à l'asservissement en vitesse des roues des robots et à la localisation.

Le codeur incrémental utilise un disque translucide comprenant des zones opaques placées régulièrement et un faisceau de lumière traversant le disque. Lorsque le disque tourne, le faisceau de lumière est interrompu par les zones opaques et le signal extrait de ce capteur a donc une fréquence correspondante à la vitesse de rotation. Cependant en utilisant un seul faisceau de lumière il est impossible d'obtenir le sens de rotation. C'est pourquoi la plupart des codeurs incrémentaux utilisent un deuxième faisceau monté en quadrature de phase par rapport au premier, c'est-à-dire que les deux faisceaux sont décalés afin qu'ils génèrent des signaux décalés d'un quart de période. Ce décalage permet de connaître le sens de rotation comme le montre la figure 2.3. Un troisième faisceau utilisant une seule marque sur la révolution du disque est également souvent présent afin de détecter une révolution complète du disque.

Idéalement ce capteur permet de connaître la distance parcourue par la roue sur laquelle il est monté en multipliant le nombre de tours effectué par le codeur avec le périmètre de la roue. Cette méthode s'appelle l'odométrie.

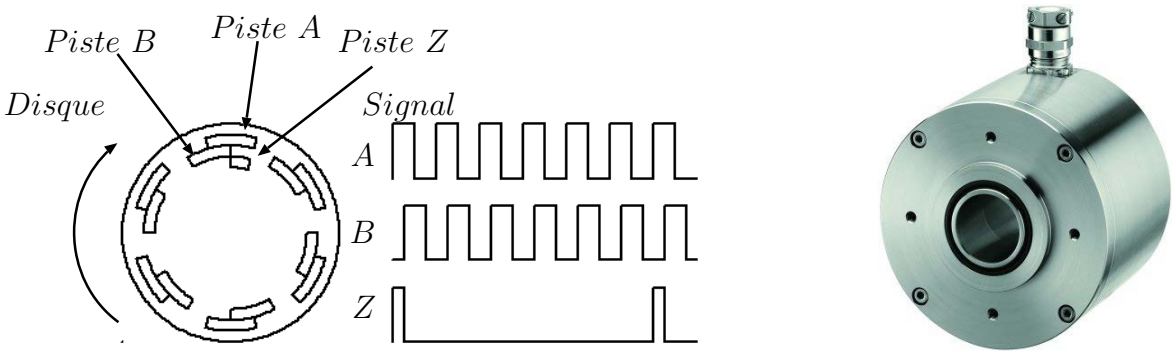


FIG. 2.3: Schéma de principe d'un codeur incrémental optique. Les voies A et B sont en quadrature de phase et la voie Z sert d'index pour définir chaque tour du codeur.

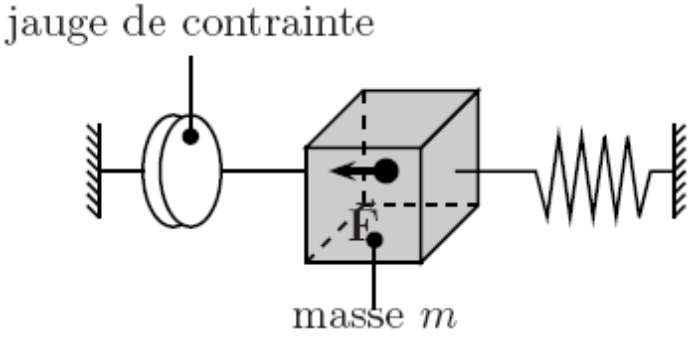


FIG. 2.4: Exemple de fonctionnement d'un accéléromètre. Lors de l'accélération, le mobile de masse m produit une force $\vec{F}$ qui est mesurée par la jauge de contrainte. L'accélération γ est donnée par la formule $\gamma = F/m$.

Cependant les erreurs de modélisation du diamètre de la roue ainsi que les glissements de la roue sur le sol entachent ces mesures d'erreurs. Dans le cas où le robot n'utilise que l'odométrie pour se positionner (*dead reconning*), cette erreur est cumulée à chaque mesure et se retrouve non bornée [65].

Les accéléromètres Les accéléromètres, dont un type de fonctionnement est présenté dans la figure 2.4 sont des capteurs qui mesurent les accélérations linéaires. Ils peuvent donc mesurer les déplacements d'un robot mais la double intégration de la mesure pour obtenir une information de position les rendent sensibles aux erreurs de mesure. De plus ils sont sensibles à l'accélération de la pesanteur ce qui les rend sensibles aux variations de l'inclinaison du robot.

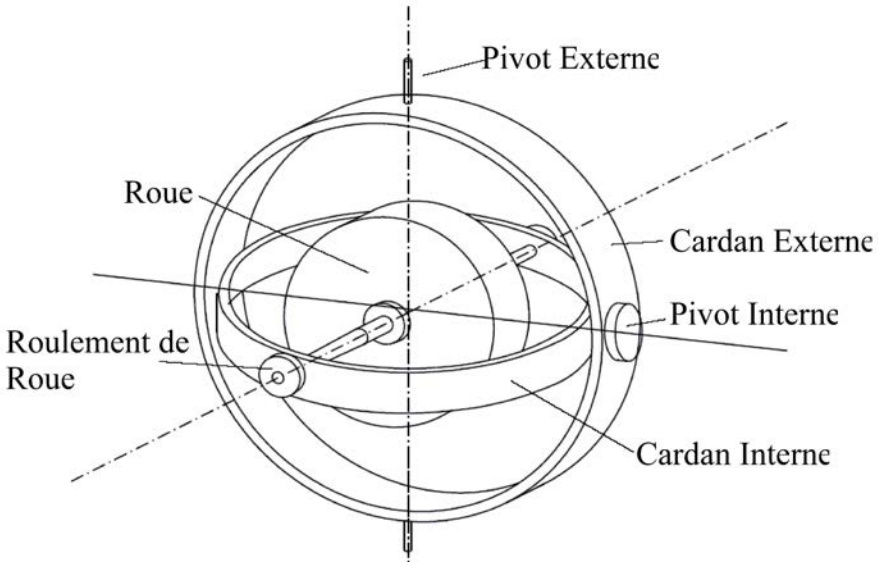


FIG. 2.5: Schéma de principe d'un gyroscope mécanique. Le gyroscope mécanique se sert du principe de conservation des forces d'un mobile en rotation (roue) qui tendent à ramener la roue dans son axe de rotation. La mesure de ces forces donne des informations sur les variations de vitesse angulaires ou linéaires.

Les gyroscopes Les gyroscopes sont des capteurs qui mesurent la vitesse angulaire. Ils permettent de mesurer les changements de direction mais également les changements d'inclinaison du fauteuil. Il existe deux types de gyroscopes, les gyroscopes mécaniques (cf figure 2.5) et les gyroscopes optiques. Les gyroscopes mécaniques sont sensibles à la rotation de la terre s'ils ne sont pas orientés dans l'axe nord-sud et les gyroscopes optiques sont trop onéreux pour être montés sur des fauteuils intelligents.

Les gyroscopes et les accéléromètres sont souvent couplés dans une unité appelée centrale inertielle qui permet de mesurer correctement les déplacements même si des dérives persistent. Ils peuvent s'avérer un complément intéressant à l'odométrie. Ainsi dans [35] un gyroscope a été utilisé pour corriger les erreurs odométriques.

2.3.1.2 Les capteurs extéroceptifs

Les capteurs extéroceptifs permettent au robot de percevoir le monde qui les entoure. Ces capteurs utilisent le contact (*bumper*), le son (capteur US) ou la lumière (IR ou laser).

(a) Capteur équipé d'un émetteur et d'un récepteur.

(b) Transducteur US et son électronique de contrôle. C'est le même composant qui émet et réceptionne l'onde US.

FIG. 2.6: Exemples de capteurs US.

Les capteurs de contact Les capteurs les plus simples sont les capteurs de contact ou « bumper ». Ils sont actionnés en cas de choc avec l'environnement et servent généralement en cas de défaillance des autres systèmes.

Les capteurs à ondes Ultra Sonore (US) Le capteur US, aussi appelé télémètre US ou sonar, est le type de capteur le plus couramment utilisé pour mesurer la distance aux obstacles [30]. Son fonctionnement est simple, il est facile à mettre en oeuvre et est peu onéreux [83] d'où son utilisation sur un grand nombre de robots mobiles [28, 62, 26, 47]

Le capteur US mesure le temps que met un train d'ondes pour aller d'un émetteur à un récepteur. Le capteur est composé soit d'un module émetteur/récepteur (figure 2.6(b)) soit de deux modules, un émetteur et un récepteur, placés côte à côte dans la direction (figure 2.6(a)) . Le signal reçu par le récepteur est l'écho du signal émis. Le temps de vol mesuré est celui de l'aller-retour du train d'onde. Connaissant la célérité $(c \approx 340m/s)$ de ce signal dans l'air, on peut alors calculer la distance parcourue par ce signal et donc la distance D à l'obstacle par la relation :

$$D = \frac{1}{2}cT$$

où T représente le temps de vol du train d'ondes. Cependant la valeur de la célérité varie légèrement en fonction de paramètres comme la température, la pression ou de l'humidité de l'air ce qui fait varier les mesures.

Le capteur le plus utilisé est l'émetteur/récepteur Polaroid [72] dont une description complète est disponible dans la littérature [32, 29]. Nous ne

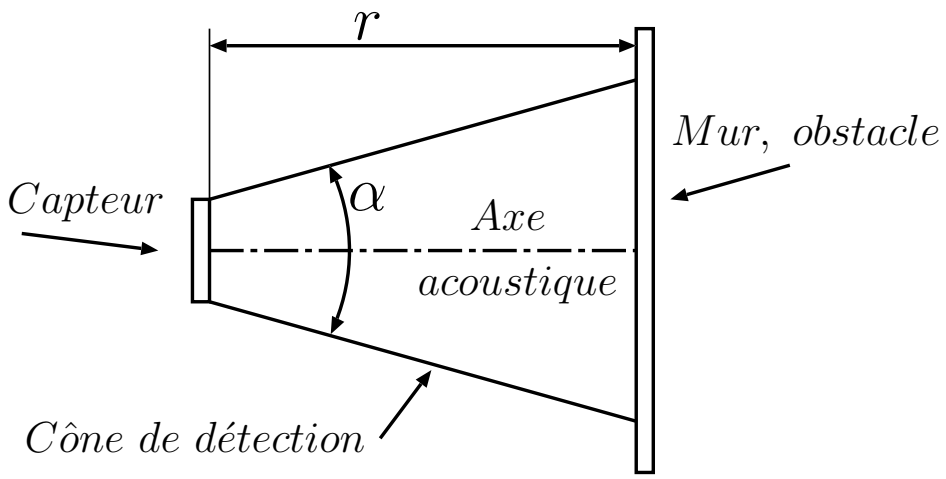


FIG. 2.7: Schéma de fonctionnement d'un capteur US.

donnerons ici que les caractéristiques générales du capteur Polaroid, celles des autres capteurs US étant semblables. La plage de mesure est de 10cm à 10m. L'onde émise par le capteur est un cône d'une ouverture d'environ 25° ce qui entraîne une perte d'intensité proportionnelle à l'inverse du carré de la distance parcourue. Cet angle d'ouverture procure au capteur une bonne vision des obstacles mais il est cependant impossible de donner la direction exacte de l'écho, l'obstacle se situe dans le cône de détection comme le montre la figure 2.7.

Les capteurs US ont trois problèmes majeurs, l'angle d'incidence entre l'onde et l'obstacle, les échos non désirés et la faiblesse de l'écho. Si l'angle d'incidence entre l'obstacle et le signal ne permet pas au signal de retourner vers le capteur, l'obstacle ne sera pas détecté.

Quant aux échos non désirés, ils proviennent de rebonds successifs du signal avant d'être captés par le récepteur ou encore d'un signal provenant d'une autre source, d'un autre capteur (on parle alors de *cross-talk*). Ceci entraîne la détection d'obstacles qui n'existent pas. Il est nécessaire d'éviter que plusieurs capteurs US émettent en même temps afin justement d'éviter le phénomène de *cross-talk* même si cela ralentit la vitesse d'acquisition.

L'atténuation de l'écho est due à la matière de l'obstacle, à l'angle d'incidence et à la surface de réflexion. Si la matière est trop absorbante, elle atténuera trop le signal et le récepteur ne pourra plus le capter.

La taille de l'obstacle et plus spécifiquement de la surface créant l'écho doit également être assez grande pour renvoyer suffisamment de signal. Si cette zone est petite comme par exemple pour un pied de table ou de chaise, la quantité de signal retournant vers le récepteur peut être trop faible pour

permettre une détection. Il en va de même si l'obstacle est arrondi car la majeure partie du signal ne retourne pas vers le capteur.

Malgré tous ces défauts le capteur US est sans doute le capteur le plus utilisé dans la robotique mobile d'intérieur car il permet de voir à plusieurs mètres pour un coût relativement faible avec une précision suffisante pour éviter les obstacles ou se localiser.

Les capteurs IR Un autre type de capteur couramment utilisé et le capteur IR (infra-rouge). Il existe deux utilisations des capteurs IR qui sont la mesure de distance et l'utilisation comme capteur TOR sans contact.

Lorsque l'on réalise une mesure de distance, en fait c'est l'angle que fait le rayon lumineux partant d'une source et arrivant à un récepteur placé sur le même axe et orienté vers la même direction comme le montre la figure 2.8 qui est mesuré. Le récepteur est constitué d'une ligne de capteurs permettant de déterminer l'angle de réflexion. Plus la distance à mesurer est grande, plus l'angle est petit.

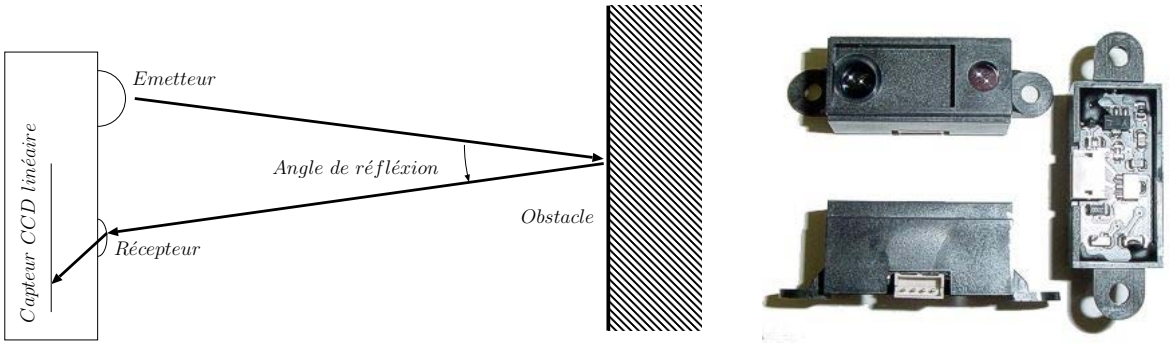


FIG. 2.8: Fonctionnement d'un capteur IR à mesure d'angle de réflexion et exemple d'un capteur IR

Il possède également quelques défauts qui sont :
- une portée limitée à environ 1m [54]
- une sensibilité à la couleur de la surface réfléchissante diminuant la précision et la distance maximale mesurable
- une grande difficulté à voir les surfaces vitrées car le rayon passant au travers
- une sensibilité à la lumière extérieure bien qu'une modulation du signal puisse éliminer ce problème.

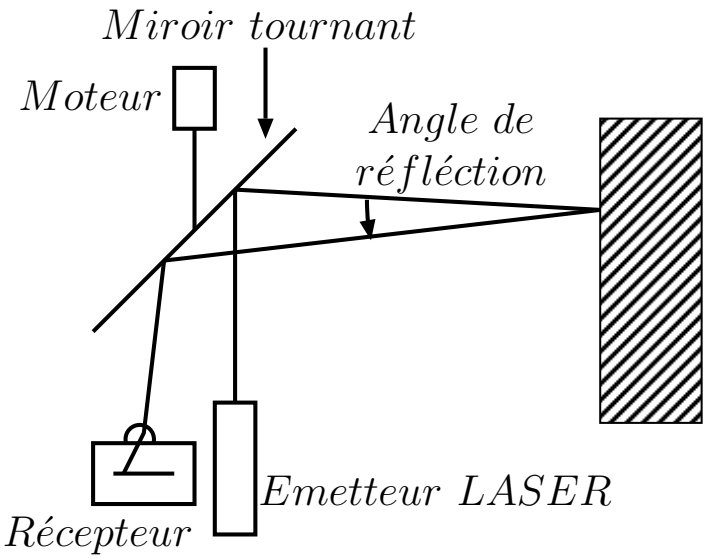
Il possède cependant une vitesse de mesure bien plus élevée que celle des capteurs US.

La deuxième utilisation des IR donne une information TOR, comme un

bumper mais sans contact. Ce capteur est constitué d'une DEL (Diode Electro-Luminescente) IR et d'un photo-transistor, tous deux orientés dans la même direction. Lorsqu'un obstacle se trouve dans le champ d'émission de la DEL et est suffisamment proche pour refléter la lumière vers le photo-transistor, celui-ci devient passant et la présence de l'obstacle est détectée. Ce système est très peu onéreux mais est sensible à la couleur de l'obstacle, à la lumière du jour et de source d'IR importante comme une ampoule à incandescence. Afin d'éviter de mauvaises détections dues à la lumière du jour, il est possible de moduler le signal.

Les capteurs laser Des capteurs lasers sont également utilisés pour mesurer les distances. Ces capteurs fonctionnent soit sur le même principe que les capteurs IR (mesure de l'angle de réflexion) soit sur une mesure indirecte de temps de vol. La faible divergence du rayon laser leur permet d'avoir une mesure ponctuelle plutôt qu'une mesure de zone. Ils sont le plus souvent utilisés dans un scanner laser qui consiste à passer par un miroir tournant incliné à 45° (cf figure 2.9(a)). Les mesures sont alors obtenues en 2D dans le plan de mesure du capteur. Ce système est onéreux et généralement relativement encombrant même si des capteurs de petites dimensions commencent à apparaître comme celui présenté dans la figure 2.9(b). L'angle d'ouverture du scanner laser est généralement de l'ordre de 180° à 240°. Ces capteurs sont plus utilisés en robotique classique [99, 19, 40] que sur les fauteuils roulants car la position de l'utilisateur gène son installation. En effet, il doit être placé soit entre les jambes de l'utilisateur, soit devant l'utilisateur [82] ou encore au dessus de l'utilisateur comme sur le Robotic Chariot de ActivMedia Robotics[4].

[4]http ://www.activrobots.com/ROBOTS/RoboChariot.html



(a) Fonctionnement d'un scanner LASER

(b) Exemple d'un scanner laser de petite dimention (50 x 50 x 70 mm)

FIG. 2.9: Fonctionnement d'un scanner LASER

Les caméras L'utilisation d'une caméra uni-directionnelle [95, 50] ou omni-directionnelle [100] comme capteur est aussi possible. Cette méthode est cependant fastidieuse à mettre en oeuvre et requiert un traitement conséquent des données mais les résultats obtenus sont souvent riches en informations. La nature des résultats peut varier en fonction des traitements appliqués à l'image et des informations recherchées. Une caméra peut aussi être utilisée pour éviter des obstacles ou visualiser la position de l'utilisateur. Plus généralement la caméra rentre dans la boucle du système de localisation en reconnaissant soit des amers[5] artificiels (dessins de couleur et de taille définis ajoutés dans l'environnement) soit des amers naturels comme des coins de murs ou de portes qui forment des lignes verticales.

Le fait qu'il n'existe pas de capteur parfait oblige les concepteurs des fauteuils intelligents à multiplier les capteurs et même parfois à utiliser différents types de capteurs pour obtenir une couverture optimale en combinant par exemple des capteurs US + IR ou US + IR + bumper afin d'obtenir une détection des obstacles lointains et proches.

[5]Amer : (en marine) Objet fixe servant de repère sur la côte. En robotique cela désigne un point fixe remarquable par le système, généralement à base de caméra, comme un coin de mur, une porte...

Maintenant que nous avons vu de quelle manière le robot peut percevoir son environnement, nous allons nous intéresser aux différents modes de fonctionnement présents sur les fauteuils intelligents.

2.3.2 Présentation des modes de fonctionnement

Il existe trois modes de fonctionnement : autonome, semi-autonome et sans assistance. Le mode sans assistance utilise simplement le joystick de contrôle pour commander les moteurs du fauteuils. Les modes autonome et semi-autonome sont détaillés dans la partie suivante.

2.3.2.1 Fonctionnement autonome

Un fonctionnement autonome signifie que l'utilisateur possède une interface dans laquelle il sélectionne une destination et le fauteuil calcule le chemin pour y arriver. Cette méthode requiert une localisation précise du fauteuil dans l'environnement mais ne demande pas une grande attention de l'utilisateur. Cependant elle demande, soit que le fauteuil ait une connaissance complète de l'environnement, soit que l'environnement ait été modifié en lui ajoutant des repères. Dans ce cas le mode de fonctionnement ne permet pas de gérer les changements de l'environnement ni de naviguer dans un environnement inconnu [97, 6].

La connaissance de l'environnement s'effectue à l'aide d'une carte complète de celui-ci qui se trouve dans la mémoire du fauteuil. Cette carte peut être de type métrique (elle contient des informations de distance) ou topologique (elle contient des informations de connexion entre les lieux connus) et retrace l'intégralité de l'environnement connu. Cependant la réalisation d'une carte complète d'un environnement n'est pas une tâche aisée. Une présentation plus complète des différents modèles d'environnement et des méthodes de localisation se trouve dans le chapitre 2.4.

2.3.2.2 Fonctionnement semi-autonome

Le fonctionnement semi-autonome des fauteuils intelligents permet aux utilisateurs d'avoir des aides à la navigation leur permettant de commander le fauteuil malgré les pathologies dont ils souffrent. Ce qui demande donc une plus grande attention des utilisateurs ainsi qu'un effort constant de planification de trajet. Cette aide se limite parfois aux seuls évitements d'obstacles mais il se compose plus généralement de plusieurs compor-

tements. Le choix entre ces comportements est soit laissé à l'utilisateur [61] (adaptation manuelle) soit réalisé de manière automatique [74, 36, 53] (adaptation automatique). Le mode de sélection automatique le plus couramment utilisé celui de subsumption [16] mais d'autres systèmes sont utilisés tels que des systèmes à base raisonnement probabiliste [91] ou à base de raisonnement à partir de cas [77].

Dans ces deux modes de fonctionnements, les fauteuils utilisent généralement un système à base de comportement pour générer les commandes de vitesse et de direction.

2.3.3 Présentation des comportements

Les comportements représentent la manière dont le fauteuil réagit aux situations qu'il rencontre. Il utilise pour cela les données provenant des capteurs.

- Le comportement le plus utilisé sur les fauteuils intelligents est l'évitement d'obstacles qui permet au fauteuil d'éviter de heurter des obstacles en modifiant sa trajectoire. Il utilise pour cela les données issues des capteurs de distance présents sur le fauteuil (US, IR, laser) et calcule une trajectoire courte passant à une distance suffisante des obstacles. Ces paramètres sont très subjectifs et dépendent des choix réalisés par les développeurs ou l'utilisateur [74]. L'évitement d'obstacles est un comportement directement issu de la robotique mobile [75] avec cependant quelques problèmes liés au fait que le fauteuil soit commandé par une personne dont il est impossible, *a priori,* de connaître les intentions. Il existe donc des problèmes lorsque la personne désire accoster à un bureau ou passer dans un endroit étroit. C'est pourquoi dans ces cas particuliers des comportements spécifiques sont créés. Il n'est donc pas rare de voir apparaître les comportements *docking* (accostage) ou *door crossing* (passage de portes) dans la littérature.
- Un autre comportement courant est le suivi de mur qui aide l'utilisateur à garder une trajectoire rectiligne dans un couloir ou le long d'un mur. En utilisant uniquement l'évitement d'obstacles le fauteuil peut se mettre à osciller dans le couloir, renvoyant constamment le fauteuil d'un mur à l'autre sans amortir le rebond. Ce défaut est inconfortable pour l'utilisateur et, en plus, il démontre un manque d'efficacité dans l'aide à la définition de trajets.

- Le passage de porte permet de passer dans un endroit étroit. En effet les portes étant relativement petites par rapport aux fauteuils du commerce qui mesurent entre 70 et 80 cm de large alors qu'une porte standard mesure 90 cm. Les capteurs US, lorsqu'ils sont configurés pour mesurer une grande distance (quelques mètres), sont dans l'impossibilité de voir une distance faible (en dessous d'une dizaine de centimètres). Les capteurs IR sont alors souvent ajoutés pour les faibles distances. Cependant cette faible marge peut empêcher l'évitement d'obstacles de permettre le passage de la porte. En effet, les montants de porte sont détectés comme des obstacles et empêchent l'algorithme de commande, basé la plupart du temps sur les champs de vecteurs, de donner la bonne consigne de direction et de vitesse. La solution est donc d'utiliser un comportement spécifique qui réalise cette tâche.
- Le retour en arrière (*backtraking*) permet au fauteuil de revenir sur ses pas. Dans les cas où le fauteuil se retrouve dans un endroit où il ne peut pas faire demi-tour il faut alors que le fauteuil puisse faire une marche arrière. Mais l'utilisateur ne pouvant pas voir ce qui se passe derrière lui, le fauteuil revient donc sur ses pas afin d'éviter de rentrer en collision avec les obstacles qu'il avait évités.
- Le comportement d'accostage permet aussi de pallier aux mêmes problèmes que le passage de porte puisqu'il permet au fauteuil d'aller contre un obstacle ce qu'interdit le comportement d'évitement d'obstacles.
- Le suivi d'objet est également présent sur certains fauteuils. C'est un comportement plus complexe que les autres puisqu'il permet de suivre un objet sélectionné par l'utilisateur comme une personne ou un autre fauteuil. Ce comportement utilise une caméra qui traque l'objet à suivre dans l'image ou des capteurs US situés à l'avant du fauteuil pour générer les commandes. Aucune intervention de l'utilisateur est alors nécessaire.
- La répétition de trajectoire qui permet au fauteuil de refaire une trajectoire qu'il a déjà effectuée.
- le suivi de ligne qui permet de suivre une ligne détectée par le fauteuil. Cette ligne peut être de couleur [20] ou magnétique [98].
- Le comportement de demi-tours permet de réaliser un demi-tour sur place où en trois temps car les fauteuils ne sont pas des robots holonomes à l'exception du projet OMNI [12].
- Le comportement de contact et sauvegarde (*bump and backup*) qui per-

FIG. 2.10: Un prototype du CALL Center, université d'Edimburgh (Angleterre)

met, en cas de contact avec l'environnement, de stopper le fauteuil et de reculer pour se dégager.

Maintenant que nous avons vu les différents éléments qui composent les fauteuils intelligents, nous allons nous intéresser à quelques projets existants.

2.3.4 Les projets de fauteuils intelligents

Dans cette partie nous allons présenter quelques projets en détaillant leurs fonctionnalités présentes sur le projet ainsi que les outils utilisés lorsque c'est possible.

Smart Wheelchair du CALL center[6]

Le projet Smart Wheelchair du CALL center (Communication Aids for Language and Learning) de l'université d'Edinbourgh [66, 20] est un des projets très abouti puisqu'il est utilisé dans plusieurs centres de la région d'Edinbourgh. Le but de ce fauteuil est de rendre la mobilité à des enfants lourdement handicapés. Plusieurs prototypes ont été développés pour répondre à des demandes spécifiques comme par exemple celui de la figure 2.10 qui utilise un système de visée oculaire comme entrée de commande. Le fauteuil est fabriqué sur une base de fauteuil électrique standard dont le joystick de

[6]http ://callcentre.education.ed.ac.uk/Smart_WheelCh/smart_wheelch.html

FIG. 2.11: Le prototype ROLLAND de l'université de Brême (Allemagne)

contrôle a été remplacé par un "contrôleur intelligent" composé, au besoin, d'un ou plusieurs boutons poussoirs, de systèmes optiques ou encore de reconnaissance vocale. Les autres fonctionnalités présentes sont l'utilisation de "bumpers" pour que le fauteuil puisse réagir en cas de collision et un mode "suivi de ligne" qui permet de suivre des lignes de bande réfléchissantes au sol. Le but de ce fauteuil est de faire apprendre la conduite d'un fauteuil standard. Il permet donc de passer du mode autonome avec bumper et suivi de ligne à un mode sans assistance où l'utilisateur se sert d'un joystick de commande en passant par différents modes semi-autonomes.

ROLLAND[7]

Le projet Rolland du groupe Cognitive Robotics dans le département de mathématiques et d'informatique de l'université de Brême (Allemagne) [85] sert de banc d'essais pour la recherche en navigation semi-autonome. Il est utilisé pour tester différents outils et deux prototypes ont été créés. Le premier prototype est équipé de capteur US, IR, bumper, codeurs odométriques et une caméra. Il utilise des outils de localisation à l'aide de la caméra et d'amers visuels mixés avec les données capteurs pour se localiser dans une carte. Il utilise alors cette localisation pour naviguer entre des positions dans cette carte à l'aide d'un système à base de réseau de neurones. Les capteurs US et IR servent aussi à réaliser des comportements tels que l'évitement d'obstacles, le suivi de mur, le passage de portes ou la possibilité de revenir sur ses pas "backtracking". Le choix entre les comportements

[7]http ://www.informatik.uni-bremen.de/rolland/index_e.htm

FIG. 2.12: Le prototype NavChair

s'effectu en fonction d'amers naturels présents dans l'environnement.

Un second prototype [64] (figure 2.11), équipé de capteurs US, de codeurs et d'une caméra, utilise un système plus complexe d'évitement d'obstacles et les comportements : tourner sur place, suivi de mur et retour sur ses pas ainsi qu'un système de choix de comportements basé sur les amers visuels. L'utilisateur apprend au fauteuil les comportements à utiliser sur les trajets. Une fonction de gestion automatique de la vitesse est également implémentée.

Plus récemment un scanner laser a été ajouté au fauteuil pour générer automatiquement des cartes géométriques de l'environnement [86, 34].

NavChair[8]

Le projet NavChair (figure 2.12) été conduit de 1991 à 1999 par l'université du Michigan [9]. Il est équipé de capteurs US et de codeurs. Le NavChair utilise un joystick comme méthode d'entrée et remplace la consigne de l'utilisateur quand c'est nécessaire. Il réalise 3 comportements dont une présentation se trouve dans [53] : l'évitement d'obstacles, le passage de porte et le suivi de mur. Le choix entre ces comportements est automatique [91]. L'évitement d'obstacles est réalisé avec un algorithme dérivé du VFH baptisé MVFH (Minimum Vector Field Histogram) afin de mieux s'adapter aux passages de porte. Bien que le projet NavChair se soit arrêté en 1999. Il est cependant poursuivi par R.C. Simpson à Pittsburg sous le nom de Hepaestus [90] qui développe un module adaptable sans grandes modifications du

[8]http ://www-personal.umich.edu/~johannb/navchair.htm

FIG. 2.13: Le dernier prototype du CCPWNS

fauteuil.

CWA[9]

Le projet CWA (Collaborative Wheelchair Project) du National Institute of Singapore [15] a pour but d'aider les personnes ayant des problèmes d'orientation à naviguer le long d'un chemin pré-enregistré en utilisant uniquement les codeurs. Le nombre restreint d'équipement pour réaliser cette tâche a pour but d'en diminuer le coût. Un système de commande neuronale est également à l'étude pour être utilisé par des personnes ne pouvant utiliser le joystick [84]. Il se base sur un système de bouton clignotant aléatoirement et il capte les variations d'activité du cerveau dues à l'effet de surprise qui apparaît lorsque le bouton correspondant à l'action désirée clignote.

CCPWNS[10]

Le projet CCPWNS, présenté dans la figure 2.13, (Computer-Controlled Power Wheelchair Navigation System) de University of Notre Dame (US) [37] utilise des codeurs sur les roues motrices, des capteurs US et deux caméras CCD pour réaliser des fonctions d'aide à la réalisation de trajectoire. Il utilise un système d'apprentissage manuel des trajectoires dans lequel le fauteuil est soit poussé à la main soit guidé à l'aide du joystick.

[9]http ://guppy.mpe.nus.edu.sg/~rebsamen/project/
[10]http ://www.nd.edu/~gdelcast/

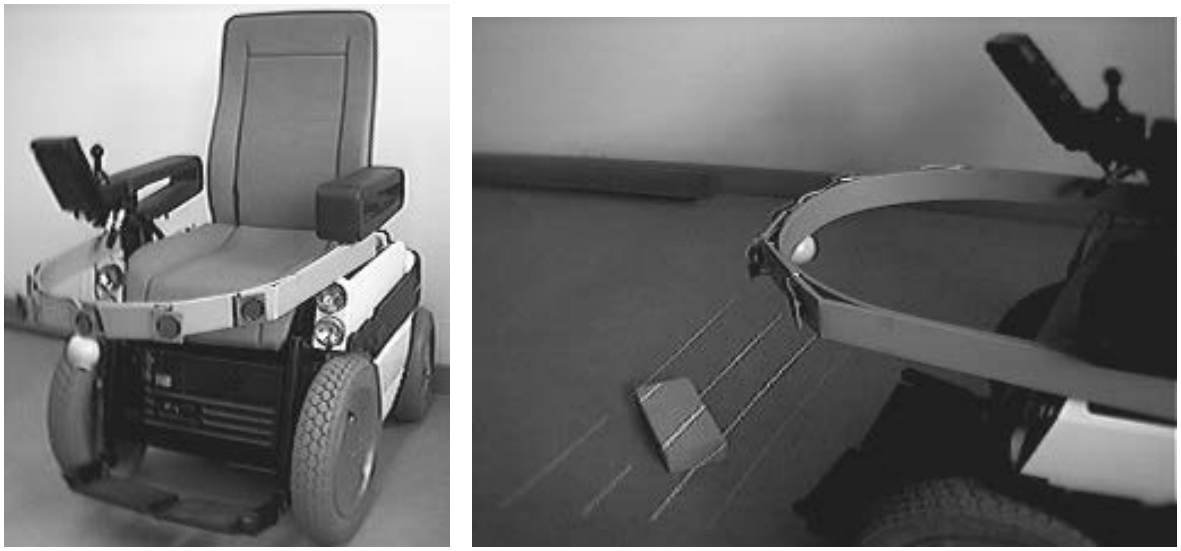


FIG. 2.14: Le prototype INRO et le détail du capteur actif. Le capteur actif utilise un laser projetant trois lignes parallèles permettant à la caméra de mieux voir les petits objets et autres obstacles

INRO

Le projet INRO (Intelligenter Rollstuhl = Fauteuil Intelligent) de Universität Mannheim [87] utilise des capteurs US pointant vers l'avant, des codeurs sur les roues motrices, une caméra CCD avec un marqueur laser et un GPS. Le marqueur laser projette trois lignes parallèles dont les déformations sont mesurées par la caméra (voir figure 2.14) ce qui permet de détecter des petits obstacles ou des escaliers et le GPS n'est utilisé qu'à l'extérieur. Ce fauteuil a la particularité de pouvoir être utilisé en intérieur et en extérieur. Les fonctionnalités présentes sont l'évitement d'obstacles, le suivi d'un fauteuil, la répétition d'itinéraires pré-enregistrés, une aide au "retour à la maison" (en extérieur) et un signal d'alerte pour prévenir le personnel soignant en cas de problème. Le suivi de fauteuil permet la conduite par une infirmière d'un groupe de fauteuil en commandant uniquement le premier fauteuil du groupe, les fauteuils se suivant les uns les autres.

Intelligent Wheelchair System[11]

Le projet Intelligent Wheelchair System du Miura Laboratory de l'Université d'Osaka (Japon) [1] utilise deux caméras et des capteurs US pointant vers l'avant. Une des caméras pointe vers l'avant alors que l'autre pointe vers l'utilisateur. La caméra pointant vers l'utilisateur est utilisée comme

[11]http ://www-cv.mech.eng.osaka-u.ac.jp/research/hi_group/y-adachi/research.html

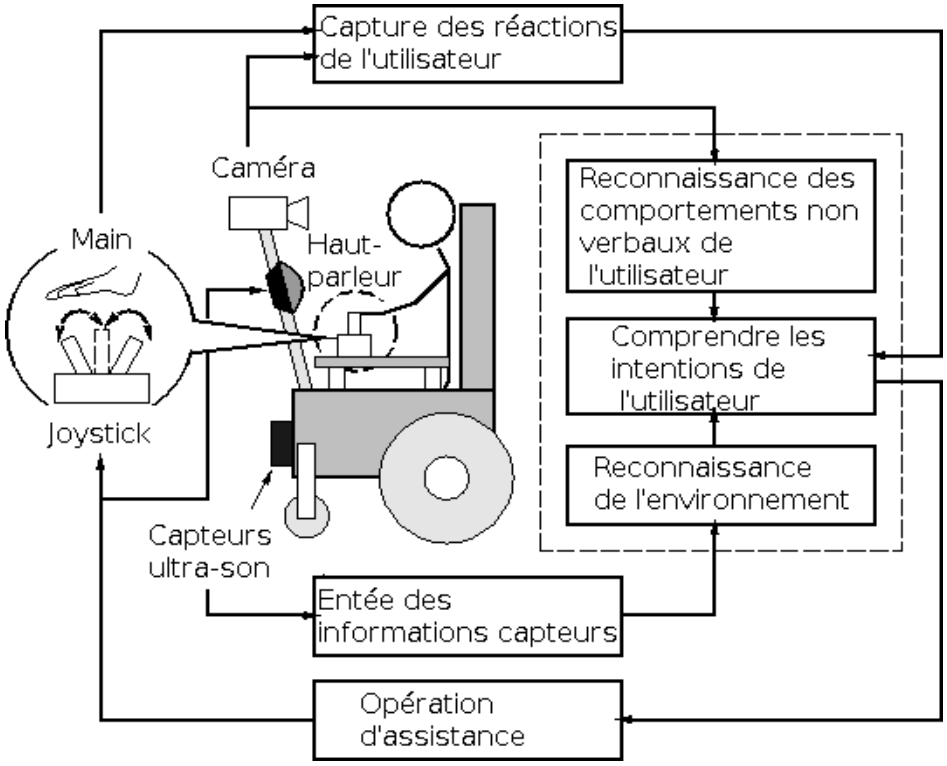


FIG. 2.15: Schéma représentant la fusion des capteurs utilisés sur le projet de fauteuil intelligent de l'Université d'Osaka (Japon)

entrée de consigne à l'aide d'un système de reconnaissance faciale qui permet de reconnaître la direction du visage de l'utilisateur. Cette direction est alors fusionnée avec la consigne lue sur le joystick et les données des capteurs US afin d'extraire une consigne de vitesse et de direction comme le montre la figure 2.15. La deuxième caméra pointe vers l'avant dans le but de réaliser un suivi de cible et également une reconnaissance faciale des personnes venant en face pour contrôler si elles ont bien vu le fauteuil et réaliser un évitement le cas échéant.

MAiD

Le projet MAid (Mobility Aid for Elderly and Disabled People) du laboratoire FAW (Applied Knowledge Processing) de l'université de Ulm (Allemagne) [73] est équipé de codeurs sur les roues motrices, d'un gyroscope optique, de 24 capteurs US autour du fauteuil, de deux scanners IR (capteurs montés sur des servomoteurs), d'un scanner laser et d'un PC embarqué. Il possède deux modes de fonctionnement semi-autonome et autonome. Le mode semi-autonome est utilisé dans les environnements étroits. Il permet à l'utilisateur d'effectuer des manoeuvres locales avec des comportements tel que

FIG. 2.16: Le prototype OMNI

le passage de porte, l'accostage à une table ou le suivi de mur. Le mode autonome est lui utilisé dans les espaces larges et changeant rapidement comme un hall de supermarché ou une gare où il évite les obstacles et les personnes en estimant leurs vélocités. Dans les deux cas la consigne est donnée à l'aide du joystick.

OMNI[12]

Le projet OMNI (Office Wheelchair with High Manoeuvrability and Navigational Intelligence for People with Severe Handicap) du Control Systems Engineering group (PRT) de l'Université de Hagen (Allemagne) [41] (figure 2.16) est équipé de capteurs US et IR montés par bloc d'un US et un IR et dont les valeurs sont fusionnées afin d'augmenter la précision à faible distance (IR) tout en gardant une détection à longue distance (US). La particularité de ce fauteuil est d'être holonome ce qui lui permet de se déplacer de manière transversale et donc de manoeuvrer plus aisément dans les endroits étroits. Les comportements présents sont l'évitement d'obstacles, le retour sur ses pas et la répétition de trajectoires.

RobChair[13]

Le projet RobChair (ROBotic wheelChAIR), présenté dans la figure 2.17, du ISR (Institute of Systems and Robotics) de l'Université de Coimbra Polo

[12]http ://prt.fernuni-hagen.de/pro/omni/omni-eng.html
[13]http ://www.isr.uc.pt/~gpires/frame_index.html?/~gpires/

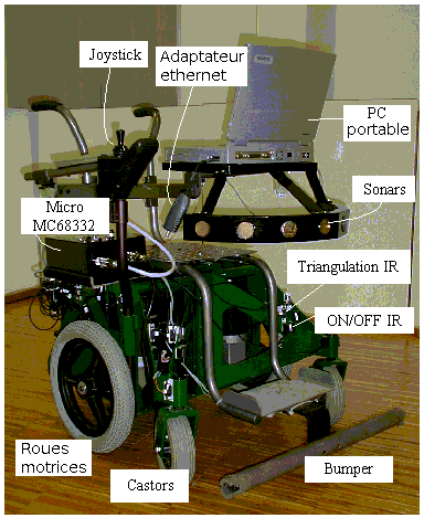


FIG. 2.17: Le prototype RobChair

II (Portugal) [71] est équipé de 14 capteurs de distance IR, 14 capteurs ON/OFF IR , de 7 capteurs US montés à l'avant, un bumper à l'avant et de codeurs sur les roues motrices. La commande du fauteuil s'effectue à l'aide d'un système de commande vocale. Il utilise 3 comportements principaux : l'évitement d'obstacles intelligent, le suivi de contour et la détection de collision qui n'autorise que la marche arrière en cas de collision détectée par le bumper. L'évitement d'obstacles contient également des comportements de passage de porte ou d'accostage qui sont activés explicitement par le système de commande vocale. Le suivi de contour est également activé par commande vocale. La fusion des données des comportements et des commandes vocales est réalisée à base d'un système à logique floue.

SmartChair[14]

Le projet SmartChair du GRASP Laboratory (General Robotic, Automation, Sensing, Perception) de l'Université de Pensilvanie (US) [68] est utilisé comme banc d'essais pour développer des systèmes autour de l'intelligence partagée. Il est équipé de capteurs IR, d'un scanner laser, d'une caméra omnidirectionnelle et de codeurs. Il utilise un système de caméra/vidéo-projecteur comme interface d'entrée visible avec le fauteuil dans la figure 2.18. Les images projetées contiennent des boutons représentant des actions et des directions calculées à partir des images de l'environnement prises par la caméra omnidirectionnelle. Les comportements présents sont l'évitement d'obstacles, le retour sur ses pas ou un déplacement autonome

[14]http ://www.cis.upenn.edu/group/smartchair/

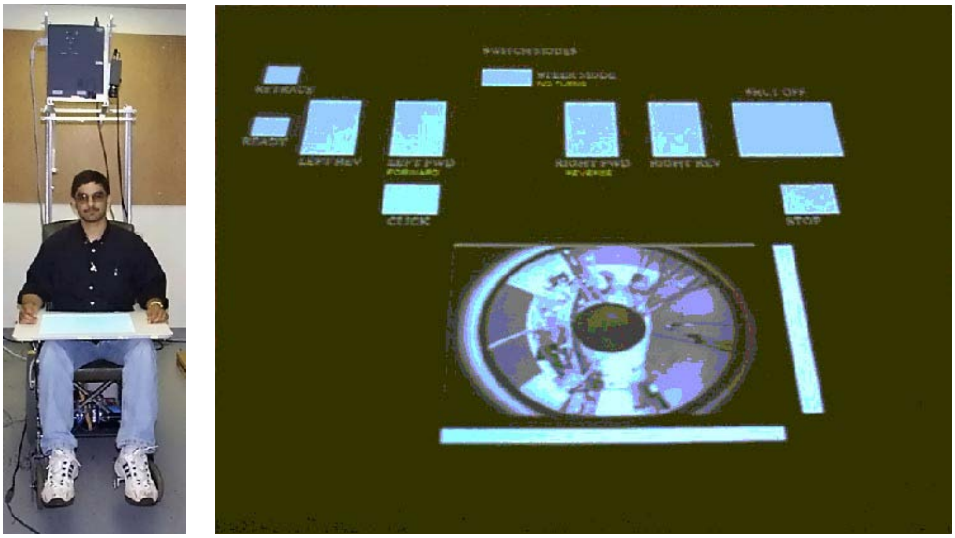


FIG. 2.18: Le prototype SmartChair et son interface projetée sur la tablette du fauteuil.

jusqu'à un point indiqué par l'utilisateur.

WAD[15]

Le projet WAD (Wheelchair Attractor Dynamics) de l'université des Sciences et du Sport de l'Université de la Méditerranée [55] (figure 2.19) est équipé de 20 capteurs IR répartis autour du fauteuil et des codeurs sur les roues motrices. Il fonctionne soit en mode sans assistance soit en mode autonome. Dans ce dernier cas l'utilisateur choisit la pièce dans laquelle il désire aller sur l'écran du PC portable embarqué à l'aide d'une interface à défilement ou du joystick.

WATSON[16]

Le projet WATSON du robotics laboratory au Nara Institute of Science en Technology [57] contient deux prototypes présentés dans la figure 2.20. Le WATSON1 est équipé de 7 capteurs US pour réaliser un évitement d'obstacles. Le second prototype, WATSON2, est équipé de 2 caméras et d'un scanner laser. Il utilise les caméras pour réaliser une commande par mouvement de la tête. Le scanner laser permet de réaliser un évitement d'obstacles. Il réalise également une tâche de localisation.

[15]http ://www.laps.univ-mrs.fr/~mallet/SmartWheelchairProject2.html
[16]http ://robotics.naist.jp/research/watson/index_e.html

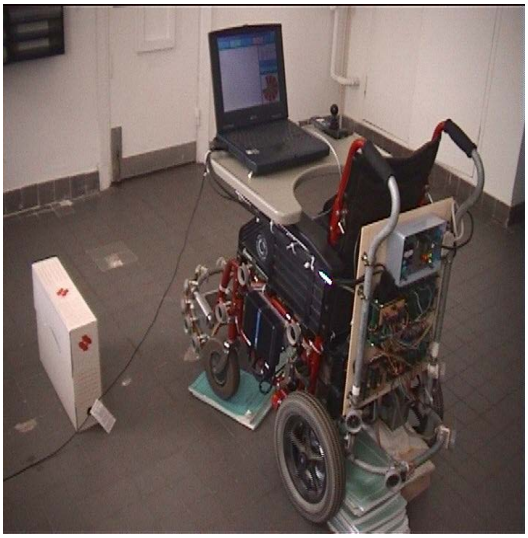


FIG. 2.19: Le prototype WAD

FIG. 2.20: Les prototypes WATSON 1 et 2

FIG. 2.21: VAHM2 (à gauche) et VAHM3 (à droite)

VAHM[17]

Le projet VAHM du Laboratoire d'Automatique et des Systèmes Coopératifs (LASC) est composé de 3 fauteuils intelligents. Le premier, qui n'est plus en service, était composé d'une base Robuter sur laquelle était montés un fauteuil. Il réalisait une navigation semi-autonome et autonome [13]. Les deux autres fauteuils intelligents, montrés dans la figure 2.21, sont des fauteuils du commerce et sur chacun des fauteuils ont été ajoutés 16 capteurs US répartis autour du fauteuil. L'idée du projet est d'obtenir une symbiose entre l'utilisateur et le fauteuil [78] et ils réalisent une navigation semi-autonome. Ils utilisent différents comportements avec une sélection automatique basée sur l'utilisation d'une base de cas [38].

2.4 La localisation en robotique mobile

Dans certains cas il est très difficile pour l'utilisateur de définir, de maintenir une commande de direction ou de planifier un trajet. Dans ces cas c'est au système de tenter de pallier ce problème et, afin de permettre au fauteuil de réagir convenablement, il est nécessaire que celui-ci connaisse sa position dans l'environnement dans lequel il évolue. Ce problème est identique dans la robotique mobile autonome puisque, pour naviguer dans un envi-

[17]http ://www.lasc.univ-metz.fr/article.php3?id_article=41

ronnement, il est nécessaire que le robot puisse se localiser avant d'effectuer une tâche de planification de trajet. Une fois la trajectoire planifiée, il peut connaître la direction dans laquelle il doit se déplacer pour atteindre son but qui est souvent d'atteindre une position dans l'environnement. Nous allons donc étudier les méthodes existantes à la fois en robotique mobile et dans le domaine des fauteuils intelligents. Pour simplifier, nous utiliserons le terme robot pour désigner à la fois les robots autonomes et les fauteuils intelligents.

Une étude plus complète des outils de navigation se trouve dans les études complémentaires[31, 60] qui présentent un état de l'art sur la localisation, l'apprentissage des modèles d'environnement et la planification de trajet. La navigation en robotique repose sur 3 processus :

- L'apprentissage d'une carte de l'environnement qui mémorise les données par le robot pendant l'exploration dans un format adéquat.
- La localisation qui déduit la position du robot sur cette carte.
- La planification de trajectoire qui choisi la suite d'action à effectuer pour atteindre un but en partant de la position actuelle.

Étant donné que dans notre étude c'est l'utilisateur qui réalise la planification de trajectoire, nous ne nous intéresserons qu'aux deux premiers points. Mais ces points sont étroitement liés entre eux du fait qu'une carte soit nécessaire pour se localiser et l'estimation de la position soit nécessaire pour construire une carte de l'environnement. Cette étroite relation fait partie du problème de la localisation et de la construction de la carte simultanée, aussi appelé SLAM (Simultaneous Location And Mapping).

Nous allons donc dans un premier temps étudier les différents types de modèles utilisés pour la localisation avant de nous intéresser à différentes méthodes de localisation.

2.4.1 Les modèles d'environnements

Les environnements utilisés en localisation et navigation peuvent être répartis selon trois grandes classes :

- les modèles géométriques,
- les modèles à grille d'occupation,
- les modèles topologiques.

Les deux premiers types de modèles représentent l'environnement comme un ensemble d'objets avec des coordonnées dans un espace en 2D . Le troisième modèle représente l'environnement comme un ensemble de lieux

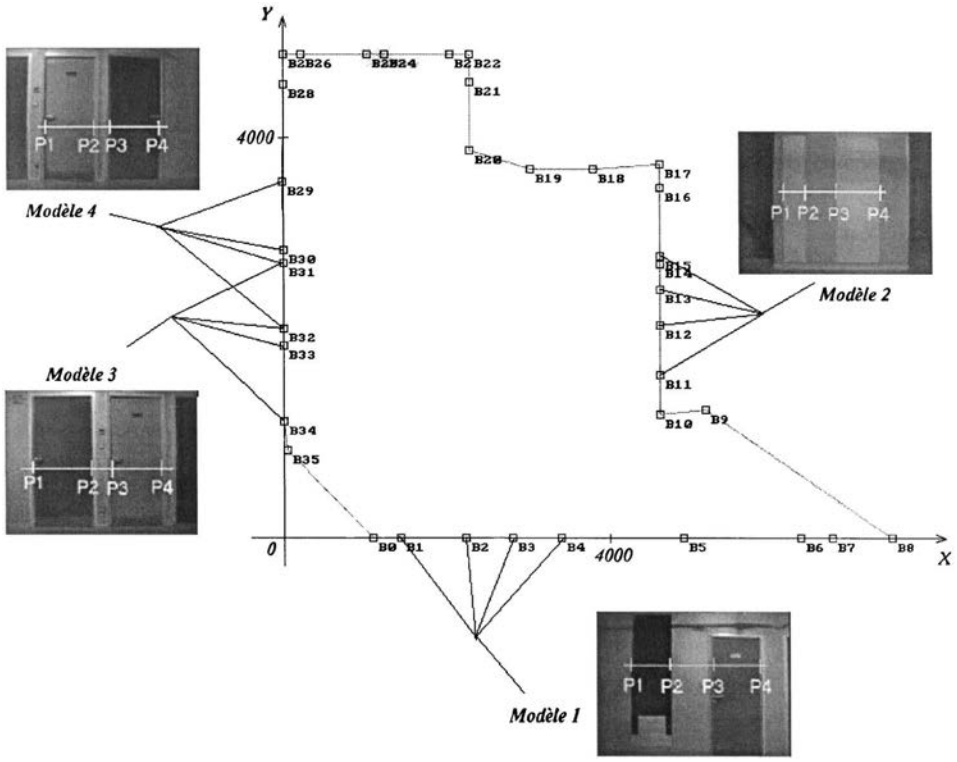


FIG. 2.22: Exemple de carte géométrique extraite de [56]. Cette carte représente les positions des verticales qui sont comparées à celles extraites de l'image omnidirectionnelle.

distinctifs et par le chemin que peut prendre le robot pour aller d'un lieu à un autre.

2.4.1.1 Modèles géométriques

Les modèles géométriques décrivent l'environnement à l'aide de primitives géométriques (points, droites, arcs de cercle, ...) représentant les informations comme les obstacles ou des caractéristiques de l'environnement tels que des angles de murs comme montré dans la figure 2.22 extraite de [56].

Ces modèles peuvent être réalisés pour tous les capteurs de mesure de distance présents sur les robots. Ainsi dans [26] l'auteur extrait des segments de droite à partir du nuage de points provenant des capteurs US. Il reconstruit ainsi l'environnement direct du robot et le compare avec le modèle de l'environnement global de ce dernier afin d'en estimer la position.

Dans [4] les auteurs utilisent un scanner laser pour percevoir l'espace libre autour du robot et le modélisent sous forme de droites. Ils utilisent également une caméra pour retrouver les lignes verticales de l'environnement qui correspondent généralement aux intersections des murs. Un filtre de Kalman est alors utilisé pour corriger la position du robot en fonction de

la carte locale ainsi construite et de la carte globale de l'environnement.

D'autres systèmes, tel que celui de [56], utilisent uniquement un système de vision pour retrouver les horizontales et/ou verticales de l'environnement qu'ils comparent avec le modèle global de l'environnement pour estimer ou corriger la position du robot.

Dans [48] deux méthodes sont proposées. La première se sert des données des capteurs US pour construire un modèle local constitué de points qui sont mis en correspondance avec le modèle géométrique global. Un algorithme des moindres carrés est utilisé pour minimiser la distance entre ces deux modèles. La seconde se sert une caméra pour détecter les verticales dans l'environnement et utilise un arbre d'appariement pour rechercher les correspondances entre les verticales stockées dans le modèle et celles détectées par la caméra.

2.4.1.2 Modèles à grille d'occupation

Les modèles à grille d'occupation représentent l'environnement sous la forme d'une grille dans laquelle chaque cellule correspond à une portion de l'espace et à laquelle on associe une valeur correspondant à sa probabilité d'occupation. Un exemple de représentation d'un modèle à grille est donné dans la figure 2.23. La construction et la mise à jour de carte peuvent être réalisées selon plusieurs méthodes telles que la logique floue [22, 67], des méthodes bayésiennes [21] ou encore des méthodes de MONTE CARLO [27].

Les modèles à grille d'occupation sont construits à partir des données provenant des capteurs. Elles permettent d'introduire l'incertitude de la mesure dans la création de la grille. Cette incertitude est introduite dans la grille en ajoutant les probabilités de détection provenant du modèle de capteur.

2.4.1.3 Modèles topologiques

Les modèles topologiques de l'environnement peuvent être construits sans aucune connaissance des caractéristiques métriques de l'environnement. Ils représentent la structure ou le squelette de l'environnement sous forme de noeud et d'arc. Les noeuds comportent uniquement des caractéristiques tels que des lieux (cuisine, chambre,...), des objets (lit, canapé, ordinateur ...) ou encore des points caractéristiques de l'environnement (porte, couloir,...) tandis que les arcs représentent les liaisons existantes entre ces

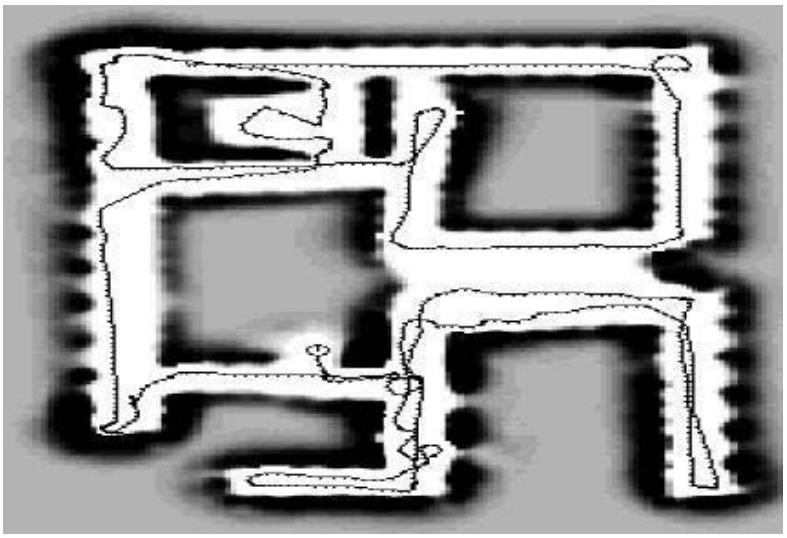


FIG. 2.23: Exemple de carte à grille d'occupation. Cette carte est extraite de [95]. Les zones claires représentent les espaces les plus probablement libres et les zones sombres représentent les obstacles de l'environnement.

noeuds. Un exemple de carte topologique tirée de [2] est visible dans la figure 2.24.

Le principal avantage de cette modélisation est de déterminer le cheminement pour aller d'un noeud à un autre.

Cependant le fait qu'un modèle topologique contienne peu ou pas d'informations métriques, il est difficile de les utiliser directement dans la localisation. C'est pourquoi ils sont souvent couplés avec un modèle métrique afin de faciliter la localisation comme dans [95].

2.4.2 Les méthodes de localisation

Il existe trois types de localisations différents . La plus simple est celle qui utilise uniquement les capteurs proprioceptifs (odométrie, accéléromètre, boussole, gyroscope,...) pour calculer les déplacements du robot [29, 75]. Cependant, ces systèmes réalisant généralement une intégration ou une double intégration de la mesure, leurs erreurs de mesure, de calibration et de modélisation sont ajoutées sans bornes dans le résultat. Séparées, elles ne représentent donc pas une manière fiable de se localiser, mais des méthodes de fusion de données améliorent ces résultats [7, 23, 48]. Cette méthode ne donne cependant pas de localisation absolue mais une localisation relative au point de départ du robot et elle n'est pas toujours présentée dans les méthodes de localisation.

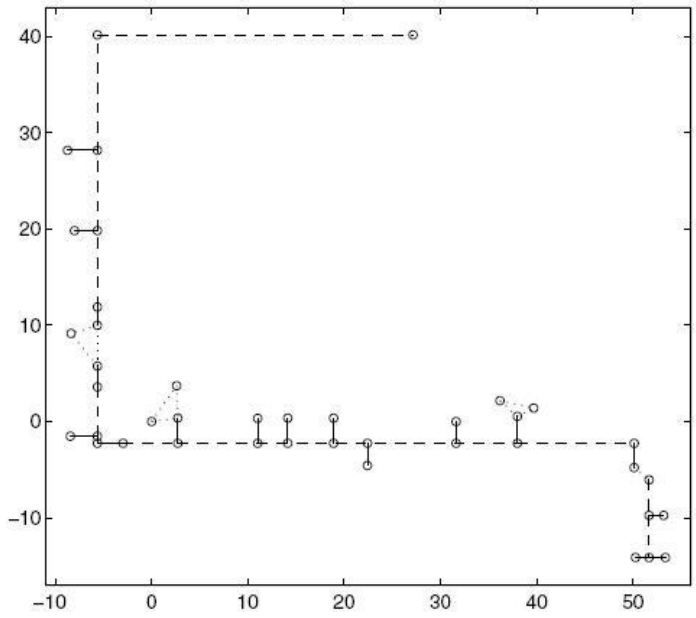


FIG. 2.24: Exemple de carte topologique extraite de [2]. Les ronds représentent les noeuds et les arcs sont de trois types différents : les couloirs (en tirets), les pièces (en pointillés) et les portes (en traits pleins).

Néanmoins, l'odométrie est un outil de base dans tous les modes de localisation afin de donner une information de déplacement. Les erreurs induites sont alors corrigées par les outils de localisation [48] à l'aide des mesures faites par les capteurs.

Les deux autres méthodes réalisent une localisation globale sur une carte de l'environnement [31]. La première de ces méthodes de localisation globale utilise une estimation précédente de la position et les données des capteurs pour estimer la position actuelle. Cette méthode est généralement référencée sous le nom de suivi de position (*position tracking*). Cependant elle n'est possible que si le robot possède une connaissance *a priori* de sa position. Dans le cas contraire, une méthode de localisation globale est nécessaire. Cette méthode utilise uniquement les données des capteurs pour localiser le robot dans l'environnement et elle permet de résoudre le problème du robot perdu ou du robot kidnappé [96].

Dans ces deux cas, la localisation nécessite donc de mettre en corrélation la perception de l'environnement avec un modèle d'environnement. On distingue deux types de localisation couramment utilisés. Le premier type de localisation utilise une méthode de balises ou amers avec, en général, une méthode de triangulation pour se localiser [76]. Le second se sert les données provenant des capteurs de distance (US, IR, laser), soit directement

soit dans un modèle local centré sur le robot, qu'il compare avec le modèle global de l'environnement.

2.4.2.1 Localisation à l'aide de balises

Dans ce type de localisation, le robot utilise des balises artificielles qui sont présentes dans l'environnement dans lequel le robot évolue. Celles-ci sont facilement détectables et leurs positions sont connues par le robot. Lorsque le robot perçoit au moins trois balises simultanément, il mesure, soit l'angle entre lui et les balises, soit la distance qui le sépare de celles-ci. Un calcul respectivement de triangulation ou de trilatéralisation est effectué, ce qui permet au robot de connaître sa position dans l'environnement [76].

2.4.2.2 Localisation autonome

La localisation autonome, avec uniquement les capteurs présents sur le robot, n'est pas aussi simple qu'avec des balises. Parmi de nombreuses méthodes [96, 94, 5, 11, 21, 26, 27, 28, 33, 34, 35, 44, 47, 67], certaines se rapprochent de système à balises en utilisant des amers naturels qui sont des points remarquables naturellement présents dans l'environnement [51]. Les autres méthodes directes réalisent alors un appariement entre les données capteurs et le modèle [69].

D'autres systèmes utilisent une représentation de l'environnement local centrée sur le robot. Cette représentation peut être, comme pour les modèles d'environnement, topologique, géométrique ou à grille d'occupation [31].

Les modèles métriques représentent l'environnement proche, visible par les capteurs, sous forme géométrique ou à grille d'occupation. Les positions des obstacles qu'elles contiennent sont mises à jour à l'aide des données provenant des capteurs et les coordonnées des éléments géométriques ou des valeurs des cellules qui sont déplacées en fonction de l'odométrie. La localisation s'effectue alors par appariement entre cette grille et le modèle. L'avantage de cette méthode est que la mise à jour de la carte locale peut se faire également à partir des données provenants des capteurs et de la carte de l'environnement global. Cela permet alors de se localiser avec une carte partielle de l'environnement tout en ajoutant les nouvelles zones dans la carte globale. Cette méthode s'appelle la *cartographie et localisation concurrente* (CLC) aussi appelée *cartographie et localisation simultanées*, CML (Concurrent Mapping and Localization) ou encore SLAM (Simultaneous

Location And Maping) [3, 52, 60].

2.4.2.3 Algorithmes d'appariement

Les algorithmes d'appariement les plus courants sont issus des Filtres de Kalman [4, 51], des moindres carrés [25, 11], des Modèles de Markov [33, 88] et de l'algorithme CONDENSATION (aussi appelé filtre particulaire ou filtre de MONTE CARLO) [27, 44]. Cet appariement peut, soit être réalisé en fonction d'une position précédente connue, soit sans *a priori* sur la position du robot [31].

Kalman Le Filtre de Kalman en robotique mobile effectue un calcul d'estimation de la position à partir de la position précédente, d'un modèle géométrique de l'environnement, d'une position *a priori* du robot, d'un modèle d'évolution et d'un modèle de mesure. Le modèle d'évolution représente l'évolution de la position du robot en fonction de l'entrée de commande et le modèle de mesure représente une estimation des observations des capteurs en fonction de la position *a priori* et du modèle de l'environnement. Le filtre de Kalman ne peut cependant pas être utilisé lorsque la position précédente n'est pas connue. Il est donc sensible au phénomène du robot « téléporté », c'est-à-dire qu'au cas où le robot ait été déplacé alors qu'il était éteint. Ne connaissant pas sa position de départ, il lui est impossible de se localiser. Ce problème apparaît également avec la méthode des moindres carrés puisque, dans ce cas, le but est d'essayer de réduire l'erreur entre une estimation de la position et la position réelle.

Modèles de Markov Les algorithmes de localisation issus des modèles de Markov les plus courants sont les HMMs (*Hidden Markov Models*) et les POMPDs (*Partialy Observable Markov Decision Process*) dont une description détaillée des modèles de Markov se trouve dans la partie 4. La différence principale entre ces deux algorithmes est que les POMPDs intègrent des informations utiles à la navigation en plus de l'outil de localisation qui est similaire dans les deux algorithmes. Les modèles de Markov sont des réseaux bayésiens constitués de noeuds et de transitions. On associe à chaque noeud une probabilité de présence, une position et une rotation possibles ($\{x, y, \theta\}$) qui sont la partie "cachée" du modèle et des probabilités d'observations qui sont liées aux valeurs des capteurs. Les transitions sont pondérées par la probabilité de passer d'un état à l'autre. L'évolution s'effectue lorsque le robot avance. Les valeurs odométriques donnent des

informations de différence de position et de rotation qui sont utilisées, sous forme de gaussienne, pour la propagation des probabilités de présence des états. Ainsi en avançant, le robot augmente les probabilités de présence sur les noeuds présentant des observations similaires à celles qu'il vient de faire et diminue les autres probabilités. Le système converge alors successivement vers les noeuds représentant les positions du robot au cours de son trajet. L'avantage de ce système est de pouvoir se « re-localiser » en cas de perte de localisation ou de « téléportation » du robot.

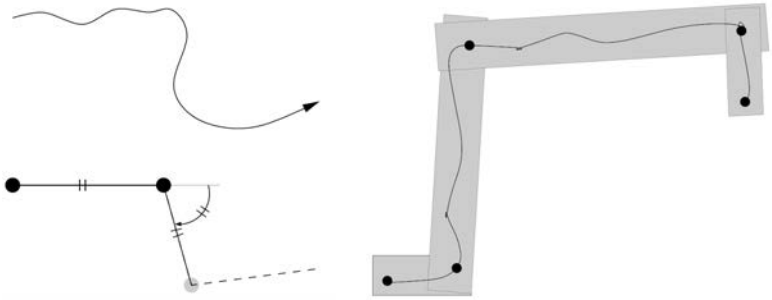
CONDENSATION Les algorithmes liés à l'algorithme CONDENSATION, dont une explication détaillée se trouve dans la partie 5.4, partent du principe qu'il n'est pas concevable de tester toutes les solutions de positions possibles à chaque instant. Un échantillonnage est donc réalisé parmi toutes les positions possibles et celles étant les plus proches des observations sont propagées à l'instant suivant en incorporant également des nouvelles positions prises au hasard. Ce système possède également l'avantage de pouvoir se localiser sans *a priori* sur sa position et il est capable de se « re-localiser » après une « téléportation ».

2.4.2.4 Autres modes de localisation

Cependant la localisation ne se limite pas aux seuls éléments que nous venons de voir. D'autres systèmes ne nécessitant pas une position géographique exacte du robot existent. C'est le cas du système *RoutLoc* présenté dans [49] qui analyse les données odométriques pour extraire, comme le montre la figure 2.25(a), des informations de longueurs de segments et d'angles entre deux segments consécutifs. La limite utilisée pour définir le changement de couloir, représentée par les rectangles dans la figure 2.25(b), est obtenue avec un système basé sur les histogrammes des capteurs US pointant à gauche et à droite du fauteuil. La localisation est effectuée par comparaison entre les segments reconnus et une carte topologique contenant les informations d'angles et de longueurs de couloirs.

2.5 Conclusion

Après avoir présenté les besoins liés aux différents types de handicaps, nous avons présenté quelques projets de fauteuils intelligents ainsi que les principaux modes de fonctionnement et les différents types de capteurs pré-



(a) Ségmentation des donnés de l'odométrie (b) Représentation de l'environnement

FIG. 2.25: Segmentation des données odométriques utilisées sur le fauteuil intelligent "Rolland" [49]. Les données retenues sont la longueur du segment et son angle avec le segment précédent.

sents dans les fauteuils intelligents. Pour finir nous avons vu les différents modes de localisation existants en robotique mobile.

Pour conclure on peut dire les besoins concernent principalement l'interface d'entrée de commande et l'aide à la navigation. Dans la littérature, la plupart des projets utilisant l'aide à la navigation avec localisation utilisent le fauteuil comme moyen de transport où l'utilisateur entre sa destination et se laisse guider par le fauteuil.

La partie suivante présente en détail le système VAHM et les motivations qui entourent ce mémoire.

3 Le système VAHM

LE projet Véhicule Autonome pour Handicapé Moteur (VAHM) se place dans le cadre des fauteuils intelligents. Il a pour rôle essentiel d'aider une personne handicapée à se déplacer dans un environnement plus ou moins connu et plus ou moins variable dans le temps. Le point le plus important de ce projet est de créer une symbiose entre l'utilisateur et le fauteuil ce qui implique que l'utilisateur ne se sente pas transporté dans le fauteuil mais soit toujours maître de ses mouvements.

3.1 Introduction

Dans le cadre de la conduite d'un fauteuil par un utilisateur handicapé, il est nécessaire que celui-ci soit maître de son véhicule. Cependant sa situation d'handicap physique ne lui permet pas de définir avec précision ce mouvement, ni de le maintenir longuement. Pour ces raisons, la machine peut lui apporter une assistance sur deux plans :

- La définition du mouvement à imposer au fauteuil est réalisée à l'aide d'une interface (joystick ou système à balayage [78]) qui, en général et selon le handicap, traduit l'action à générer en terme de direction. Le fauteuil prend à sa charge l'évaluation de la pertinence de cette information et la traduit en comportement à effectuer. La demande de l'utilisateur est traduite en comportement afin de garantir au système une plus grande stabilité dans la commande et une meilleure gestion des erreurs de définition.
- La planification du mouvement globale est déterminée entièrement par l'utilisateur, par contre, le fauteuil vient assister la personne dans la reproduction des chemins fréquemment employés. Si à la sortie d'une pièce particulière l'utilisateur a l' « habitude » d'aller dans une direction, il serait intéressant que le système rende cette direction privilégiée en évitant ainsi la nécessité d'établissement d'une nouvelle commande de direction.

FIG. 3.1: Détail du capteur TOR utilisé lors des tests.

L'idée est alors de considérer le système global Homme-Machine. Les études précédentes ont abouti à la définition d'une structure multi-agent. Elle réalise la reconnaissance de l'état du système et de son environnement proche et permet de définir le comportement le plus approprié afin de réaliser le mouvement désiré par l'utilisateur. Elle permet de réaliser ainsi une symbiose entre les deux parties dans laquelle la machine assiste l'homme dans la réalisation de son mouvement [74]. On parle alors de système symbiotique.

Nous allons maintenant voir en détails les différents éléments qui composent le VAHM en commençant par la partie matérielle puis par la partie logicielle.

3.2 Architecture matérielle.

Le projet VAHM s'articule autour de fauteuils roulants du commerce équipés de capteurs ultra-sons pour la perception de l'environnement, de codeurs placés sur les axes de moteurs, d'une entrée prévue pour un capteur TOR (Tout Ou Rien) (figure 3.1). Le tout étant relié à un calculateur embarqué (un PC complet) assurant la partie intelligente du fauteuil ainsi que l'interaction homme-machine. Les fauteuils actuellement disponibles sont les VAHM 2 et VAHM 3 (figure2.21).

La principale différence entre les versions 2 et 3 du VAHM se situe au niveau de la position des roues motrices. Celles-ci sont montées à l'avant pour le VAHM 2 et à l'arrière pour le VAHM 3. Cette différence change le comportement du fauteuil notamment dans l'approche des passages de portes.

Les deux versions sont équipées de 16 capteurs ultra-sons répartis autour du fauteuil comme montré dans la figure 3.2. Ces capteurs, qui servent à avoir une vision des obstacles présents autour du fauteuil, sont de type Polaroïds et ont une portée comprise entre 12cm et 3 mètres. Leur répartition

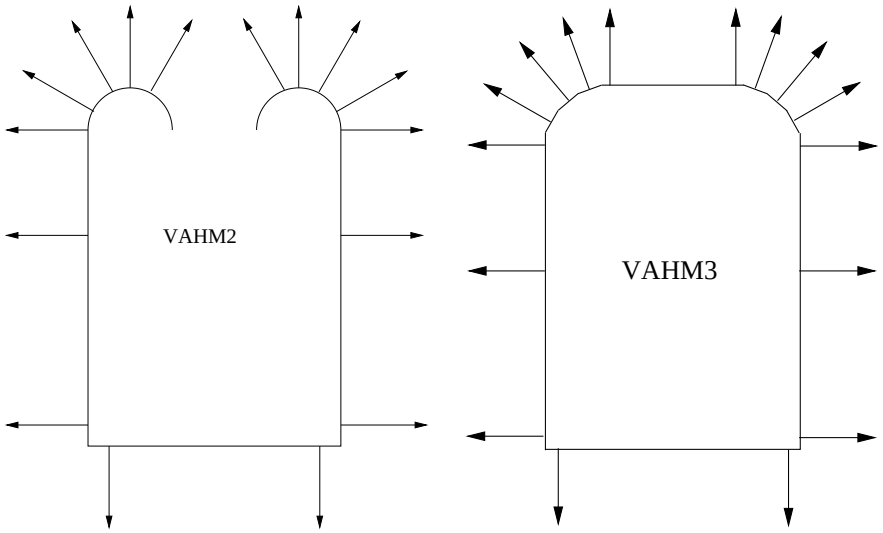


FIG. 3.2: Répartition des capteurs ultra-son sur les VAHM2 et VAHM3

a été définie afin d'obtenir une bonne vision vers l'avant et les côtés. L'arrière ayant moins besoin d'une vision puisque l'utilisateur, étant dans l'impossibilité de voir derrière, ne peut pas donner de direction pertinente vers l'arrière. C'est pourquoi l'architecture logicielle du fauteuil lui permet de revenir sur ses pas plutôt que d'effectuer une réelle marche arrière [39, 38]. Ces capteurs sont reliés à un micro-contrôleur qui effectue 10 mesures par seconde et qui envoie ces mesures au PC par liaison série.

Chaque fauteuil est équipé de deux codeurs incrémentaux montés sur l'axe des moteurs. Ces codeurs sont reliés au PC via une carte spécialisée qui compte les impulsions et donne également l'information de sens de rotation.

La commande des moteurs est réalisée au travers du système du fauteuil. Le boîtier de commande original a été modifié afin de remplacer les tensions de commande du joystick de contrôle par des commandes analogiques fournies par la carte d'entrées/sorties pilotée par le PC. Un sélecteur a également été ajouté dans le boîtier de commande afin de sélectionner soit les commandes issues du joystick, soit les commandes issues du calculateur.

Les PC équipant les VAHMs sont des PC industriels alimentés directement sur les batteries du fauteuil. Les PC sont équipés d'un écran LCD couleur pour l'interaction homme-machine et d'un ensemble clavier/souris pour le déverminage des applications. Le VAHM 2 est équipé d'un PII cadencé à 300Mhz avec 128Mo de RAM, le VAHM 3 quant à lui est équipé d'un P4 2.66Ghz avec 512Mo de RAM fonctionnant sous Windows XP Pro.

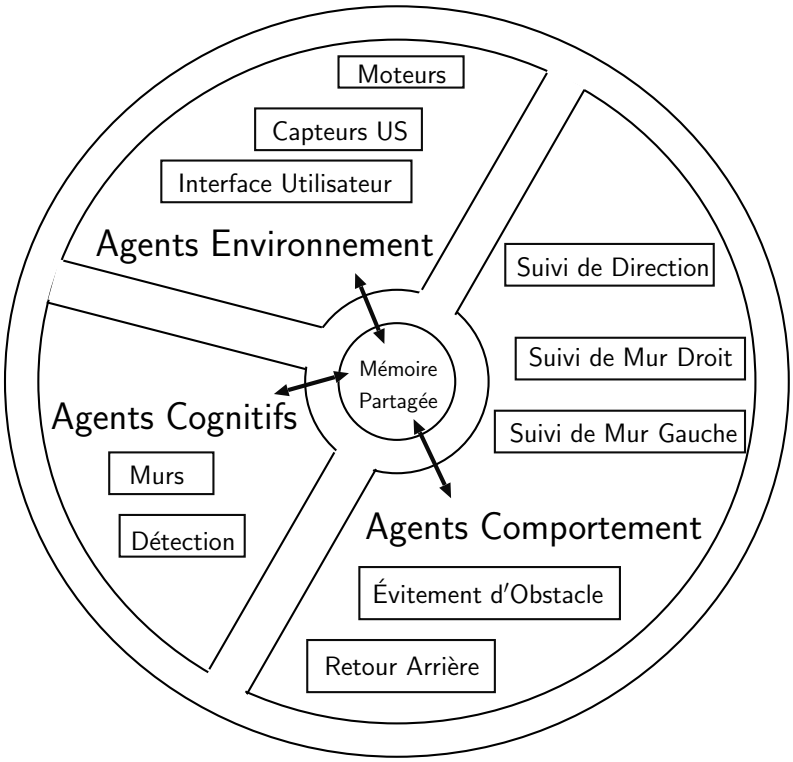


FIG. 3.3: Structure multi-agent

3.3 Architecture de commande

Le logiciel du VAHM est basé sur un système multi-agent [77] représenté dans la figure 3.3. Les agents sont autonomes et communiquent entre eux à l'aide d'une mémoire partagée.

Les agents utilisés, détaillés ci-dessous, sont de 3 types différents : les agents comportement, les agents cognitifs et les agents environnement. Bien qu'il ne soit pas vraiment un agent mais comme il partage également la mémoire commune, le simulateur est également présenté ci-dessous.

3.3.1 Les agents comportement

Les agents comportement définissent les réactions du fauteuil en fonction de l'environnement et des commandes de l'utilisateur [74]. Le choix du comportement actif sur le fauteuil est effectué à l'aide d'un raisonnement à partir de cas [38]. Cette méthode consiste à rechercher dans une liste de cas, celui qui correspond le plus au cas actuel que l'on désire traiter. Chaque agent comportement possède une liste de cas idéaux correspondants à un ensemble de paramètres de fonctionnement du système. Le raisonnement à partir de cas calcule une fonction de similitude pour chaque cas dont la

valeur maximale du résultat correspond à la pertinence de l'utilisation de cet agent. C'est là qu'intervient la mise en concurrence des comportements. Chaque valeur de pertinence est accompagnée des commandes de mouvement correspondantes. L'agent moteur applique alors les consignes de vitesses linéaires et angulaires accompagnant la pertinence la plus élevée.

L'avantage de cette méthode est la robustesse par redondance. Chaque comportement est défini par ses entrées qui elles-mêmes sont décrites ou calculées à l'aide des informations issues des capteurs.

La multiplication des comportements activés par des entrées ayant des sources différentes et leur fonctionnement parallèle permettent de garantir une bonne robustesse. Le non fonctionnement d'un agent par le fait d'absence d'information source ou pour d'autres raisons fonctionnelles sera immédiatement remplacé par un autre agent. Certainement pas aussi efficace dans l'exécution de la tâche mais qui ne rend pas la situation impossible à gérer. Dans le même ordre d'idée, le retrait ou l'ajout d'un agent est transparent pour l'ensemble du fonctionnement du système.

L'architecture reposant sur une séquence de comportements permet l'émergence de comportements globaux face à certaine situation. C'est notamment le cas lorsque, face à un obstacle, le fauteuil se dégage avant de l'éviter.

Les comportements présents sur le fauteuil sont répartis en deux familles, les comportements réflexes et les comportements de suivi de vecteur. La famille des comportements réflexes ne compte que l'évitement d'obstacles tandis que celle des comportements de suivi de vecteur contient les comportements de suivi de direction, de suivi de mur et de retour sur ses pas.

3.3.1.1 Évitement d'obstacles

Ce comportement utilise l'algorithme décrit en détail dans [75] qui associe une direction à chaque capteur comme le montre la figure 3.4. La direction de l'évitement, donc de la vitesse angulaire, est donnée par la somme des angles pondérés par l'inverse de leur distance de détection selon la fonction suivante :

$$\theta = \frac{\sum_{i=1}^{m} \frac{\theta(i)}{dist(i)}}{\sum_{i=1}^{m} \frac{1}{dist(i)}}$$

où m est le nombre de capteurs, $\theta(i)$ est l'angle associé au capteur i, $dist(i)$ est la distance mesurée par le capteur i et θ est l'angle instantané de l'évitement. Ainsi un capteur voyant un objet proche aura plus de poids qu'un capteur voyant un objet plus lointain.

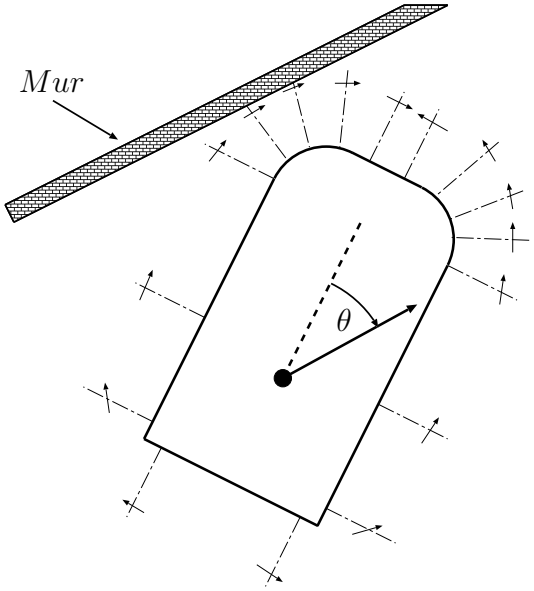


FIG. 3.4: Schéma de fonctionnement de l'évitement d'obstacles. La vitesse angulaire est fonction de l'angle associé à chaque capteur et de sa distance de détection.

3.3.1.2 Suivi de direction

Ce comportement est le plus simple des comportements car il ne fait que répéter les commandes de direction de l'utilisateur. Il est utilisé dans des espaces vides ou dans les déplacements dans des directions dans lesquelles il n'y a pas d'obstacles.

3.3.1.3 Suivi de mur

Le suivi de mur est réalisé par un agent comprenant deux comportements qui sont le suivi de mur droit et le suivi de mur gauche. L'utilité d'un comportement suivi de mur est d'éviter une variation de direction qui peut survenir lorsque l'on passe à coté d'une porte.

3.3.1.4 Retour sur ses pas (*backtracking*)

Ce comportement est utilisé quand l'on doit effectuer une marche arrière avec le fauteuil. L'utilisateur du fauteuil n'ayant aucune vision vers l'arrière, il est préférable lors d'un déplacement vers l'arrière de revenir sur ses pas plutôt que d'aller droit derrière, un obstacle non détecté par les capteurs pouvant s'y trouver.

Les valeurs utilisées pour le calcul de pertinence de chaque cas et les

valeurs des murs sont issues des agents cognitifs.

3.3.2 Les agents cognitifs

Les agents cognitifs servent à donner des informations construites à partir des données des capteurs aux autres agents. Le projet VAHM compte 2 agents cognitifs : l'agent de calcul des murs et l'agent de calcul des détections.

3.3.2.1 L'agent Mur

L'agent de calcul des murs sert à calculer la distance et l'angle des murs à droite et à gauche du fauteuil. Il utilise pour cela les 6 capteurs US (3 de chaque côté) qui pointent sur les côtés du fauteuil. Les calculs des murs sont indépendants et ils sont effectués tant que les mesures des 3 capteurs composant le mur ne dépassent pas 2m.

3.3.2.2 L'agent Détection

L'agent détection calcule les distances de détection en fonction des zones prédéfinies (devant, à droite, à gauche, derrière) qui servent de référence dans la base de cas (présentée dans la section 3.3.1) pour le calcul de la pertinence. Cet agent retourne la distance minimale pour chaque zone de détection.

Tous les agents que nous avons vus jusque là n'ont aucune action directe avec l'extérieur. Ils passent par les agents d'environnement.

3.3.3 Les agents d'environnement

Les agents d'environnement regroupent tous les agents qui interagissent avec l'extérieur du fauteuil. Ces agents comprennent :

- l'interface utilisateur,
- la gestion des commandes des moteurs et de l'odométrie par lecture des codes incrémentaux présents sur les roues,
- la gestion de la lecture des capteurs US et de la consigne utilisateur.

L'utilité de cette séparation est de permettre de modifier le fauteuil ou les types de capteurs en toute transparence pour le reste du système.

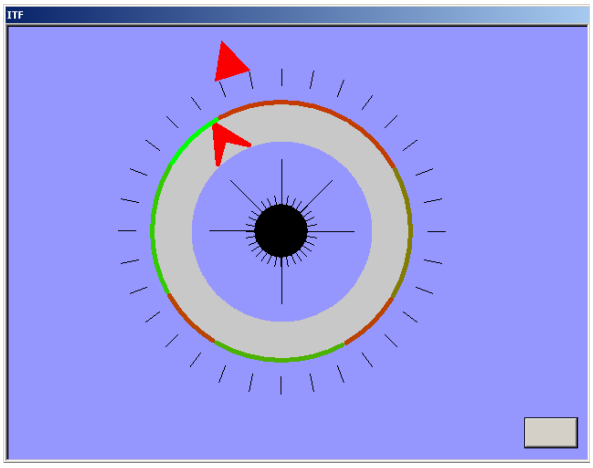


FIG. 3.5: Interface de commande de l'utilisateur final. La flèche intérieur désigne la direction sélectionnable par l'utilisateur et la flèche extérieur montre la direction de commande du fauteuil. Les couleurs sur le rond extérieur, du rouge au vert, montrent l'espace libre autour du fauteuil en fonction des données provenant des capteurs.

3.3.3.1 L'interface utilisateur

Nous disposons de deux interfaces utilisateurs. La première, visible dans la figure 3.5, est destinée à l'utilisateur final du fauteuil. Elle est composée d'une flèche tournant soit de manière continue sur 360°, soit selon 6 positions pré-définies. Elle se commande soit en tout ou rien, soit en analogique à l'aide d'un joystick.

La deuxième interface dont nous disposons est une interface dite de déverminage (figure 3.6) sur laquelle est affichée les valeurs des capteurs US, les résultats des calculs des murs, les vitesses linéaires et angulaires, 3 valeurs de déverminage, les positions (coordonnées X, Y et orientation θ) relatives et absolues du fauteuil dans son environnement ainsi que la distance et l'angle des murs présents sur les côtés du fauteuil. La position absolue est remise à 0 lors du démarrage du programme ou lors de l'appui sur le bouton initialisation de l'interface. La position relative est définie pour le mouvement en cours, c'est-à-dire qu'elle est remise à 0 à chaque nouvelle commande de direction entrée par l'utilisateur. Ceci veut également dire que nous n'avons aucune valeur fiable de position et d'orientation du fauteuil dans son environnement car la position initiale du fauteuil n'est pas définie. Ces valeurs servent uniquement comme information pour le déverminage et ne sont fiables que dans le simulateur, les valeurs réelles étant soumises au glissement des roues.

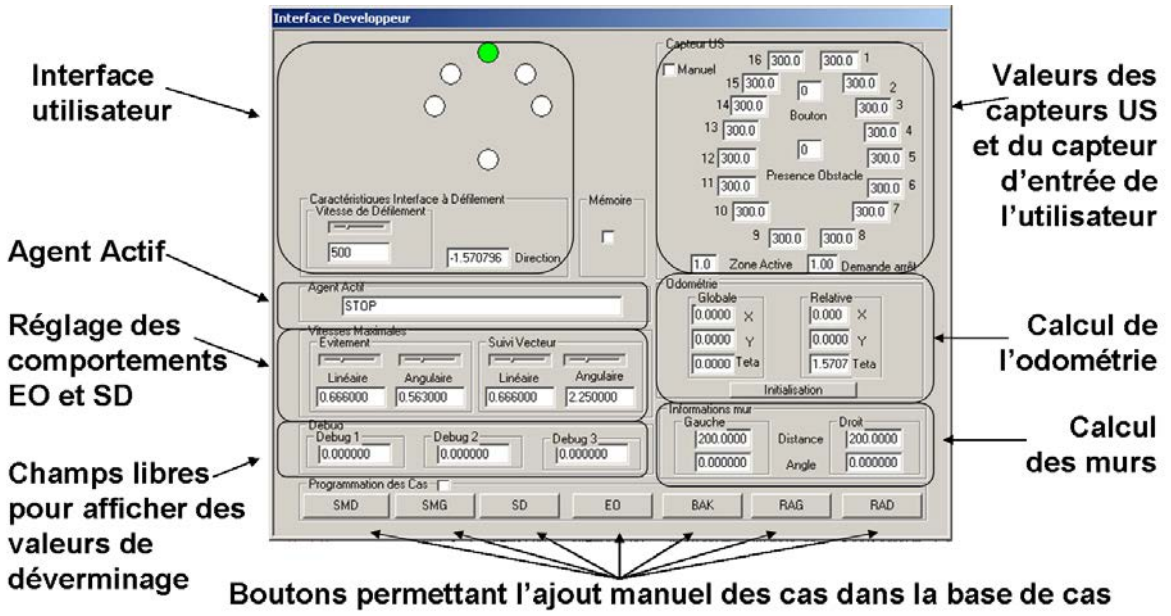


FIG. 3.6: Interface de déverminage avec : EO = Évitement d'obstacles, SMD = suivi de mur droit, SMG = suivi de mur gauche, BAK = Backtraking (retour sur ses pas) et SD = suivi de direction.

3.3.3.2 L'agent Capteur US

L'agent de lecture des capteurs US communique par liaison série avec le micro-contrôleur qui gère la lecture des capteurs US et de l'entrée TOR de l'utilisateur. Il sert de pont entre la liaison série et la mémoire partagée.

3.3.3.3 L'agent Moteur

L'agent moteur inclut toute la gestion des moteurs. Il est composé de deux parties distinctes la première étant la génération des commandes des moteurs et la seconde partie gère l'odométrie. La commande des moteurs est en fait réalisée en remplaçant les tensions du joystick dans le boîtier de contrôle du fauteuil. Les tensions générées sont les consignes avant-arrière et droite-gauche du joystick. Elles sont calculées à partir des commandes de vitesse linéaire et angulaire qui sont définies par l'agent comportement ayant le cas le plus probable.

La seconde partie de cet agent gère l'odométrie qui calcule les positions X, Y et de l'angle θ. Il est à noter qu'aucune gestion du glissement n'est effectuée, ces valeurs sont donc entachées d'erreur et ont une dérive non bornée.

Maintenant que nous avons vu tous les agents présents sur le VAHM nous

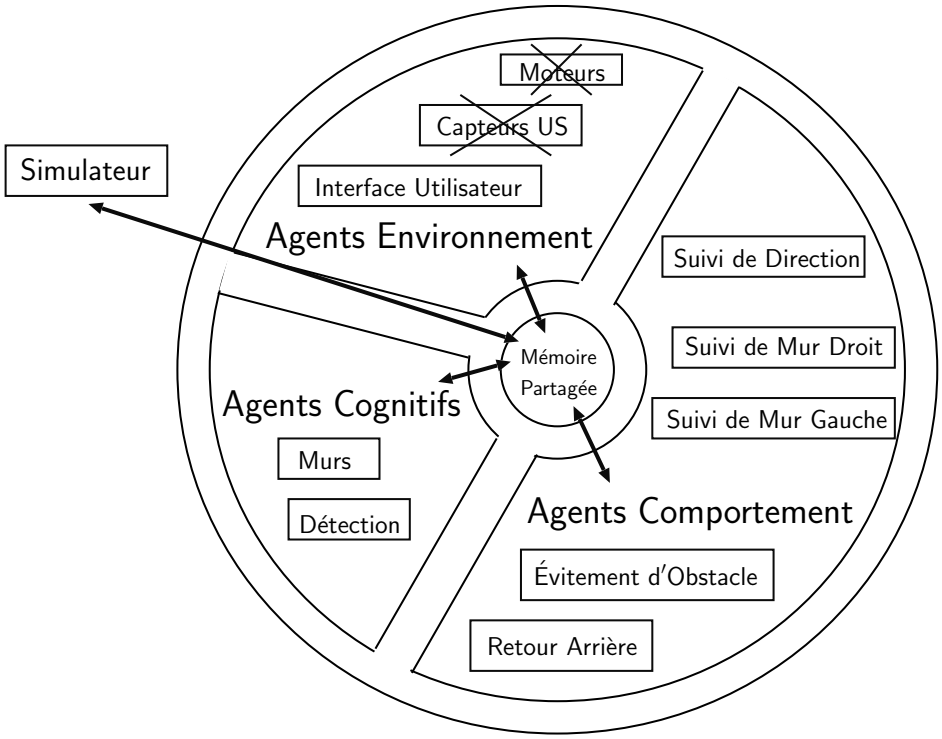


FIG. 3.7: La structure multi-agent avec le simulateur.

allons nous intéresser au simulateur qui peut être assimilé à un agent car il remplace les agents moteur et capteur US lors de son fonctionnement.

3.3.4 Le simulateur

Le simulateur peut être considéré comme un agent un peu particulier puisque celui-ci doit remplacer la partie physique du VAHM. Comme le montre la figure 3.7, lorsque le simulateur est exécuté, les agents capteurs et moteurs sont désactivés et c'est le simulateur qui se charge de mettre les valeurs des capteurs US virtuels ainsi que les valeurs de distance parcourue. Initialement la distance parcourue était donnée par l'agent moteur à partir des données lues sur les codeurs incrémentaux montés sur les roues. L'avantage de cette méthode est d'avoir les mêmes réactions du fauteuil en simulation qu'en réel car ce sont les mêmes agents comportement qui réagissent dans les deux cas.

Dans le simulateur, l'utilisateur peut se déplacer dans un environnent en 2D en vue du dessus à l'échelle 1 pixel/cm (voir figure 3.8). Mais le but du simulateur n'est pas d'avoir une représentation fidèle d'un environnement puisque le système VAHM ne prend pas en compte les positions X,Y ou l'angle θ pour l'aide à la planification de trajet. Cependant il permet de

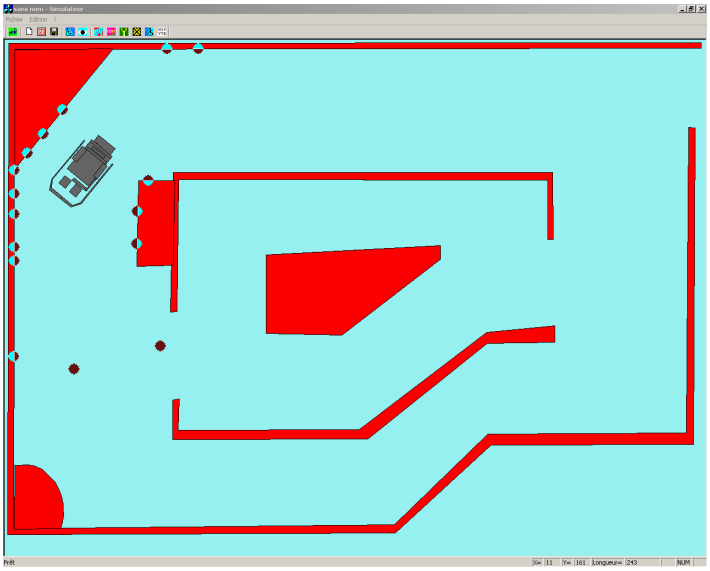


FIG. 3.8: Simulateur d'environnement accompagnant le système VAHM. Les points noir/cyan montrant les points de mesures des capteurs US simulés. Lorsque le point est complètement noir, cela veut dire qu'aucun obstacle n'est détecté. Les distances de détection maximales étant identiques à celles des capteurs présents sur les VAHM.

développer les agents en ayant des actions globales fidèles au fauteuil réel. La mesure des distances devant remplacer les données des capteurs US est effectuée par la méthode de rayon d'une largeur fixe consistant à mesurer la distance du premier obstacle, dans l'axe du capteur. Cette méthode ne représente pas fidèlement les données des capteurs car ces derniers ont un angle d'ouverture de 24° mais elle est suffisante dans notre cas. Le but n'est pas de réaliser un environnement réaliste pour réaliser des modèles utilisables en réel mais plutôt pour développer et déverminer les agents comportement. Il est à noter cependant que les cas appris dans le simulateur peuvent être utilisés pour une utilisation réelle. Pour le déplacement du fauteuil dans son environnement virtuel, les vitesses linéaires et angulaires sont simplement appliquées au fauteuil virtuel avec un coefficient. Le modèle du fauteuil est donc très simple et ne comporte aucune modélisation dynamique.

Lors de la simulation, le capteur TOR est remplacé par l'appui du bouton droit de la souris au dessus de la fenêtre de l'interface utilisateur.

3.4 Contexte de modélisation

Notre fauteuil se déplace dans des environnements construits à base de blocs modulables. Le calculateur embarqué permet d'enregistrer des données lors du parcours d'un déplacement. Les données enregistrées sont le comportement actif, la position et la rotation relative au point de départ du fauteuil, le rétrécissement avant, la distance du plus proche obstacle et les vitesses linéaires et angulaires du fauteuil. Ces données sont échantillonnées selon la distance parcourue par le fauteuil. Ainsi l'on s'affranchit des contraintes temporelles comme les arrêts et les parcours effectués à des vitesses différentes. Les valeurs maximales de lecture de distance des capteurs US du fauteuil étant de 3m, les mesures du rétrécissement ont donc une valeur maximale de 6m, et celle de la distance au plus proche obstacle est de 3m. La position et l'angle relatif (en radiant) n'ont pas de limites si ce n'est celles des variables qui les contiennent (type float = $\pm 3.4 \times 10^{38}$). Les comportements sont ceux présentés précédemment.

Une trajectoire est un ensemble d'échantillons représentant une petite partie du déplacement d'un fauteuil comme un arc de cercle où une ligne droite.

Un trajet est un ensemble de trajectoires, et donc d'échantillons, représentant une partie du déplacement du fauteuil pouvant être utilisée pour la modélisation. Il débute et s'arrête aux intersections entre les différents lieux parcourus par le fauteuil car les intersections peuvent amener à prendre différentes directions. La taille d'un trajet n'a pas d'importance.

Un chemin peut être vu comme l'ensemble des trajectoires, et donc des trajets, réalisés lors d'un déplacement du fauteuil pour aller d'un point A à un point B.

Les premiers tests de validation des algorithmes de modélisation et de reconnaissance ont été réalisés à partir de 23 trajets réels réalisés sur 4 environnements différents. Ces trajets, représentés dans la figure 3.9 avec leur environnement, ont été réalisés en utilisant la commande du VAHM basée sur un bouton TOR et de l'interface utilisateur présentés dans le paragraphe 3.3.3.1. L'ensemble est composé de 7 trajets de type 1, 6 de type 2, 3 de type 3 et 7 de type 4 qui comprennent entre 80 et 110 observations. La distance correspondante est de 8 à 11m étant donné que le système VAHM échantillonne tous les 10cm . Lors de ces trajets nous avons enregistré tous les paramètres donnés précédemment, ce qui permet d'effectuer des tests

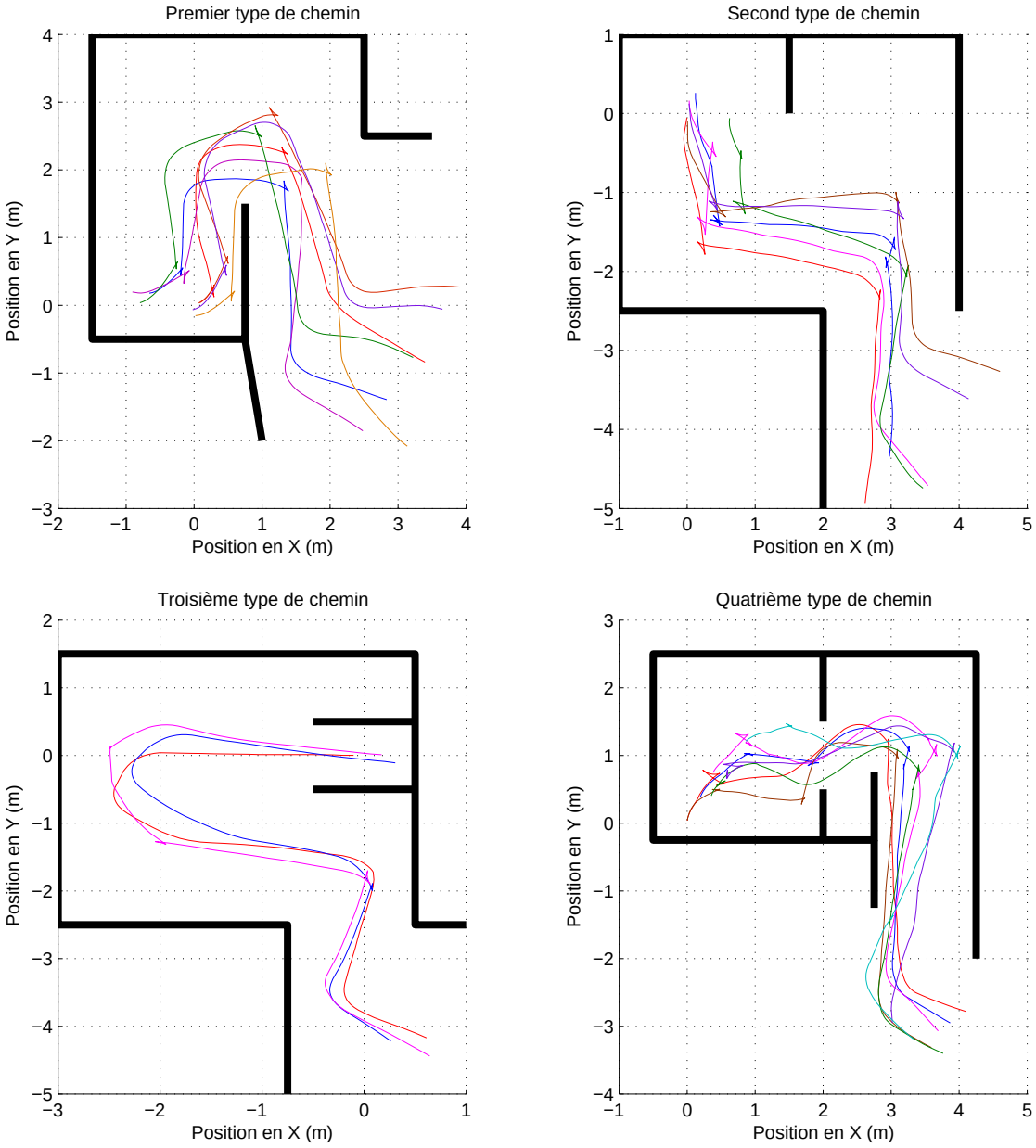


FIG. 3.9: Les quatre types de trajets réels effectués pour les tests de modélisation.

comparatifs avec plusieurs méthodes de modélisation et de reconnaissance.

Le but de tous les essais qui suivent est de dégager une méthode de localisation basée sur les trajets du fauteuil dans le but de pouvoir proposer une direction pertinente à l'utilisateur afin de lui éviter d'avoir à la définir lui-même. Cette méthode doit donc être performante et doit permettre de modéliser un environnement complet. De ce fait, les modèles doivent correspondre à des trajets de tailles différentes qui, mis ensemble, doivent représenter l'environnement dans son ensemble comme le montre la figure 3.10 par exemple.

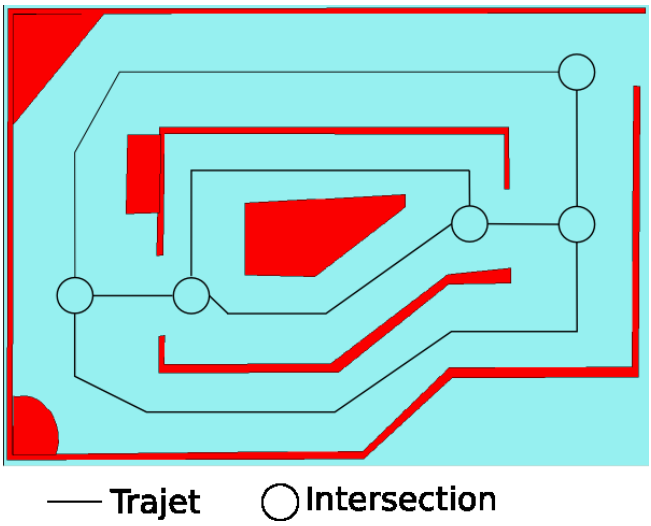


FIG. 3.10: Exemple d'environnement comprenant un ensemble de trajets qui sont séparés par leurs intersections.

3.5 Conclusion

L'objectif global est de pouvoir donner au VAHM la direction la plus probable que l'utilisateur va choisir et de l'appliquer afin d'éviter à l'utilisateur de devoir ré-émettre une commande de direction. Pour cela le VAHM doit connaître sa position dans l'ensemble de modèles représentant l'environnement.

On a vu que le système de commande est composé d'agents comportement qui régissent les réactions du fauteuil. L'idée est alors d'essayer de modéliser les trajets du fauteuil à partir uniquement de l'enchaînement de ces comportements qui ont été utilisés lors de la génération du trajet. Cet enchaînement pouvant être vu comme une suite de symbole.

4 Les MMC pour la localisation topologique

4.1 Introduction

CE tte partie se base sur une étude préliminaire [63] qui avait démontré qu'il était possible de reconnaître un trajet à partir de l'enchaînement des agents comportement qui le compose en utilisant des MMC (Modèles de Markov Cachés) [79].

La première partie de cette section présente les MMC et la seconde partie présente leur application sur les trajets réels présentés dans 3.4. Une étude sur la taille minimale de trajet à effectuer pour permettre la reconnaissance est ensuite présentée. Malheureusement les résultats ne furent pas à la hauteur de nos espérances. L'idée d'intégrer des données provenant des capteurs est alors parue intéressante. Pour cela nous avons utilisé les MMC-MD [17] qui permettent d'intégrer des vecteurs d'observations supplémentaires dans le MMC.

4.2 Présentation des modèles de Markov

Les chaînes de Markov, du nom de leur inventeur Andrei Andreevich Markov (Russie 1856-1922), font leur première apparition en 1906. Dans les années suivantes Markov tenta de modéliser l'alternance voyelles/consonnes dans des oeuvres littéraires russes à l'aide d'une chaîne de Markov à deux états. Leur popularité viendra quelques décennies plus tard grâce, notamment, aux travaux de Kolmogorov et Feller, pour la modélisation de phénomènes naturels.

La fin des années 60 voit apparaître une nouvelle approche statistique appelée Modèles de Markov Cachés (MMC) et c'est dans les années 1970, à l'initiative de Baker et Jelinek, qu'ils furent utilisés dans le domaine de la reconnaissance de la parole [70, 102], domaine dans lequel ils furent le

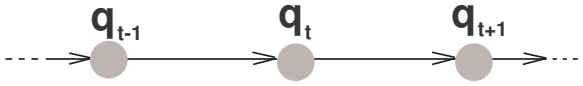


FIG. 4.1: Réseau Bayésien pour une chaîne de Markov : représente la dépendance conditionnelle entre les états d'une chaîne de Markov

plus utilisés. Plus récemment, les MMC furent également utilisés dans les domaines de la prédiction de texte [59], la reconnaissance de formes ou encore la localisation en robotique mobile [33, 88, 5].

4.2.1 Les chaînes de Markov

Une chaîne de Markov S modélise un système à temps discret pouvant prendre N états distincts tel que $S = \{s_1, s_2, \ldots, s_n\}$. Si on appelle q_t l'état dans lequel se trouve ce système à l'instant t, alors la probabilité qu'il se trouve dans l'état $q_t = s_i$ à l'instant t ne dépend que et seulement que de l'état $q_{t-1} = s_j$ dans lequel il se trouvait à l'instant $t-1$. Il ne tient donc pas compte des autres états par lesquels le système est passé (figure 4.1). Ceci est appelé proprité de Markov et se traduit par la relation

$$P[q_t = s_j | q_{t-1} = s_i, q_{t-2} = s_k, \ldots] = P[q_t = s_j | q_{t-1} = s_i],$$

qui est indépendante du temps.

On associe à ce système une matrice de transitions

$$\mathcal{A} = \begin{pmatrix} a_{11} & \ldots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \ldots & a_{NN} \end{pmatrix}$$

qui définit les probabilités de passage d'un état à un autre où

$$a_{ij} = P[q_t = s_j | q_{t-1} = s_i],\ 1 \leq i, j \leq N,\ \forall t > 1,$$

avec les contraintes

$$0 \leq a_{i,j},$$

$$\sum_{j=1}^{N} a_{ij} = 1.$$

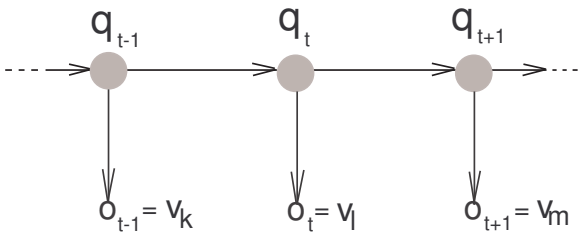


FIG. 4.2: Réseau Bayésien pour un modèle de Markov caché : représente la dépendance conditionnelle états-états et états-observations d'un tel modèle.

L'état initial est défini par le vecteur de probabilité initial $\pi = [\pi_1, \ldots, \pi_N]$ où

$$\pi_i = P[q_1 = s_i].$$

Une chaîne de Markov est ainsi définie par le doublet $(\pi, \mathcal{A})$.

4.2.2 Les modèles de Markov cachés (MMC)

Un MMC est une chaîne de Markov à laquelle on associe, pour chacun des N états un ensemble $V = \{v_1, v_2, \ldots, v_M\}$ de M observations comme le montre la figure 4.2. Les états sont supposés cachés et non directement observables, mais certains de leurs aspects « observables », v_k, sont détectés lorsqu'on se trouve dans l'un d'eux à un instant t. Ceci nous permet à chaque instant t d'être dans un état q_t et d'avoir une observation $o_t = v_k$. La transition d'un état à un autre est gouvernée par une matrice de transitions $\mathcal{A}(N \times N)$ semblable à celle des chaînes de Markov. L'information observée dans chaque état est déterminée par une matrice d'observation $\mathcal{B}(N \times M)$. Une telle extension offre l'avantage de pouvoir modéliser un système dont les probabilités d'occurrences des sorties (observations) dépendent d'un ensemble de « contextes ». Ces « contextes » sont représentés par les N états du modèle. Il est nécessaire d'insister sur le fait que seules les observations sont physiquement accessibles, alors que l'accessibilité aux états dépend de l'approche de modélisation, ce qui lui vaut l'appellation de Modèles de Markov Caché**.**

On considère qu'à chaque instant t, un état $q_t = s_j$ émet une observation $o_t = v_k$ qui ne dépend que de cet état. On définit ainsi la matrice d'observa-

tion

$$\mathcal{B} = \begin{pmatrix} b_{11} & \cdots & b_{1M} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NM} \end{pmatrix}$$

par

$$b_{jk} = b_j(k) = P[o_t = v_k | q_t = s_j],\ 1 \leq j \leq N,\ 1 \leq k \leq M,\ \forall t \geq 0,$$

$$\sum_{k=1}^{M} b_{jk} = 1,\ 1 \leq j \leq N.$$

Un MMC est ainsi défini par le triplet $\lambda = (\pi, \mathcal{A}, \mathcal{B})$.

4.2.2.1 Algorithmes de recherche

L'unique accès aux observations des systèmes modélisés par des MMC suscite deux types de questions.

La première est, étant donné un modèle λ, quelle est la probabilité que ce modèle ait généré une séquence d'observation $O = \{o_1, o_2, \ldots, o_T\}$? C'est-à-dire comment calculer la probabilité $P(O|\lambda)$?

La seconde question qui se pose est, étant donné un système qui a émis une séquence d'observation O et connaissant son modèle de Markov caché λ, quelle est la séquence d'état $Q = \{q_1, q_2, \ \ldots, q_T\}$ correspondante ? En d'autres termes, quelle est la séquence Q qui maximise la probabilité $P(Q|O, \lambda)$.

Il existe plusieurs algorithmes qui effectuent ce calcul. Nous présenterons les deux algorithmes principalement utilisés que sont l'algorithme Baum Welch et l'algorithme de Viterbi.

Algorithme Baum Welch Cette procédure permet de calculer la probabilité $P(O|\lambda)$ d'avoir une séquence d'observations O en considérant tous les chemins d'états qui peuvent la produire étant donné un MMC λ, et donc de répondre à la première question. Ce calcul peut s'effectuer suivant deux sens différents

forward :

Soit la variable « *forward* » $\alpha_t(i) = P(\{o_1, o_2, \ldots, o_t\}, q_t = s_i|\lambda)$ représentant la probabilité qu'un modèle λ ait pu générer la séquence d'observations $\{o_1, o_2, \ldots, o_t\}$ et d'être dans l'état s_i à l'instant t. L'algorithme *forward* est donné dans l'encadré 1 et est schématisé dans la figure 4.3.

backward :

Soit la variable « *backward* » $\beta_t(i) = P(\{o_{t+1}, o_{t+2}, \ldots, o_T\}|q_t = s_i|\lambda)$ repré-

Algorithm 1 Algorithme *forward*

Initialisation :
Pour $i = 1 \text{ a } N$,

$$\alpha_1(i) = \pi_i b_i(o_1)$$

Itération :
Pour $t = 1 \text{ a } T - 1$,
Pour $j = 1 \text{ a } N$,

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1})$$

Arrêt :

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

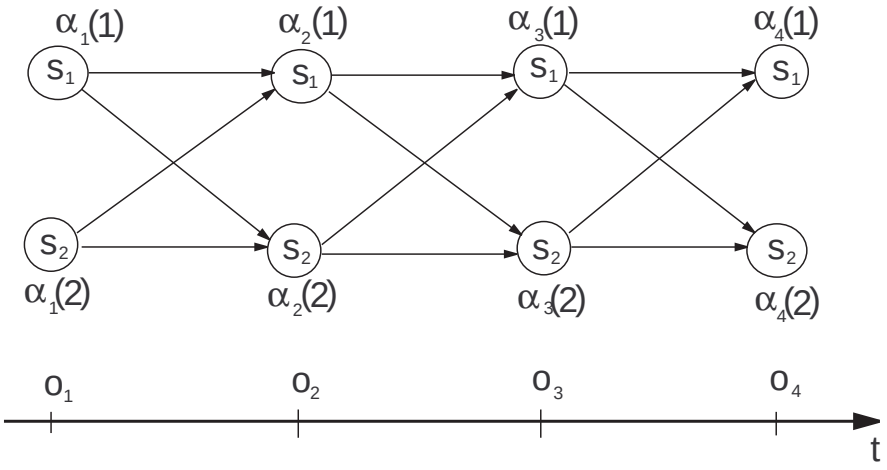


FIG. 4.3: Évolution temporelle de l'algorithme « *forward* » pour un MMC à $N = 2$ états cachés : le calcul des $\alpha_t(1)$ et $\alpha_t(2)$ s'effectue suivant le sens temporel croissant. Le calcul de la probabilité $P(\{o_1, o_2, o_3, o_4\}|\lambda)$ tient compte de tous les chemins d'états possibles, à savoir dans ce cas $2^4 = 16$ chemins possibles à travers les 2 états cachés afin de réaliser la séquence $O = \{o_1, o_2, o_3, o_4\}$. La valeur de la probabilité $P(O|\lambda)$est obtenue par l'addition $\alpha_4(1) + \alpha_4(2)$.

sentant la probabilité qu'un modèle λ ait pu générer la tranche de séquence $\{o_{t+1}, o_{t+2}, \ldots, o_T\}$sachant que ce modèle se trouvait dans l'état s_i à l'instant t. L'algorithme *backward* est donné dans l'encadré 2 et est schématisé dans la figure 4.4.

Algorithm 2 Algorithme *backward*

Initialisation :
Pour $i = 1$ a N

$$\beta_T(i) = 1$$

Itération :
Pour $t = T - 1$ a 1
Pour $i = 1$ a N

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Arrêt :

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)$$

Ces deux algorithmes ont des temps de calculs d'environ $N^2T + N$ opérations, $N(N-1)(t-1) + N - 1$ additions et $N(N+1)(t-1) + N$ multiplications.

Algorithme Viterbi L'algorithme de *Viterbi* permet de trouver les meilleures séquences d'états $Q = \{q_1, q_2, \ldots, q_T\}$ qui correspondent à une séquence d'observations $O = \{o_1, o_2, \ldots, o_T\}$, ce qui correspond à la deuxième question posée dans 4.2.2.1. Cet algorithme maximise la probabilité conjointe $P(O, Q|\lambda)$ ce qui est équivalent à maximiser $P(Q|O, \lambda)$ car $P(O, Q|\lambda) = P(Q|O, \lambda) \times P(O|\lambda)$ où $P(O|\lambda)$ est indépendante de Q. La transition retenue d'un état à un autre est celle qui est la plus probable par rapport à la probabilité de transition et la probabilité d'observation, ce qui nous amène à redéfinir la quantité δ_t,

$$\delta_t(i) = \max_{q1,q2,\ldots,q_t=s_i} P[\{q_1, q_2, \ldots, q_t\}, \{o_1, o_2, \ldots, o_t\}|\lambda],$$

représentant la meilleure probabilité, le long d'un chemin d'états qui se termine à l'instant t dans l'état s_i, ayant produit la séquence $\{o_1, o_2 \ldots, o_t\}$ de longueur t. Ce qui induit que

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(o_{t+1}). \tag{4.1}$$

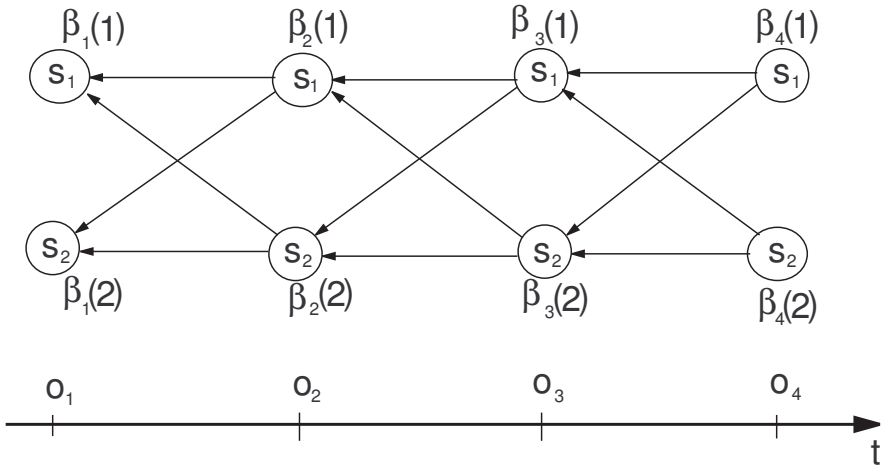


FIG. 4.4: Évolution temporelle de l'algorithme « *backward* » pour un MMC à $N = 2$ états cachés : le calcul des $\beta_t(1)$ et $\beta_t(2)$ s'effectue suivant le sens temporel décroissant. Le calcul de la probabilité $P(\{o_1, o_2, o_3, o_4\}|\lambda)$ tient compte de tous les chemins d'états possibles, à savoir dans ce cas $2^4 = 16$ chemins possibles à travers les 2 états cachés afin de réaliser la séquence $O = \{o_1, o_2, o_3, o_4\}$. La valeur de la probabilité $P(O|\lambda)$est obtenue par la formule $\pi_1 \times b_1(o_1) \times \beta_4(1) + \pi_2 \times b_2(o_1) \times \beta_4(2)$.

Pour pouvoir retrouver cette séquence d'états il faut garder une trace des valeurs qui ont maximisé 4.1 pour chaque t et j. Ce qui est fait dans le tableau $\psi_t(j)$. Les détails de cet algorithme sont donnés dans 3 et schématisés dans 4.5.

4.2.2.2 Estimation et apprentissage des paramètres du MMC

Un des critères utilisé pour l'estimation des paramètres $\{\pi, \mathcal{A}, \mathcal{B}\}$ est celui de l'optimisation du maximum de vraisemblance. Il consiste à maximiser, par rapport aux éléments de ce triplet, la fonction $P(O|\lambda)$ (ou plutôt $\log[P(O|\lambda)]$) qui est une fonction déterministe non linéaire. Des algorithmes comme celui du gradient du premier ou du second ordre peuvent être utilisés pour la résoudre, les deux procédures proposées dans [81, 70] permettent aussi de trouver, à partir d'un modèle initial λ, un nouveau modèle $\hat{\lambda}$ qui vérifie $P(O|\hat{\lambda}) \geq P(O|\lambda)$. Ces procédures sont les suivantes :

1. Procédure de *Baum-Welch* (« *forward-backward* »)
   Cette procédure utilise les deux variables α et β et estime les para-

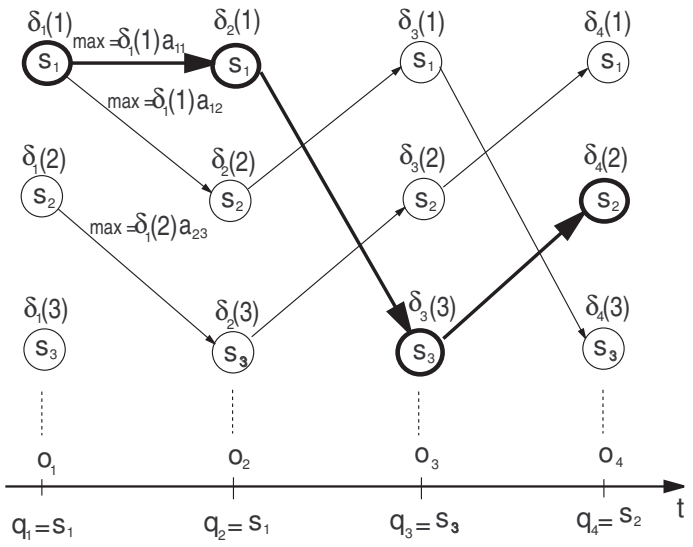


FIG. 4.5: Évolution de l'algorithme de *Viterbi* pour un MMC à $N = 3$ états cachés : à l'initialisation $t = 1$ on a $\delta_1(i) = P(q_1 = S_i, o_1|\lambda)$ et $\psi_1(i) = 0$ pour $1 \leq i \leq N$. A la première itération $t = 2$ on a $\max_{1 \leq i \leq 3}(\delta_1(i)a_{ij}) = [(\delta_1(1)a_{11}), (\delta_1(1)a_{12}), (\delta_1(i)a_{23})]^T$ d'où $\delta_2 = [(\delta_1(1)a_{11}b_1(o_2)), (\delta_1(1)a_{12}b_2(o_2)), (\delta_1(2)a_{23}b_3(o_2))]^T$ et $\psi_2 = [1\,1\,2]^T$. Le même raisonnement est ainsi poursuivi jusqu'à l'instant $T = 4$. Après la fin de l'itération $T = 4$ l'élément maximum du vecteur δ_4 correspond à $\delta_4(2) = P(\{s_1, s_2, s_3, s_4\}, \{o_1, o_2, o_3, o_4\}|\lambda)$. Cette quantité de probabilité $P(Q, O|\lambda)$ d'avoir la séquence d'observation O, produite par la séquence d'états la plus probable. La reconstitution du chemin le plus probable se fait à l'aide de la matrice spacio-temporelle $\psi\,(T \times N)$ dans le sens temporel décroissant. Ainsi le chemin le plus probable est constitué par la séquence : $Q = \{q_1 = s_1, q_2 = s_1, q_3 = s_3, q_4 = s_2\}$.

Algorithm 3 Algorithme de recherche de *Viterbi*

Initialisation :
Pour $i = 1$ a N

$$\delta_1(i) = \pi_i b_i(o_1), \tag{4.2}$$
$$\psi_1(i) = 0$$

Itération :
Pour $t = 2$ a T
Pour $j = 1$ a N

$$\delta_t(j) = \max_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}]b_j(o_t), \tag{4.3}$$
$$\psi_t(j) = \arg\max_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}]$$

Arrêt :

$$P^* = \max_{1 \leq i \leq N}[\delta_T(i)], \tag{4.4}$$
$$q_t^* = \arg\max_{1 \leq i \leq N}[\delta_T(i)]$$

Constitution du chemin
Pour $t = T - 1$ a 1

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{4.5}$$

mètres du nouveau modèle de la manière suivante :

$$\begin{aligned} \hat{\pi} &= \text{Esp}[\text{nombre de fois dans l'etat } S_i \text{ a } t = 1] \\ &= \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_1(i)\beta_1(i)} \end{aligned}$$

$$\begin{aligned} \hat{a}_{ij} &= \frac{\text{Esp[nombre de transitions de } S_i \text{ à } S_j]}{\text{Esp[nombre de transitions à partir de } S_i} \\ &= \frac{\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \end{aligned}$$

$$\begin{aligned} \hat{b}_i(k) &= \frac{\text{Esp[nombre de fois dans } S_j \text{ et } V_k \text{ observé]}}{\text{Esp[nombre de fois dans l'état } S_j]} \\ &= \frac{\sum_{t=1\ o_t=V_k}^{T} \alpha_t(i)\beta_t(i)}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)} \end{aligned}$$

2. Procédure de *Viterbi* (Ségmentation des séquences d'observations)
   L'algorithme de Viterbi donne la séquence la plus probable correspondant à une séquence d'observations donnée. L'idée est d'utiliser les transitions entre les états et l'émission des différentes observations pour l'estimation des π, $\mathcal{A}$ et $\mathcal{B}$. Ainsi pour un modèle initial λ donné et une séquence d'observations $O = \{o_1, o_2, \ldots, o_T\}$, on applique l'algorithme de *Viterbi* pour déterminer la séquence d'états $Q = \{q_1, q_2, \ldots, q_T\}$ la plus probable comme le montre la figure 4.5. Ensuite on calcul le nombre de transitions d'un état S_j à un état S_i pour $1 \leq i \leq N$, $1 \leq j \leq N$

et le nombre des transitions à partir d'un état S_i ($1 \leq i \leq N$). On calcule aussi le nombre d'occurrence du symbole observable V_k ($1 \leq k \leq M$) dans chaque état S_i ($1 \leq i \leq N$). Enfin l'estimation des paramètres du nouveau modèle est donnée par :

$$\hat{a}_{ij} = \frac{\text{nombre de transitions de } S_i \text{ a } S_j}{\text{nombre de transitions a partir de } S_i}$$

$$\hat{b}_j(k) = \frac{\text{nombre de fois dans } S_j \text{ et } V_k \text{ observe}}{\text{nombre de fois dans l'etat } S_j}$$

Afin d'éviter d'obtenir des probabilités d'observations nulles, dans le cas où ces observations n'apparaissent pas dans les données d'apprentissage, on fixe tous les $b_j(k)$ ($1 \leq j \leq N$, $1 \leq k \leq M$) à un seuil minimum [79].

Les relations d'estimations établies ci-dessus doivent être implantées de manière efficace afin d'aboutir à des résultats satisfaisants avec des temps de convergence répondant aux exigences de l'application.

Ces processus d'itérations, qui sont des solutions à des critères d'optimisations non linéaires, donnent des résultats très sensibles aux conditions initiales.

4.2.3 Modélisation par MMC

Afin de tester les MMC comme outils de modélisation, nous avons utilisé les trajets, décrits dans le chapitre 3.4, dans lesquels il n'a été retenus que les séquences de comportements comme celles représentées dans la figure 4.6 dans laquelle nous avons laissé la position de la mesure pour une meilleure visualisation.

D'un point de vue général, la séquence de comportements peut être considérée comme une séquence aléatoire de nombres finis. La difficulté est double car on ne connaît pas où l'utilisateur veut aller ni comment le fauteuil va répondre à l'environnement. La réaction du fauteuil dépend du comportement actif et de petites variations dans le trajet peut faire changer le comportement actif. Cependant tous les trajets qu'effectuent le fauteuil sont des séquences de comportements qui peuvent être vues comme des séquences de symboles d'observations utilisables par les MMC.

Les premiers tests réalisés sont des essais de classification de trajets.

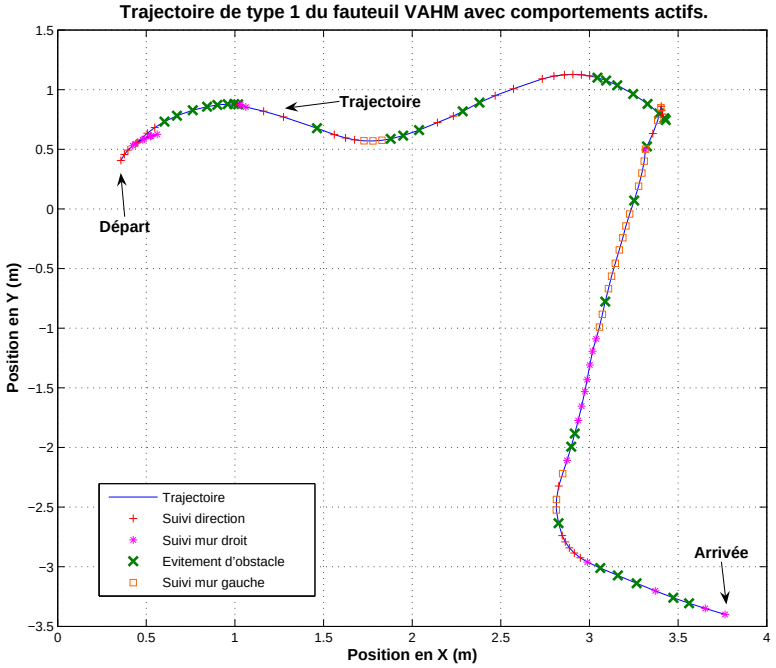


FIG. 4.6: Exemple de séquence de comportements (trajet de type 4). Cette figure représente également la position du comportement sur le trajet.

4.2.3.1 Classification des trajets

Les tests de classification consistent à construire des modèles de trajet à partir de tous les trajets des 4 types dont on dispose (cf §3.4). Ces 4 modèles sont alors utilisés pour vérifier s'il est possible de classifier les trajets appris.

Plusieurs phases d'apprentissage et d'essais sont nécessaires pour tester l'efficacité des MMC pour la modélisation de trajets. Ceci peut être résumé comme suit :

- apprentissage avec 3 types d'algorithmes : la procédure de Baum-Welch, la procédure de Viterbi avec une initialisation aléatoire et la procédure de Baum-Welch initialisés par Viterbi.
- le nombre d'états cachés est pris entre 3 et 30 pour en déterminer le nombre optimal d'états.

Après avoir fait apprendre à un MMC λ un type de trajet, tous les trajets (séquence de comportements O_i) sont présentés à ce MMC et nous calculons la probabilité $P(O_i|\lambda)$ que le modèle λ ait généré la séquence d'observations O_i. Dans le cas d'un processus de modélisation adéquat, les probabilités des trajets du type appris doivent être plus grandes que celles de autres trajets.

Les résultats pour les trajets de types 1 sont montrés dans la figure 4.7.

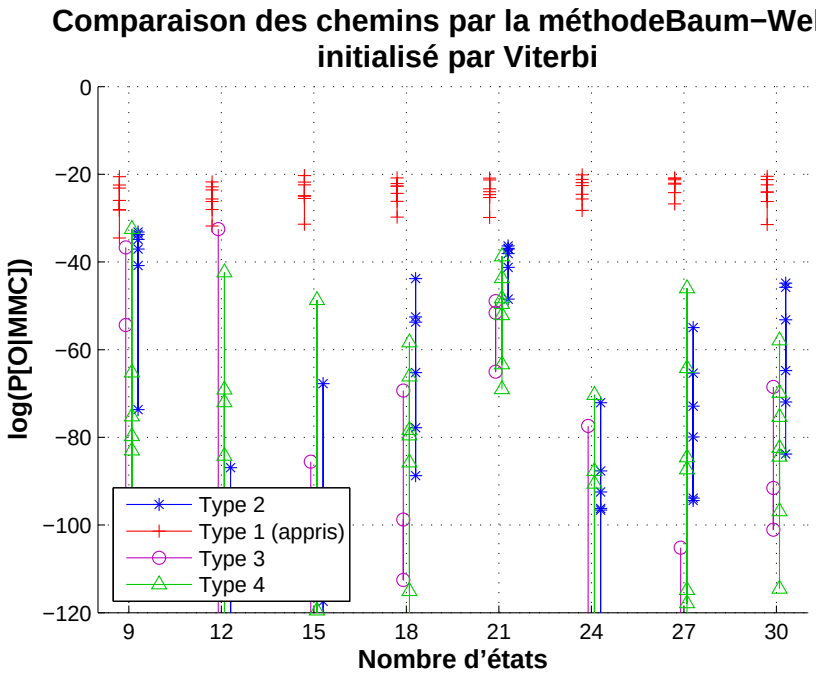


FIG. 4.7: Résultats d'apprentissage : les MMC peuvent reconnaître une classe de trajets appris par rapport à différents trajets présentés. Les probabilités sont présentées sur une échelle logarithmique afin de mieux visualiser les répartitions des résultats.

Globalement les trajets peuvent être identifiés à l'aides des MMC en se basant sur leur plus forte probabilité avec plus de 12 états cachés en utilisant la procédure de Baum-Welch.

4.2.3.2 Essais de généralisation

Les essais de généralisation servent à savoir si les MMC sont capables de généraliser le modèle à des trajets non appris. Ils se déroulent de la manière suivante. Dans un premier temps, les modèles sont appris à partir des 4 premiers trajets de chaque type. Une fois ces trajets appris on effectue un test de reconnaissance sur l'ensemble des trajets.

Les résultats des tests de généralisation, présentés dans la figure 4.8, montrent que pour les trajets appris, la classification fonctionne. Cependant, lorsqu'un trajet provenant d'une classe apprise mais non utilisé dans le processus d'apprentissage est passé dans le MMC, la classification ne fonctionne pas même si la probabilité pour ces trajets non appris reste relativement élevée.

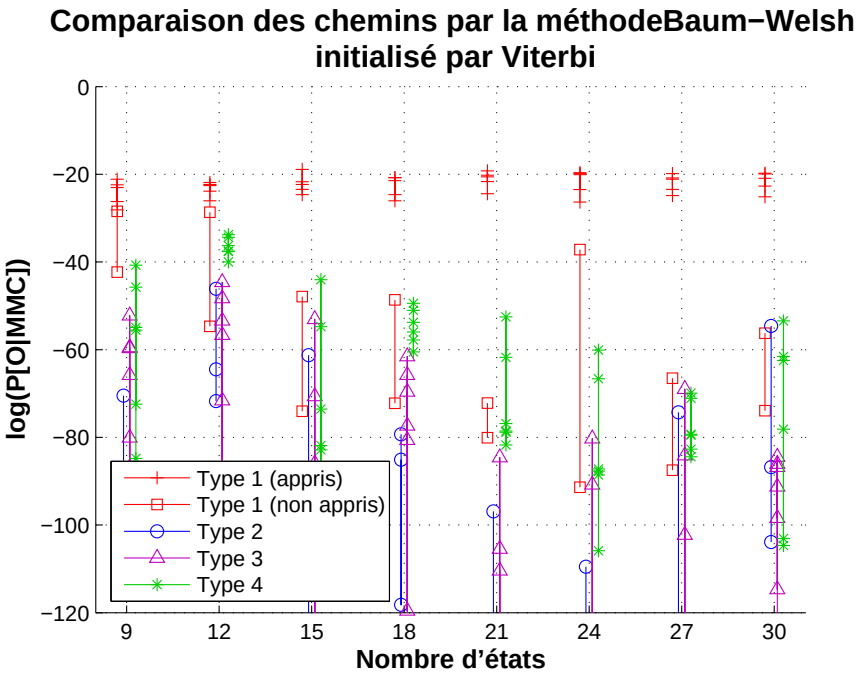


FIG. 4.8: Résultats de généralisation : la classification de trajets non appris, bien que gardant une probabilité haute, n'obtient pas de résultats probants.

4.2.3.3 Taille minimale d'un trajet

Nous avons également réalisé une autre série de tests sur les MMC dans le but de connaître la taille minimale d'un trajet reconnaissable par les MMC. Ceci doit nous permettre de savoir à partir de quelle distance parcourue par le fauteuil il est possible de réaliser une reconnaissance. Cette série de tests est réalisée en réutilisant les modèles précédemment appris et en utilisant la procédure de Baum-Welch initialisés par Viterbi. Chaque modèle, allant de 3 à 30 états cachés pour les 4 types de trajets, est testé avec une fenêtre glissante allant de 8 à 76 observations. Les résultats, présentés dans le graphique 4.9, ont permis de montrer qu'il fallait plus de 76 observations pour réaliser une reconnaissance correcte soit quasiment la totalité des trajets.

4.2.3.4 Conclusion sur la modélisation par MMC

Ces premiers résultats de classification montrent que les MMC peuvent être utilisés pour la modélisation et la classification de trajets à partir de séquences de comportements. Hélas, la taille de la séquence nécessaire à la classification est grande, ce qui peut être problématique car c'est cette taille qui déterminera la longueur des séquences d'observations utilisées pour l'apprentissage et la classification. Dans notre cas le fauteuil devra

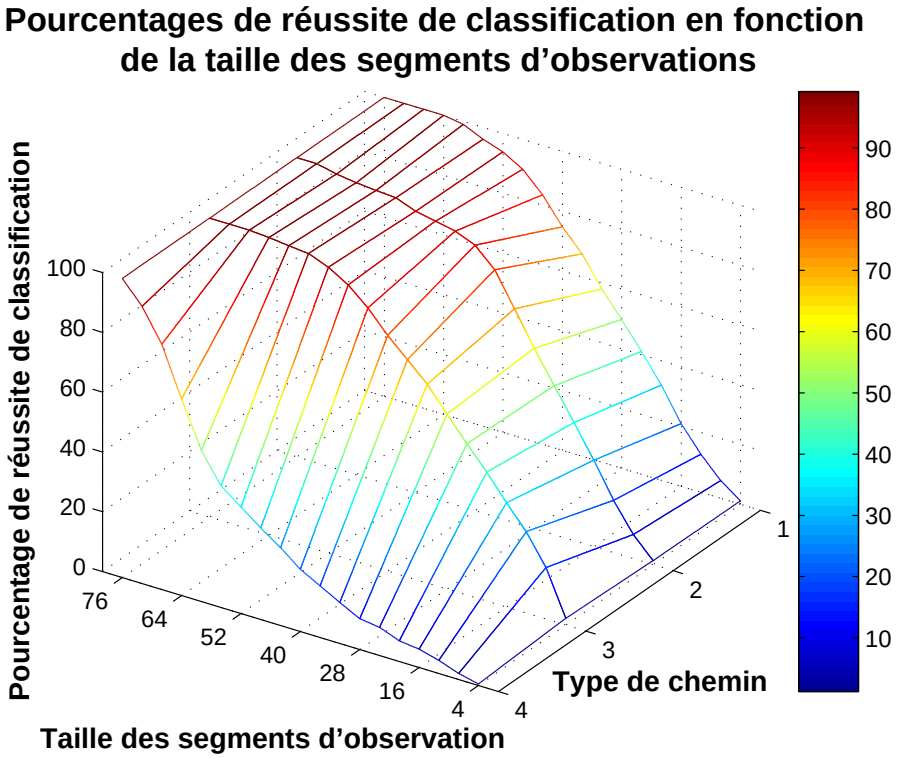


FIG. 4.9: Résultats des tests de détermination de la taille minimale de la fenêtre d'observation à considérer pour réaliser la classification.

parcourir plus de 7.6m avant de réussir à classifier un trajet.

Afin de diminuer cette taille minimale et d'améliorer la qualité de la classification, une augmentation de la « quantité d'information » contenue dans les modèles est nécessaire. Cette information peut, entre autre, être trouvée dans les valeurs lues par les capteurs ultra-sons présents sur le VAHM. Cependant pour ajouter des informations dans les modèles il faut que les algorithmes utilisés puissent les intégrer.

4.3 Intégration de vecteurs d'observations supplémentaires dans les MMC

Dans les MMC standard, le vecteur d'observation O correspond à une seule ligne entrée du système. Dans le cas où le système possède plusieurs lignes entrées, il serait intéressant de pouvoir les inclure dans le modèle. Les observations ne sont donc plus intégrées dans un vecteur mais dans une matrice $O = [O^1, O^2, \ldots, O^K]^T$ avec $O^k = [o_1^k, o_2^k, \ldots, o_T^k]$. De plus les symboles d'observations M_k ne sont pas semblables en fonction des lignes d'entrées du système. Le système d'observations se présente donc comme suit :

$$O = \begin{cases} o_1^1 \quad o_2^1 \quad o_3^1 \quad \cdots \quad o_T^1 & \text{avec} \quad o_i^1 \in [1\, M_1] \\ o_1^2 \quad o_2^2 \quad o_3^2 \quad \cdots \quad o_T^2 & \text{avec} \quad o_i^2 \in [1\, M_2] \\ \vdots \\ o_1^K \quad o_2^K \quad o_3^K \quad \cdots \quad o_T^K & \text{avec} \quad o_i^K \in [1\, M_K] \end{cases}$$
$$\text{avec } i \in [1\, T]$$
$$M = \begin{cases} 0 \quad 1 \quad 2 \quad \cdots \quad \cdots \quad M_1 \\ 0 \quad 1 \quad 2 \quad \cdots \quad M_2 \\ \vdots \\ 0 \quad 1 \quad 2 \quad \cdots \quad \cdots \quad \cdots \quad M_K \end{cases}$$

L'inclusion de ces différentes séquences d'observations dans un MMC peut être réalisée de différentes manières. La première consiste à combiner les K vecteurs en un seul vecteur en ligne. La seconde utilise les MMC-MD [101, 17] et demande une modification de la matrice d'observations $\mathcal{B}$.

4.3.1 Vectorisation des observations

On vectorise les observations de la manière suivante :

$$o'(t) = \sum_{k=1}^{K} (o^k(t). \prod_{i=1}^{k-1} M(i))$$

$$M' = \prod_{k=1}^{K} M(k)$$

Par exemple la matrice O suivante de longueur $T = 4$ avec 3 lignes d'entrées

$$O = \begin{bmatrix} 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

avec

$$M = \begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix}$$

donne une séquence d'observations "vectorisée"

$$O' = [14\, 15\, 7\, 23]$$

avec

$$M' = 24$$

Dans cette solution, l'algorithme d'apprentissage doit apprendre un grand ensemble de symboles d'observations en utilisant un MMC standard. De ce fait, si quelques valeurs du vecteur d'observation V' n'apparaissent jamais dans les séquences d'apprentissage O', la reconnaissance risque alors de ne pas être efficace car il subsistera des probabilités d'observations nulles. Cependant cette méthode ne demande aucune modification des algorithmes de recherche.

4.3.2 Les MMC Multidimensionnels

Les MMC standard utilisent, à chaque instant t, un seul symbole d'observation. Cependant dans notre cas nous désirons utiliser à chaque instant t un vecteur de K observations. Dans ce cas, nous utilisons une extension des MMC appelée MMC-MD [18] qui permet d'intégrer plusieurs sources d'observations dans un MMC. Pour cela, on a besoin d'avoir une matrice de probabilités d'observations par vecteur d'observations. La matrice de probabilités d'observations $\mathcal{B}$ est alors composée de K matrices. Ce qui donne $\mathcal{B} = [\mathcal{B}^1, \mathcal{B}^2, \ldots, \mathcal{B}^K]$ où $\mathcal{B}^i$ est de taille $[N \times M(i)]$. Les matrices des MMC multidimensionnels sont :

$$\lambda = [\mathcal{A}, \mathcal{B}, \pi]$$

$$\mathcal{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,N} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \ldots & a_{N,N} \end{bmatrix}$$

$$\mathcal{B} = \begin{bmatrix} \begin{bmatrix} b^1_{1,1} & b^1_{1,2} & \ldots & b^1_{1,M^1} \\ b^1_{2,1} & b^1_{2,2} & \ldots & b^1_{2,M^1} \\ \vdots & \vdots & \ddots & \vdots \\ b^1_{N,1} & b^1_{N,2} & \ldots & b^1_{N,M^1} \end{bmatrix} & \begin{bmatrix} b^2_{1,M^2} \\ b^2_{2,M^2} \\ \vdots \\ b^2_{N,M^2} \end{bmatrix} \\ \begin{bmatrix} b^2_{1,1} & b^2_{1,2} & \ldots & \ldots & b^2_{1,M^2} \end{bmatrix} & \begin{bmatrix} b^K_{1,M^K} \\ b^K_{2,M^K} \\ \vdots \\ b^K_{N,M^K} \end{bmatrix} \\ \begin{bmatrix} b^3_{1,1} & \ldots & b^3_{1.M^3} \end{bmatrix} & \begin{bmatrix} b^K_{1,1} & b^K_{1,2} & \ldots & b^K_{1,M^K} \\ b^K_{2,1} & b^K_{2,2} & \ldots & b^K_{2,M^K} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \end{bmatrix}$$

$$\pi = [\pi_1 \, \pi_2 \, \ldots \, \pi_N]$$

« Cette solution produit des résultats similaires à ceux obtenus avec les MMC, à la différence que l'on introduit des produits de quantités $b^k_j(.)$ à

la place des quantités $b_j(.)$. Cela traduit la nécessité pour le MMC-MD de générer simultanément les différents *processii* associés à chacune de ses dimensions. » [18] La probabilité globale doit être calculée en utilisant des algorithmes modifiés de *Viterbi* 4 et de *Baum-Welch* 5 dans lesquels les quantités $b_j(.)$ sont substituées par des produits $b_j(O^t) = \prod_{k=1}^{K} b_j^k(O^t)$.

Algorithm 4 Algorithme de *Viterbi* pour les MMC-MD.

– Algorithme de recherche de *Viterbi*
Initialisation :
$1 \leq i \leq N$

$$\delta_1(i) = \pi_i \prod_{k=1}^{K} b_i^k(o_1^k)$$

Itération
$2 \leq t \leq T$
$1 \leq j \leq N$

$$\delta_t(j) = \max_{1 \leq 1 \leq N} [\delta_{t-1}(i).a_{ij}] \prod_{k=1}^{K} b_j^k(o_t^k)$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i).a_{ij}]$$

Stop

$$\rho^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

Reconstruction du chemin
$T - 1 \geq t \geq 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

– Estimation des paramètres du MMC
Estimation de la matrice de transitions $\mathcal{A}$
$1 \leq i \leq N$
$1 \leq j \leq N$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T}(q(t) = S_i, q(t+1) = S_j)}{\sum_{t=1}^{T} q(t) = S_i}$$

Estimation de la matrice de probabilité d'observation $\mathcal{B}$
$1 \leq j \leq N$
$1 \leq m \leq M$
$1 \leq k \leq K$

$$\hat{b}_j^k(m) = \frac{\sum_{t=1}^{T}(q(t) = S_j, o^k(t) = V_m)}{\sum_{t=1}^{T}(q(t) = S_j)}$$

Algorithm 5 Algorithme de *Baum-Welch* pour les MMC-MD.

– « *forward* »
Initialisation
$1 \leq i \leq N$

$$\alpha_1(i) = \pi_i \prod_{k=1}^{K} b_i^k(o_i^k)$$

Itération
$1 \leq t \leq T-1$
$1 \leq j \leq N$

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] \prod_{k=1}^{K} b_i^k(o_{t+1}^k)$$

Stop

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

– « *backward* »
Initialisation
$1 \leq i \leq N$

$$\beta_T(i) = 1$$

Itération
$T-1 \geq t \geq 1$
$1 \leq i \leq N$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \beta_{t+1}(j) \prod_{k=1}^{K} b_j^k(o_{t+1}^k)$$

Stop

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i) \prod_{k=1}^{K} b_i^k(o_1^k)$$

– Estimation des paramètres du modèle
Estimation de la matrice initiale π
$1 \leq i \leq N$

$$\pi_i = \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{N} \alpha_1(j)\beta_1(j)}$$

Estimation de la matrice de transitions $\mathcal{A}$
$1 \leq i \leq N$
$1 \leq j \leq N$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} \beta_{t+1}(j) \prod_{k=1}^{K} b_j^k(o_{t+1}^k)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

Estimation de la matrice d'observations $\mathcal{B}$
$1 \leq j \leq N$
$1 \leq m \leq M$
$1 \leq k \leq K$

$$\hat{b}_j^k(m) = \frac{\sum_{t=1_{o_t^k = v_m}}^{T} \alpha_t(j)\beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)}$$

Taille ρ en cm	Symbole d'observation
$\rho \leq 40$	0
$40 < \rho \leq 100$	1
$100 < \rho \leq 200$	2
$200 < \rho$	3

Angle θ en °	Symbole d'observation
$\theta < -10$	0
$-10 \leq \theta < -2$	1
$-2 \leq \theta \leq 2$	2
$2 < \theta \leq 10$	3
$10 < \theta$	4

TAB. 4.1: Classes de rétrécissement et vitesse angulaire.

4.4 Modélisation par MMC avec plusieurs sources d'observations

Les résultats précédents ont prouvés que l'utilisation des modèles de Markov avec la séquence de comportement comme valeurs d'observations ne donne pas de résultats probants car il ne leur est pas possible de généraliser le modèle à des trajets non appris. Cependant la séquence de comportements n'est pas la seule séquence d'observations disponible dans le système VAHM. Les valeurs des capteurs ultra-son ainsi que les données odométriques sont disponibles dans la mémoire commune (cf figure 3.3). L'idée est alors d'ajouter ces informations dans les modèles de Markov. Nous avons trouvé deux solutions pour intégrer ces données dans les MMC : la vectorisation des matrices d'observation ou l'utilisation des MMC-MD.

Les informations ou vecteurs d'observations choisis sont visibles dans la figure 4.10. Ces vecteurs sont les rétrécissement avant ρ, composé de la somme des distances des deux capteurs à l'avant axés sur les côtés opposés, et la vitesse angulaire θ calculée comme étant la différence angulaire entre deux échantillons du VAHM. Ces données ont été choisies car elle étaient présentes dans les relevés des trajets utilisés lors des tests précédants. Nous avons divisé ces vecteurs en petites classes pour pouvoir les intégrer dans les MMC-MD. Ces classes, données dans le tableau 4.1, ont été réalisées à la main à partir d'une observation globale des trajets. Le but de ces classes était de réaliser un premier test de validation de l'algorithme avant de passer éventuellement à un système de classes générées avec des méthodes évoluées ou encore d'utiliser des modèles de Markov continus [80].

Les tests réalisés, comme les précédents, ont pour but de tester l'efficacité de l'algorithme, en terme de performances et de temps de calculs, pour la reconnaissance de trajets. Nous avons testé les MMC multidimensionnels avec plusieurs états cachés et plusieurs combinaisons de vecteurs

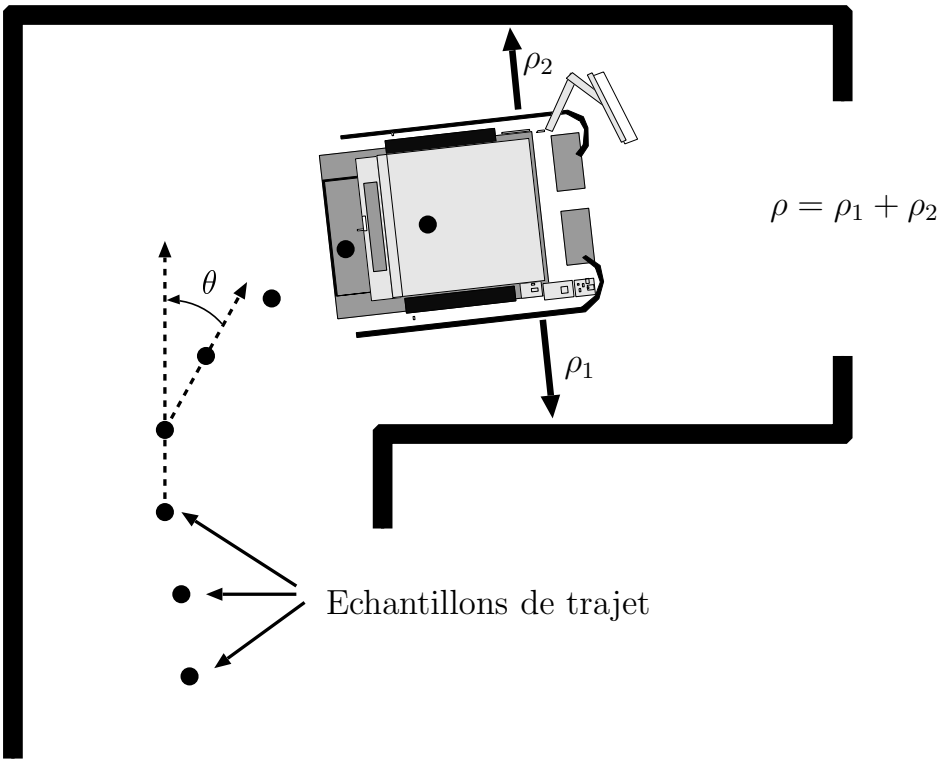


FIG. 4.10: Vecteurs d'observations utilisés pour les tests des MMC multidimensionnels.

d'observations. Cependant ces tests se divisent en deux catégories selon la méthode employée pour intégrer les vecteurs d'observations qui sont les tests avec vectorisation des observations et ceux utilisant l'algorithme des MMC-MD. Nous avons réalisé ces tests à partir des trajets précédemment décrits dans 3.4. Ceci nous permet de réaliser une comparaison directe des performances des méthodes ainsi que leurs avantages et inconvénients.

4.4.1 Tests de modélisation

Les tests de modélisation sont réalisés en utilisant de 6 à 12 états cachés et en utilisant les vecteurs d'observations choisis qui sont les comportements, la vitesse angulaire et le rétrécissement. Dans un premier temps les trois vecteurs sont appris séparément puis deux par deux et finalement les trois ensembles, ceci afin de mettre en évidence les contributions des différents vecteurs d'observations dans les résultats. Ces tests sont réalisés en utilisant les méthodes d'apprentissage de Baum-Welch, Viterbi et Baum-Welch initialisé par Viterbi.

4.4.1.1 Modélisation avec vectorisation des observations

Sur tous les résultats des essais de modélisation en vectorisant les matrices d'observations, seuls ceux utilisant l'algorithme d'apprentissage de Baum-Welch, qui sont présentés dans la figure 4.4.1.1, donnent des résultats corrects. On peut ainsi voir qu'il est possible d'apprendre et de classifier les données apprises en utilisant 12 états dans le MMC mais uniquement en utilisant les données provenant des capteurs, c'est-à-dire sans utiliser les comportements comme vecteur d'observation.

4.4.1.2 Modélisation avec les MMC-MD

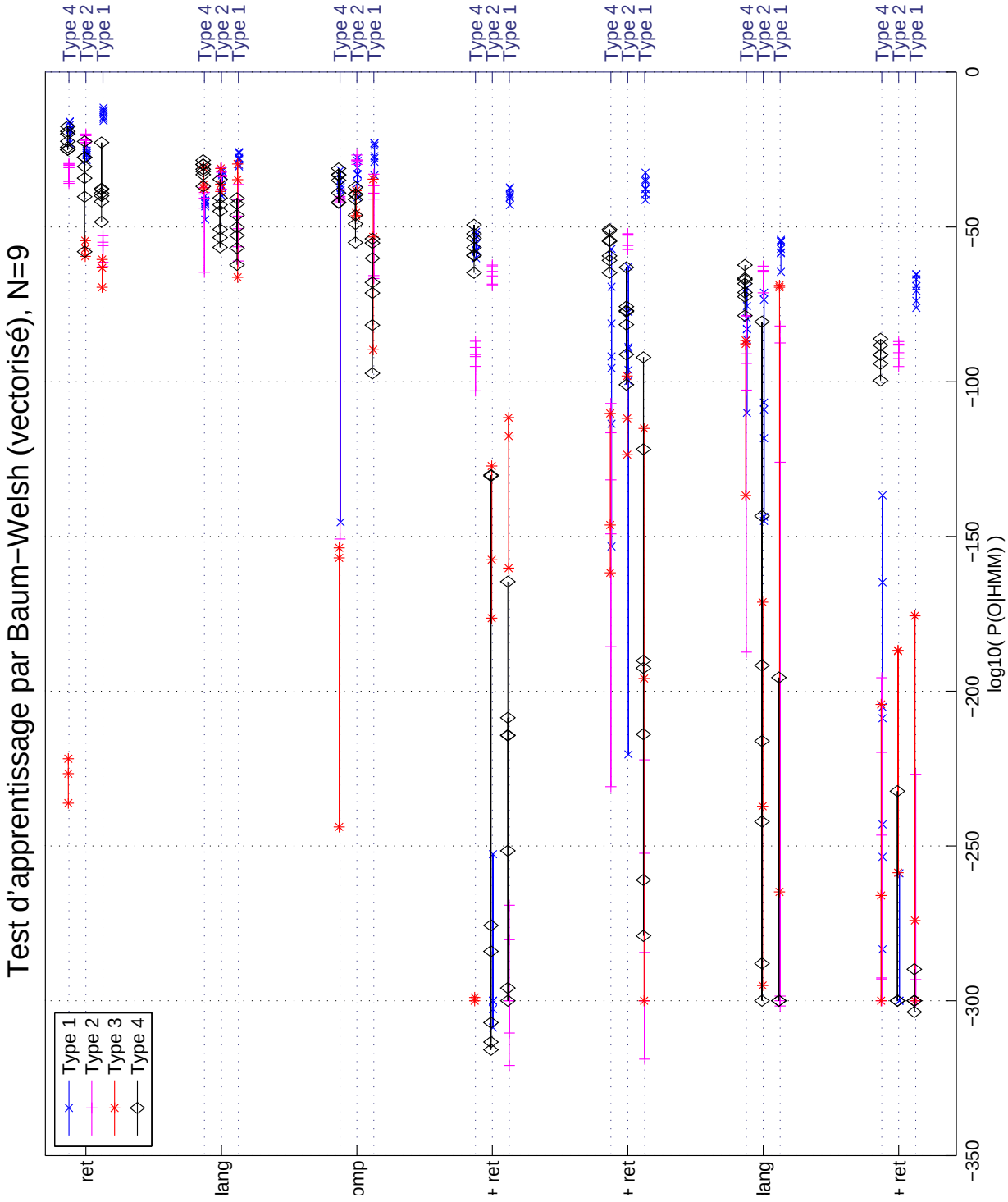
Les résultats de ces tests de modélisation à l'aide de MMC-MD montrent qu'il est possible de réaliser un apprentissage sans erreur en utilisant la vectorisation des observations et avec les MMC-MD. Cependant seule la méthode d'apprentissage par Baum-Welch, présentée dans la figure 4.4.1.2, a donné des résultats acceptables. Ils montrent qu'il est possible d'effectuer une classification des modèles appris en utilisant 9 états dans le MMC-MD. Comme pour la méthode de vectorisation, ces résultats ne sont valables que si l'on utilise uniquement les données provenant des capteurs.

Il existe donc un avantage pour les MMC-MD puisque la classification s'effectue sans erreur avec 9 états cachés alors que la méthode de vectorisation en requière 12 ce qui demande plus de temps de calculs. Il est aussi bon de noter que ces résultats ne s'obtiennent qu'en utilisant les données capteurs comme vecteurs d'observation. Cependant l'ajout des comportements comme vecteur d'observation diminue grandement la qualité de la classification en plus d'en augmenter les temps de calcul à cause des calculs supplémentaires à réaliser.

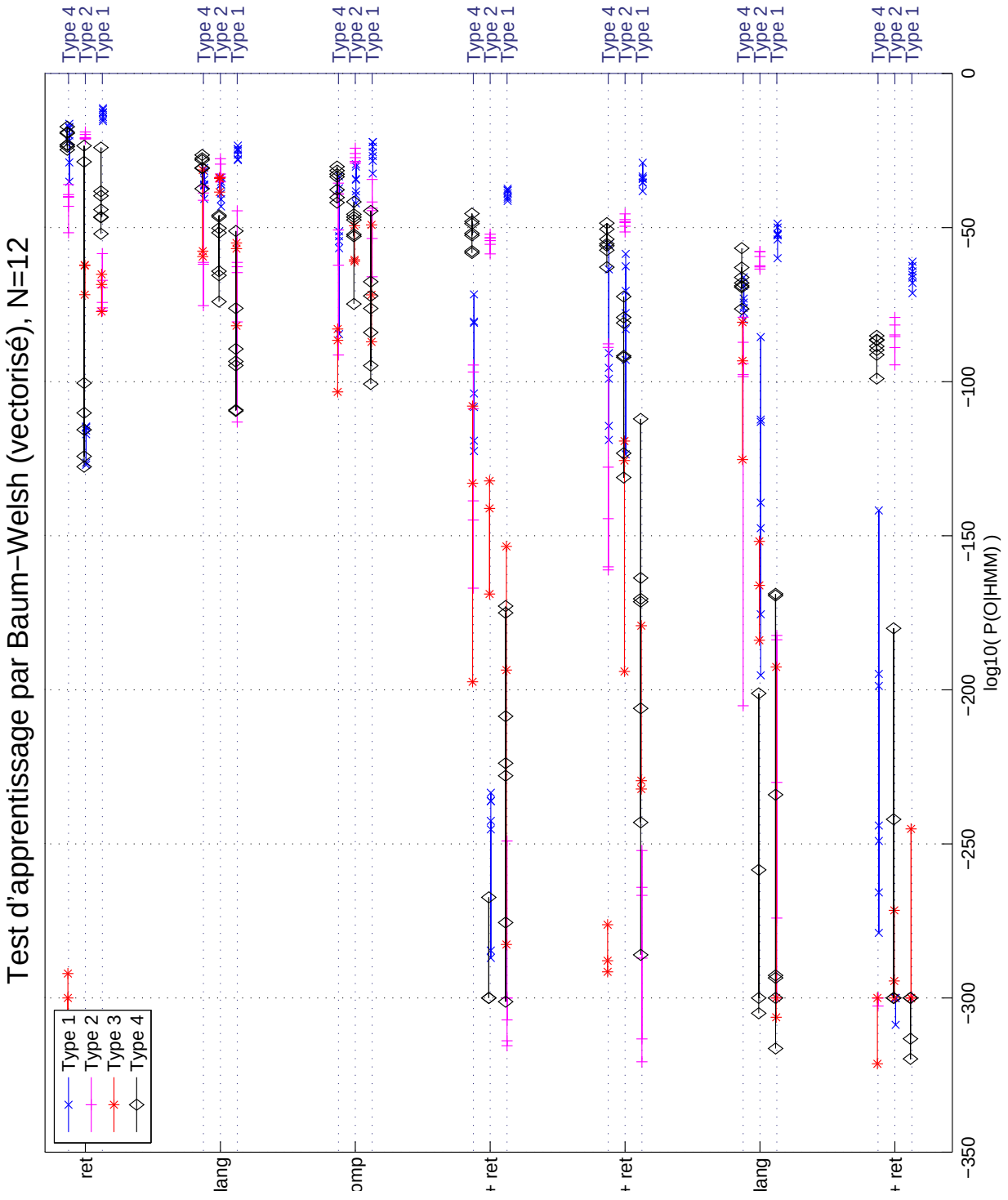
4.4.2 Tests de généralisation

Les tests de généralisation se sont déroulés de la même manière que les tests d'apprentissage à l'exception du fait que seuls les 4 premiers trajets de chaque type de trajets ont été appris, les autres servant à tester la capacité de généralisation des modèles de trajets.

Les résultats de ces tests, visibles dans la figure 4.13 montrent que, dans notre cas, la généralisation n'est pas réalisable, en tout cas pas avec aussi peu d'exemples disponibles pour l'apprentissage. Or, dans notre cas, il est

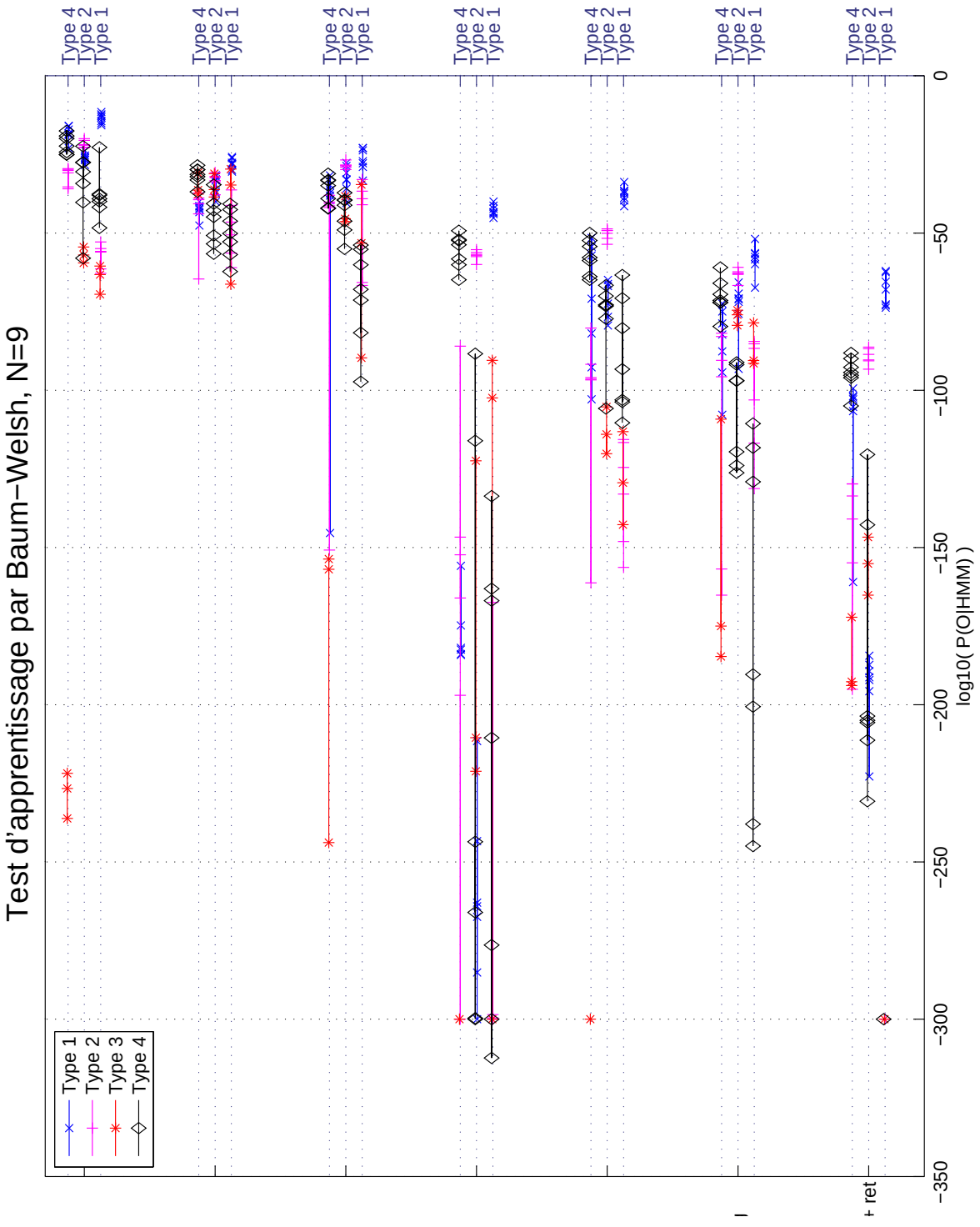


(a) Modélisation par l'algorithme de Baum-Welch, vectorisation des observations, N=9 états. Le type appris est donné à droite.

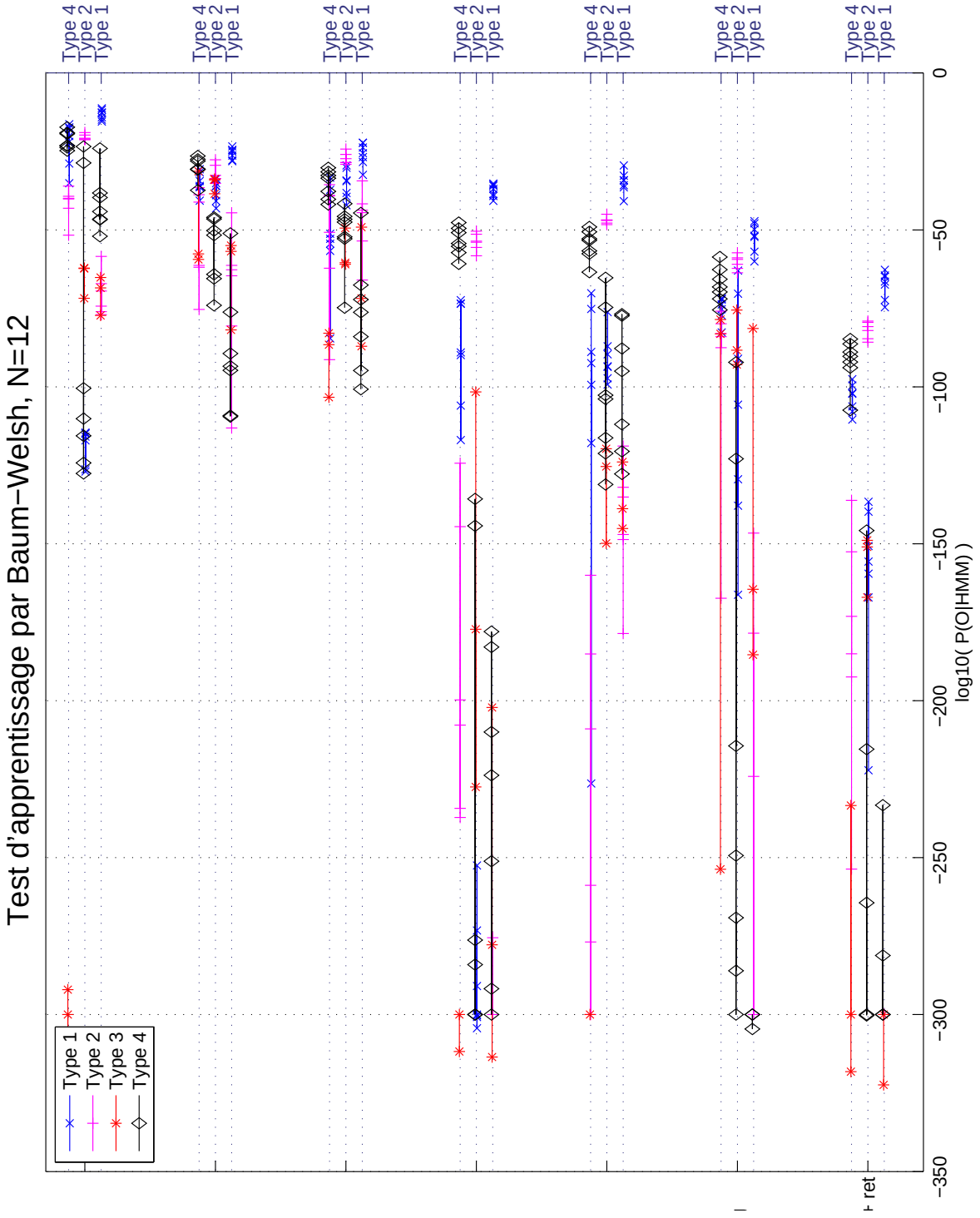


(b) Modélisation par l'algorithme de Baum-Welch, vectorisation des observations, N=12 états. Le type appris est donné à droite.

FIG. 4.11: Modélisation des observations vectorisées avec apprentissage par l'algorithme de Baum-Welch. (ret=rétrécissement, dang=vitesse angulaire et comp=comportement)



(a) Modélisation par l'algorithme de Baum-Welch multidimentionnel, N=9 états. Le type appris est donné à droite.



(b) Modélisation par l'algorithme de Baum-Welch multidimentionnel, N=12 états. Le type appris est donné à droite.

FIG. 4.12: Modélisation avec les MMC-MD avec apprentissage par l'algorithme de Baum-Welch. (ret=rétrécissement, dang=vitesse angulaire et comp=comportement)

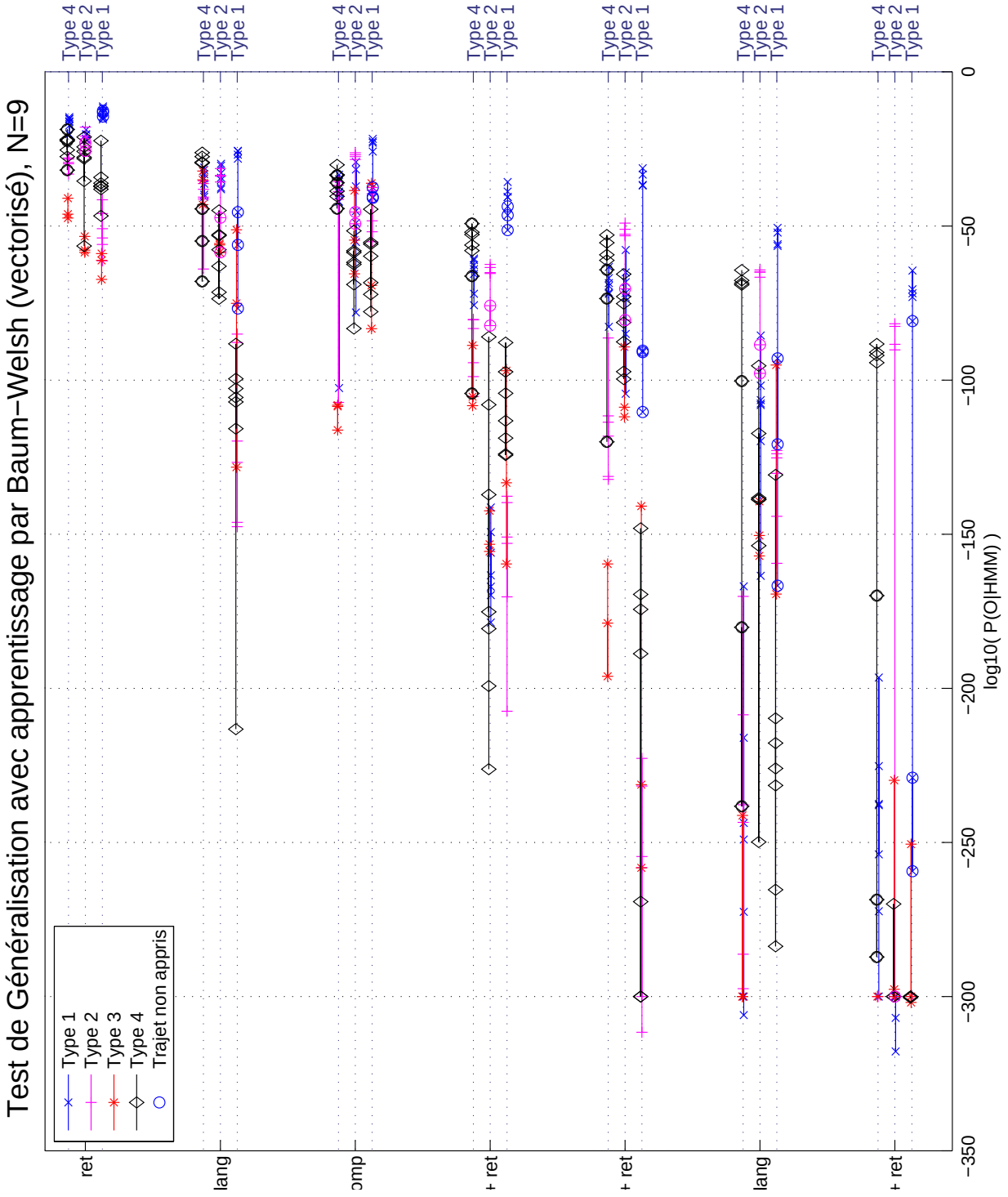
Vecteurs de données	Temps d'apprentissage (s/trajet)	Temps de reconnaissance (s/trajet/modèle)
Comportement	227	0.196
Vitesse angulaire	204	0.118
Rétrécissement	245	0.132
Comportement + Vitesse angulaire	442	0.098
Comportement + Rétrécissement	290	0.110
Vitesse Angulaire + Rétrécissement	451	0.109
Comportement + Vitesse angulaire + Rétrécissement	455	0.158

TAB. 4.2: Temps moyen d'apprentissage du modèle et de reconnaissance pour les MMC-MD avec 9 états.

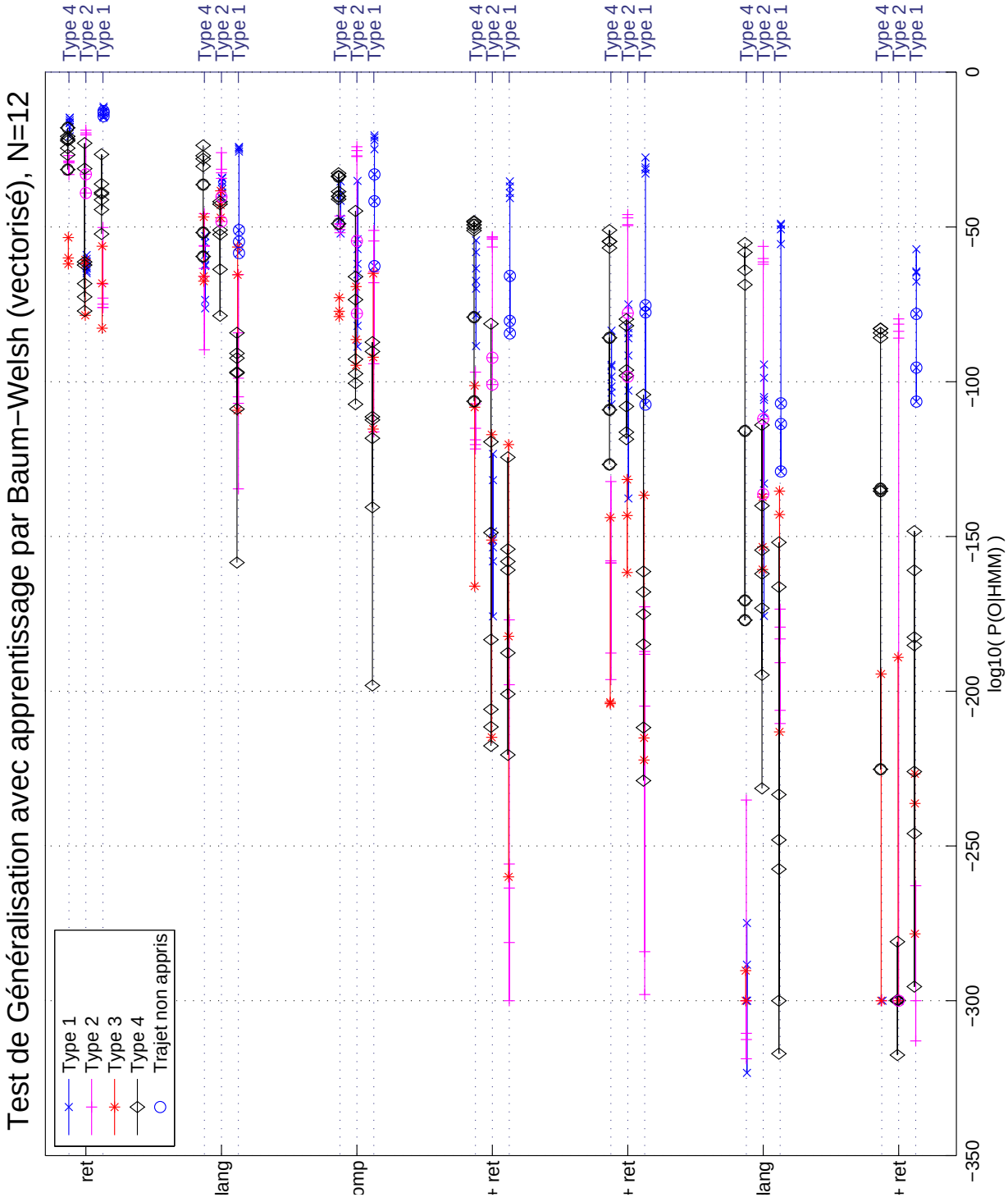
impératif de pouvoir généraliser les modèles à partir d'un petit nombre d'exemples. En effet ce système ayant pour but d'être monté sur un système autonome utilisé par une personne handicapée, il n'est pas concevable que la personne doive réaliser de nombreuses fois les mêmes trajets pour pouvoir les modéliser.

4.4.3 Temps de calculs

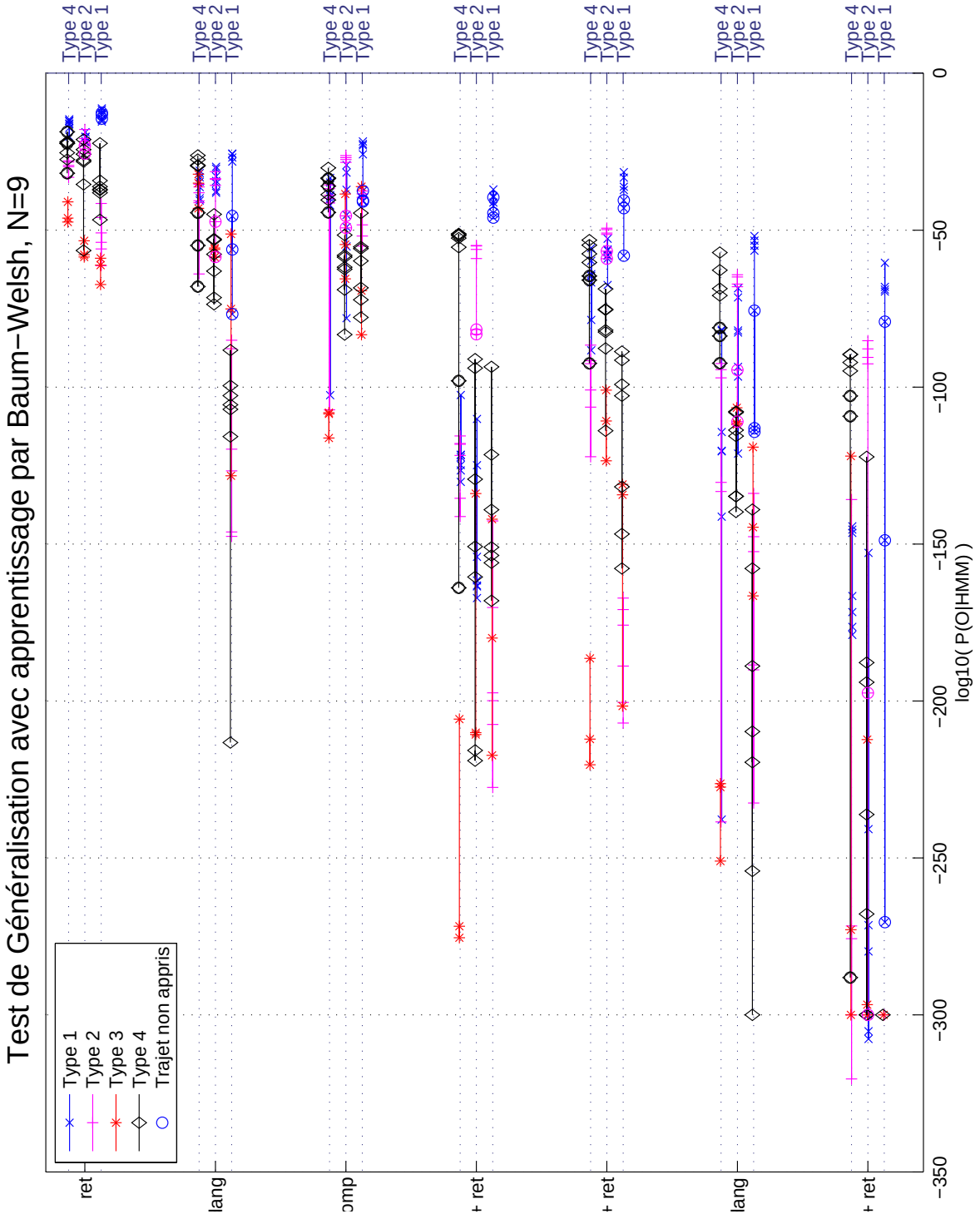
Les temps de calculs nécessaires à la modélisation et à la reconnaissance sont très importants car le système doit être utilisé sur un système embarqué. Les calculs ont été effectués sous Matlab avec un PC équipé d'un P4 cadencé à 2.66Ghz avec 512Mo de RAM. Les temps moyens de calculs pour les MMC-MD qui sont donnés dans la table 4.2 montrent des temps d'apprentissage de 455s/trajet pour le cas où on apprend les vecteurs d'observations comportement + vitesse angulaire + rétrécissement pour un modèle à 9 états cachés. Le temps de parcours moyen dans ces mêmes conditions est de 0.158s/trajet/modèle. Ces temps, bien que pouvant être grandement diminués en utilisant un langage plus rapide tel le C/C++, sont très importants surtout si l'on considère qu'un milieu hospitalier peut contenir aisément quelques dizaines de modèles.



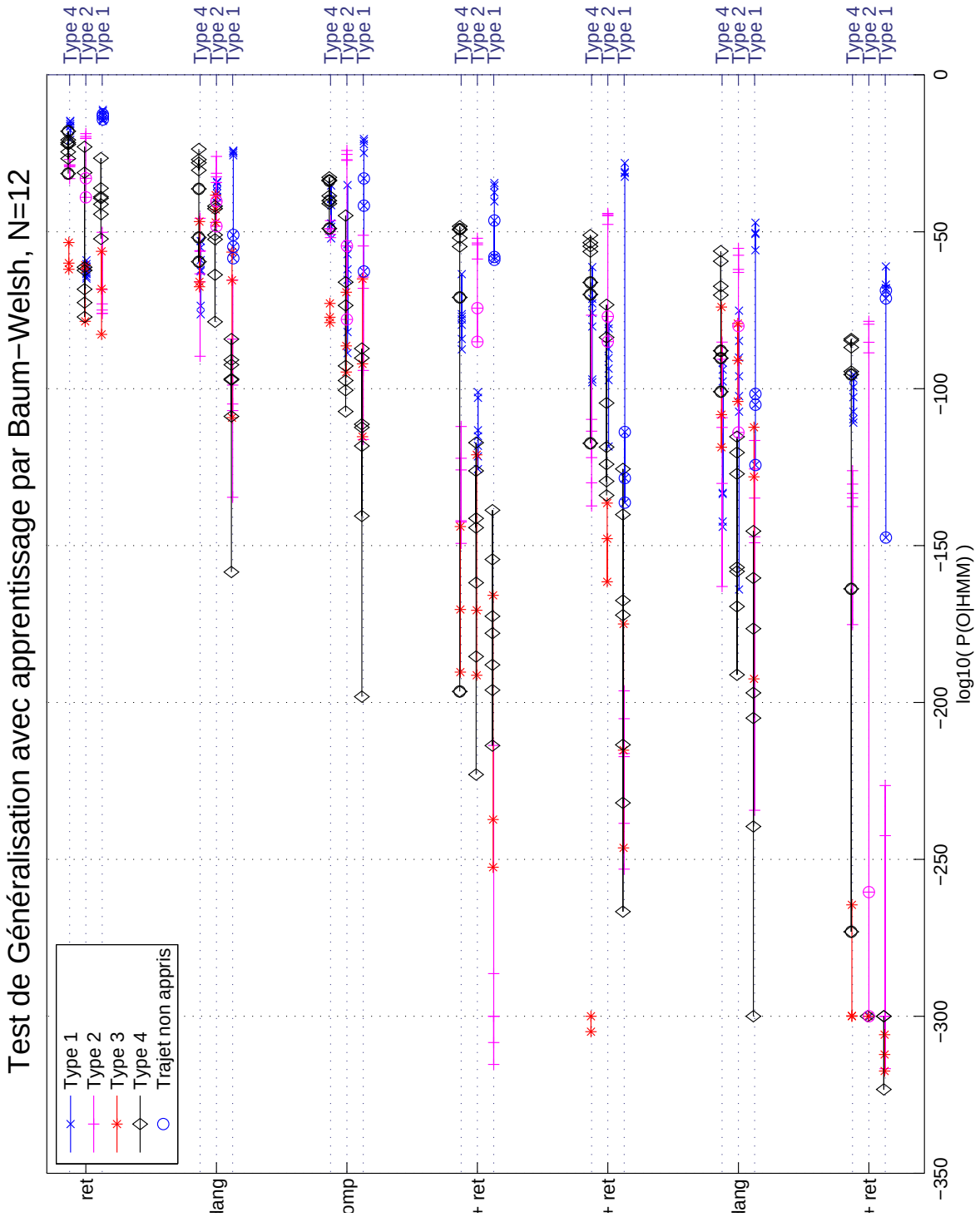
(a) Test de généralisation avec apprentissage par l'algorithme de Baum-Welch, vectorisation des observations, N=9 états. Le type appris est donné à droite.



(b) Test de généralisation avec apprentissage par l'algorithme de Baum-Welch, vectorisation des observations, N=12 états. Le type appris est donné à droite.



(c) Test de généralisation avec apprentissage par l'algorithme de Baum-Welch multidimentionnel, N=9 états. Le type appris est donné à droite.



(d) Test de généralisation avec apprentissage par l'algorithme de Baum-Welch multidimentionnel, N=12 états. Le type appris est donné à droite.

FIG. 4.13: Test de généralisation avec apprentissage par l'algorithme de Baum-Welch. (ret=rétrécissement, dang=vitesse angulaire et comp=comportent)

4.5 Conclusion

Cette étude a clairement mis en évidence que les séquences de comportements ne constituent pas des données suffisamment pertinentes pour la réalisation d'apprentissage à l'aide de modèles de Markov Cachés.

Cependant ces résultats ont montré que les données provenant des capteurs étaient plus pertinentes que les comportements. Un autre problème lié aux MMC en général est qu'ils sont sensibles à la taille du trajet appris ce qui n'autorise pas de liberté dans le découpage des trajets et donc dans la construction d'un environnement composé d'un ensemble de trajets. De plus les temps de calculs sont importants et posent des problèmes sur la possibilité de réalisation d'un système temps réel.

Pour toutes ces raisons, les recherches ont été orientées vers les calculs de différences directes entre les courbes provenant des données des capteurs et plus précisément avec un algorithme issu des filtres de MONTE-CARLO, l'algorithme CONDENSATION.

5 Mise en correspondance de courbes

LE s résultats précédents utilisant les MMC et les MMC-MD nous ont montré qu'il était difficile de modéliser des trajets en utilisant la séquence de comportements. L'utilisation des MMC résultait du fait que nous ne voulions initialement prendre en compte que les données de haut niveau que sont les comportements. Cependant cette étude montre que l'utilisation des données provenant des capteurs est plus pertinente. Nous nous sommes donc tournés vers une solution utilisant uniquement ces données. En effet, la représentation de l'évolution des données capteurs forment des courbes qui peuvent être comparées. C'est sur cette remarque que se base la suite du travail.

5.1 Introduction

Suite à l'étude de la reconnaissance de trajet à l'aide des MMC, nous nous sommes rendu compte que l'utilisation des courbes formées par les données provenant des capteurs permettait la reconnaissance d'un trajet. Cependant cette reconnaissance possédait plusieurs inconvénients dont celui de ne pas donner la position de la reconnaissance dans le modèle. Ce qui est un inconvénient pour obtenir une localisation précise. Notre attention c'est alors portée sur l'utilisation d'une méthode d'appariement des courbes avec des modèles de trajets.

Il existe de nombreuses méthodes d'appariement de courbes de valeurs continues. Les méthodes principales sont les mêmes que celles utilisées pour l'appariement entre les données locales et globales pour la localisation géométrique. Ces méthodes, décrites dans le chapitre 2.4.2.3, utilisent les algorithmes des moindres carrés, de Kalman[45], des MMC [80] et de CONDENSATION [42] (aussi appelés filtres particulaires ou algorithme de MONTE-CARLO). A ces algorithmes nous pouvons ajouter ceux des contours actifs (SNAKES) [46].

Chacun de ces algorithmes ont leurs avantages et inconvénients. Notre première idée fut d'utiliser l'algorithme le plus simple à savoir celui des moindres carrés en utilisant une méthode d'appariement directe. Cette étude est détaillée dans la première partie de ce chapitre. Mais, bien que des correspondances existent entre les différents trajets d'un même type, des erreurs dues à l'odométrie, tel que le glissement des roues, ou aux différences de trajectoires (virages plus ou moins serrés) déforment les courbes. Un algorithme permettant d'intégrer ces déformations dans le modèle était alors nécessaire. C'est pourquoi, après avoir rappelé les points nécessaires à cette étude, nous réalisons une comparaison rapide des avantages et inconvénients des différents algorithmes.

La suite de cette partie présente en détail l'algorithme CONDENSATION qui est l'algorithme que nous avons choisi. Puis nous présentons les légères adaptations réalisées sur les hypothèses de départ et la fonction de vraisemblance. Nous présentons ensuite les résultats concluant des essais préliminaires.

5.2 Mise en correspondance directe de courbes.

Le but de cette mise en correspondance est de pouvoir classifier le trajet en cours par rapport aux trajets déjà effectués. Si l'on prend les courbes des trajets d'un même type, on peut s'apercevoir qu'il existe des similarités dans les courbes que réalisent les données capteurs tout au long des trajets comme le montre la figure 5.1. Il semble donc possible de réaliser une reconnaissance à l'aide de fenêtres glissantes. Pour cela, on prend un trajet effectué et on le considère comme un modèle. On prend ensuite un morceau de taille fixe d'un deuxième trajet et on essaye de retrouver ce morceau de trajet dans le modèle comme le montre la figure 5.2. Les trajets utilisés sont les mêmes trajets que pour les essais avec les MMC qui sont présentés dans le chapitre 3.4.

L'algorithme de recherche par fenêtre glissante se présente de la manière suivante. Si l'on considère l'ensemble des observations O composé de K

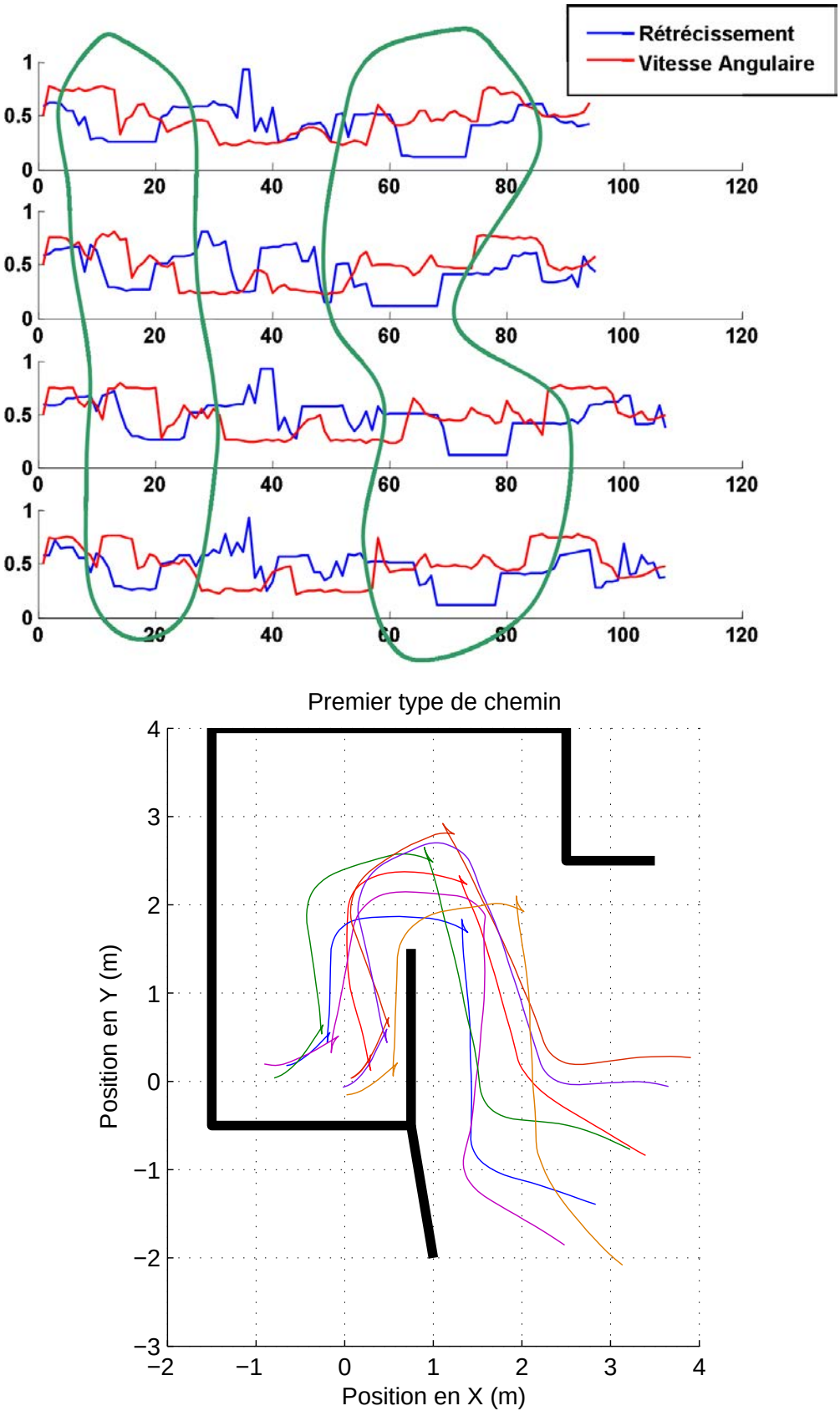


FIG. 5.1: Exemple de similarité entre des courbes des trajets du type 1 avec l'environnement de type 1.

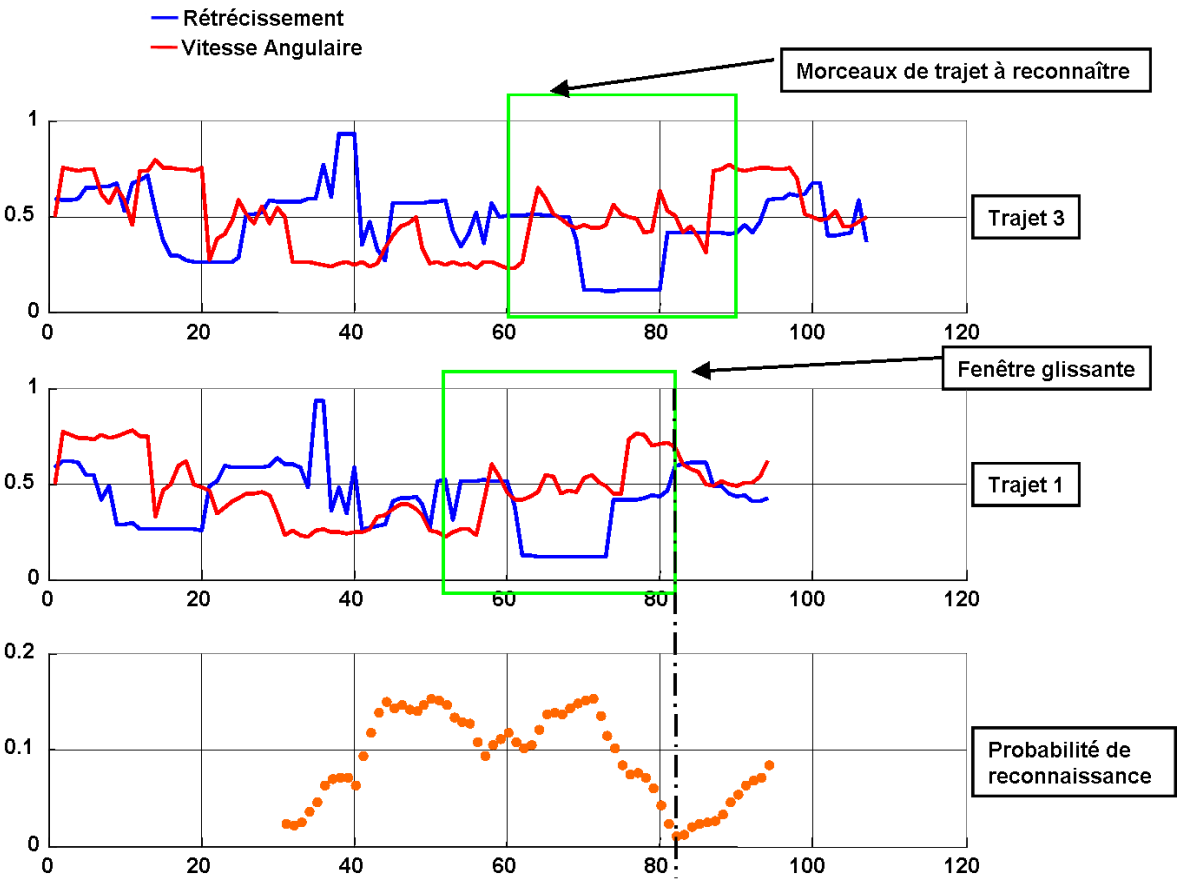


FIG. 5.2: Mise en correspondance de courbes en utilisant l'algorithme des moindres carrés. L'exemple est donné sur les courbes des trajets 1 et 3 du type 1. La zone à reconnaître débute à la position 60 dans le trajet 3 et la reconnaissance a eu lieu à la position 82 dans le trajet 1 qui a été pris pour modèle.

vecteurs de N observations

$$O = \begin{bmatrix} o_1^1 & \cdots & o_n^1 & \cdots & o_N^1 \\ \vdots & \ddots & \vdots & & \vdots \\ o_1^k & \cdots & o_n^k & \cdots & o_N^k \\ \vdots & & \vdots & \ddots & \vdots \\ o_1^K & \cdots & o_n^K & \cdots & o_N^K \end{bmatrix},$$

avec $n \in [0, N]$ et $k \in [0, K]$ correspondant au trajet en cours et I modèles M composé également de K vecteurs de longueur L tel que

$$M_i = \begin{bmatrix} m_1^1 & \cdots & m_n^1 & \cdots & m_N^1 \\ \vdots & \ddots & \vdots & & \vdots \\ m_1^k & \cdots & m_n^k & \cdots & m_N^k \\ \vdots & & \vdots & \ddots & \vdots \\ m_1^K & \cdots & m_n^K & \cdots & m_N^K \end{bmatrix}$$

avec $i \in [0, I]$. La mise en correspondance s'effectue sur une fenêtre de taille W avec $W < N$ et $W < L$. On prend donc les W dernières observations et on parcourt les modèles en faisant glisser la fenêtre du début à la fin du trajet pris comme modèle. A chaque incrément j de position on calcule la probabilité de vraisemblance V de la manière suivante :

$$V_{i,j} = \prod_{k=0}^{K} \frac{\sum_{w=0}^{W} (O_{N-w}^k - M_{i-w}^k)^2}{W}.$$

On considère que le modèle i et la position j à laquelle la vraisemblance V est minimale est la position où se trouve le fauteuil. Cependant cette méthode est très longue puisqu'elle demande de parcourir tous les modèles à toutes les positions. De plus il se trouve que si le fauteuil ne passe pas exactement au même endroit, les courbes sont plus ou moins déformées comme on peut le voir dans la figure 5.3.

Ces problèmes peuvent être importants notamment dans les cas de fort glissement. Nous avons donc dû trouver une solution pour déformer le modèle de manière à le faire « coller » au mieux à la courbe du trajet en cours. Un choix de l'algorithme d'appariement doit donc être réalisé.

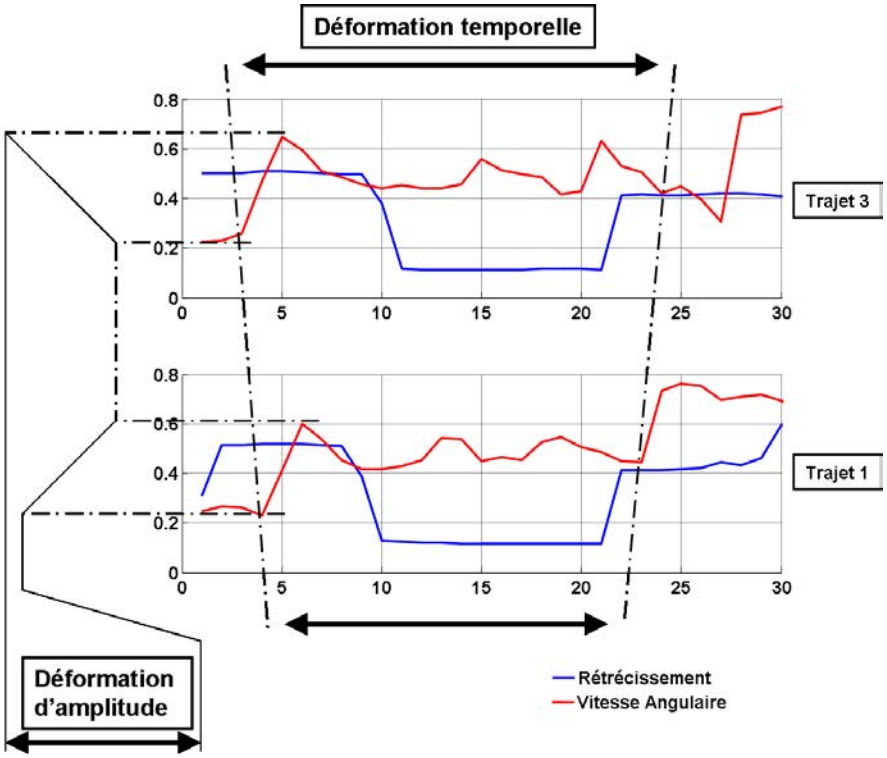


FIG. 5.3: Exemple de déformations entre les courbes des trajets 1 et 3 aux points de reconnaissance de la figure 5.2.

5.3 Choix de l'algorithme d'appariement

L'algorithme d'appariement doit répondre à certaines contraintes pour réaliser la reconnaissance mais aussi pour permettre la localisation. Ces contraintes sont les suivantes :

- déformer le modèle de trajet
- fonctionner en temps réel avec plusieurs vecteurs d'observation
- permettre de répondre au problème du robot téléporté (initialisation automatique)
- donner la position dans le modèle

Les algorithmes les plus courants sont basés sur les moindres carrés, les filtres de Kalman [45], les MMC continus [80], CONDENSATION (filtres particulaires)[42] ou les contours actifs (SNAKES) [46]. Il n'est pas question ici de détailler ces algorithmes mais uniquement de comparer leurs capacités. Le tableau 5.1 donne si oui on non les algorithmes sont capables, en fonction de leur nature même, de répondre à nos besoins.

Les résultats de cette rapide étude des différents algorithmes d'appariement nous donnent que seul l'algorithme CONDENSATION (**CONditionnal DENSity** Propag**ation**) répond à tous les critères que nous avons fixés pour

	Déformations des modèles	Temps-réels avec plusieurs vecteurs d'observation	Résout le problème du robot téléporté	Donne la position dans le modèle
moindres carrés	non	oui	non	oui
Kalman	oui	probablement	non	oui
MMC-continus	oui	non	oui	non
Condensation	oui	oui	oui	oui
SNAKES	oui	probablement	non	oui

TAB. 5.1: Comparaison des algorithmes d'appariement en fonction de nos besoins pour réaliser la reconnaissance de trajet.

réaliser la reconnaissance de trajet. La partie suivante présente donc cet algorithme en détail.

5.4 Utilisation de l'algorithme CONDENSATION

L'algorithme CONDENSATION fût introduit en 1996 par Michael Isard et Andrew Blake [42] dans le domaine du traitement d'images pour traquer les objets dans une vidéo. Il fut ensuite utilisé pour reconnaître des séquences temporelles [10] représentées par les mouvements d'un curseur sur un tableau blanc.

Cet algorithme se rapproche des filtres particulaires dans la mesure où il utilise une multitude d'éléments identiques pour tester les possibilités en utilisant une répartition aléatoire, au lieu de tester toutes les possibilités à chaque itération. La convergence d'un tel système dépend de sa capacité à amener, au fur et à mesure des itérations, les particules vers une solution. Ici la convergence est assurée par le choix des particules en fonction de leurs probabilités et les particules sont appelées état.

5.4.1 L'algorithme CONDENSATION pour la reconnaissance de trajets

On définit un état comme étant un ensemble de paramètres qui aligne un modèle de trajet sur les valeurs d'entrées. Un état inclut des paramètres qui contrôlent les ajustements d'amplitudes et d'échantillonnages locaux ainsi que la translation du modèle par rapport aux observations (valeurs d'entrées du système). On définit la probabilité d'un état par sa correspon-

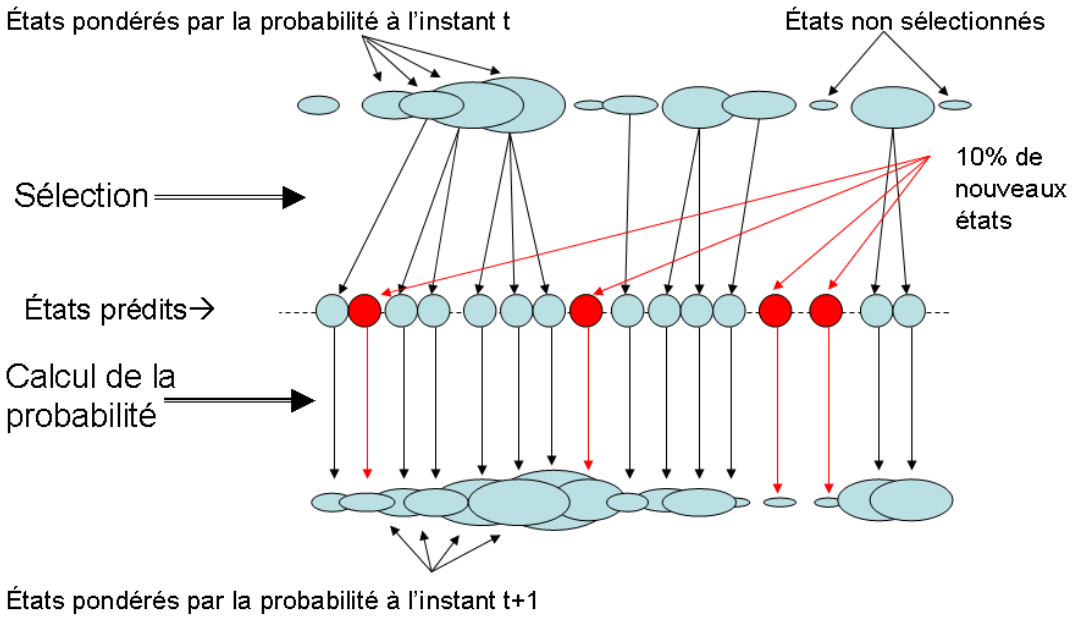


FIG. 5.4: Fonctionnement de l'algorithme CONDENSATION.

dance avec les observations. Un ensemble d'états, avec leurs probabilités, définit une distribution de probabilités échantillonnées sur l'ensemble des paramètres. Cette distribution évolue en même temps que les observations. L'algorithme CONDENSATION utilise des techniques d'échantillonnage dynamiques et aléatoires pour poursuivre cette distribution au cours de son évolution. La figure 5.4 résume le fonctionnement de l'algorithme.

Le but est donc de faire correspondre un ensemble M de modèles de trajets $\{\mathbf{m}^{(\mu)}, \mu = 1, \ldots, M\}$ avec un trajet d'entrée à D dimensions (D vecteurs d'observations), donnée par $\mathbf{z}_t = (z_{t,1}, \ldots, z_{t,D})$, à un instant t et au travers d'une fenêtre temporelle de dimension w. Les modèles sont des courbes discrètes échantillonnées avec un paramètre de phase $\phi \in [0, \phi_{\max}]$ représentant la position dans le modèle. Les valeurs d'un modèle à la position ϕ sont données à travers un vecteur de D valeurs $\mathbf{m}_\phi^{(\mu)} = (m_{\phi,1}^{(\mu)}, \ldots, m_{\phi,D}^{(\mu)})$ interpolées à la position ϕ.

On définit un état à l'instant t comme étant un vecteur de paramètres $s_t = (\mu, \phi, \alpha, \rho)$ avec :

– μ : un entier correspondant au numéro du modèle (ce nombre doit être

unique à chaque modèle),
- ϕ : position ou phase, à l'instant t, à laquelle on aligne le modèle avec les observations,
- α : un paramètre d'amplitude utilisé pour déformer le modèle en amplitude pour correspondre aux observations,
  et
- ρ : un taux qui permet de faire varier la période d'échantillonnage du modèle (ajustement horizontal du modèle).

Le but est maintenant de trouver quel est l'état $s_t^{(n)}$, $n = [1, \ldots, N]$, le plus proche des observations $Z_t = (\mathbf{z}_t, \mathbf{z}_{t-1}, \ldots)$. Prenons $Z_{t,i} = (z_{t,i}, z_{(t-1),i}, z_{(t-2),i}, \ldots)$ comme étant le i^{me} vecteur d'observation à l'instant t. On définit la probabilité d'observer $\mathbf{z}_t$ sachant l'état $s_t^{(n)}$ par

$$p(\mathbf{z}_t|s_t^{(n)}) = \prod_{i=1}^{D} p(Z_{t,i}|s_t^{(n)}), \tag{5.1}$$

où

$$p(Z_{t,i}|s_t) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-\sum_{j=0}^{w-1}\left(z_{(i-j),i} - \alpha m_{(\phi-\rho j),i}^{(\mu)}\right)^2}{2\sigma_i^2(w-1)}\right). \tag{5.2}$$

Les σ_i sont des estimations de la divergence standard pour la courbe i. En outre, $\alpha m_{(\phi-\rho j),i}^{(\mu)}$ est simplement la valeur du i^{me} coefficient dans le modèle μ interpolé à l'instant $\phi - \rho j$ et déformé par α. La figure 5.5 montre un exemple de déformations de $\pm 20\%$ d'un modèle.

La première étape de l'algorithme est l'initialisation.

1. Initialisation :
   Le but est de créer une première distribution de probabilités pour les N états du système. Pour cela on crée les états $s_t^{(n)}$ en échantillonnant uniformément leurs paramètres parmi toutes les valeurs possibles de la manière suivante :

$$\begin{aligned} &\mu \in [0, \mu_{\max}], \\ &\phi = \tfrac{1-\sqrt{y}}{\sqrt{y}}, \text{ o } y \in [0, 1], \\ &\alpha \in [\alpha_{\min}, \alpha_{\max}], \\ &\rho \in [\rho_{\min}, \rho_{\max}], \end{aligned} \tag{5.3}$$

   Il est à noter que les valeurs de ϕ sont biaisées en faveur des petites valeurs, ceci est dû au fait que l'on considère toujours entrer dans le modèle par le début (ce qui est modifié dans la suite de notre étude cf §5.4.2). La probabilité $p(\mathbf{z}_t|s_t^{(n)})$ correspondante à chaque état est

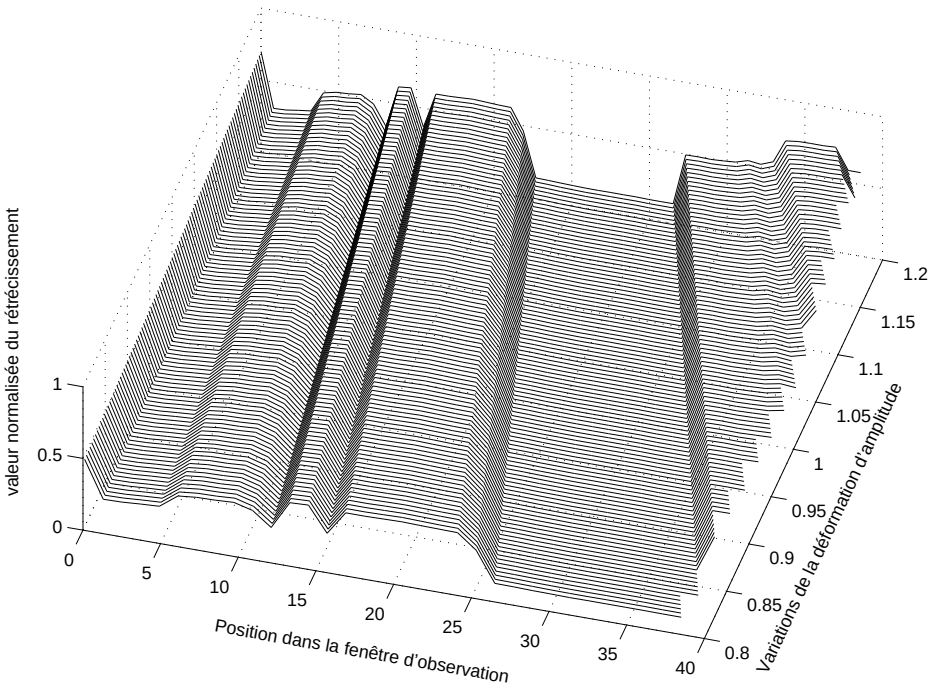


FIG. 5.5: Exemple de déformations de $\pm 20\%$ en amplitude et en échantillonage d'un modèle (courbe du rétrécissement du trajet 1 du type 1).

calculée puis on attend une nouvelle série d'observations pour entrer dans la partie de sélection des états de l'algorithme CONDENSATION.

2. Sélection :
   On normalise les probabilités de tous les états $p(\mathbf{z}_{t-1}|s_{t-1}^{(n)})$ précédemment calculées pour que leur somme soit égale à 1 ce qui nous donne les poids $\pi_{t-1}^{(n)}$

$$\pi_{t-1}^{(n)} = \frac{p(\mathbf{z}_{t-1}|s_{t-1}^{(n)})}{\sum_{i=1}^{N} p(\mathbf{z}_{t-1}|s_{t-1}^{(i)})}.$$

   On calcule ensuite la fonction de répartition $c_{t-1}^{(n)}$ comme suit :

$$\begin{aligned} c_{t-1}^{0} &= 0, \\ c_{t-1}^{(n)} &= c_{t-1}^{(n-1)} + \pi_{t-1}^{(n)}. \end{aligned}$$

   La sélection d'un état s'effectue en choisissant uniformément une valeur $r = [0, 1]$ comme le montre la figure 5.6. Le plus petit $c_{t-1}^{(n)}$ tel que $c_{t-1}^{(n)} > r$ est calculé, ce qui sélectionne l'état $s_{t-1}^{(n)}$ pour la suite de l'algorithme.
   Cette méthode permet de sélectionner le plus souvent les états ayant une probabilité élevée. De plus, pour éviter d'être piégé dans des minima locaux, on propage 5-10% d'états réinitialisés.

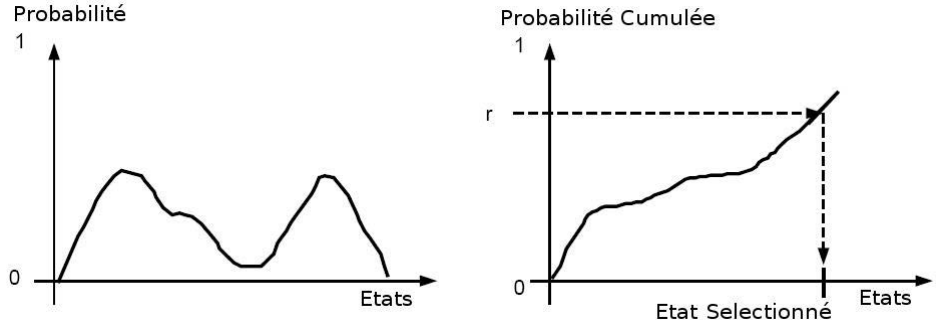


FIG. 5.6: La sélection d'un état s'effectue en échantillonnant uniformément les fonctions de répartitions. Plus la probabilité de l'état est élevée, plus il a de chance d'être sélectionné.

3. Prédiction :
   Dans cette étape on utilise l'état $s_{t-1}^{(n)}$ précédemment sélectionné et une prédiction des paramètres du nouvel état $s_t^{(n)}$ à l'instant t est effectuée. Les paramètres sont donnés par les équations ci-dessous :

$$\begin{aligned}
\mu_t &= \mu_{t-1}, \\
\phi_t &= \phi_{t-1} + \rho_{t-1} + \mathcal{N}(\sigma_\phi), \\
\alpha_t &= \alpha_{t-1} + \mathcal{N}(\sigma_\alpha), \\
\rho_t &= \rho_{t-1} + \mathcal{N}(\sigma_\rho),
\end{aligned}$$

   où $\mathcal{N}()$ est une distribution normale et les $\sigma_\star$ représente l'incertitude de prédiction des paramètres. Il est possible de noter que cette prédiction peut être vue comme un échantillonnage de la distribution de probabilité $p(s_t^{(n)} \mid s_{t-1}^{(n)})$ [43]. Pendant la prédiction, si $\phi_t > \phi_{\max}$ alors c'est que le modèle a été entièrement parcouru et l'état est réinitialisé comme expliqué ci-dessus. Lorsque les autres paramètres sont en dehors des intervalles fixés, ils sont ré-échantillonnés jusqu'à ce qu'ils soient corrects.
   Il est intéressant de noter que les $\sigma_\star$ sont utilisés pour étirer localement les paramètres. Ils peuvent être vus comme des paramètres de diffusion qui troublent la distribution de probabilités pendant l'étape de prédiction. Ils servent à réaliser une recherche locale autour d'un état. Ils autorisent aussi des déformations locales du trajet dans la fenêtre de taille w.

4. Mise à jour :
   Dans cette étape on utilise l'état $s_t^{(n)}$ précédemment prédit et on calcule sa probabilité $p(\mathbf{z}_t|s_t^{(n)})$ en utilisant l'équation 5.1. Si la probabilité

est inférieure à un seuil ε, l'étape de prédiction est répétée un certain nombre de fois et si la probabilité est toujours trop faible, l'état est réinitialisé et on calcule sa probabilité.
Une fois les N états actualisés, l'algorithme attend une nouvelle observation pour retourner à l'étape de sélection.

Quelques modifications ont été apportées à cet algorithme original afin de l'optimiser pour notre application. La première se trouve dans l'étape d'initialisation au moment du choix de ϕ_t que nous choisissons uniformément sur $\phi = [W, \phi_{\max}]$ sans biaiser le choix aux valeurs faibles de ϕ car nous considérons que nous pouvons rentrer dans un modèle à n'importe quelle position.

5.4.2 Modification de l'algorithme

L'algorithme présenté ci-avant est tel qu'il a été décrit dans [10] cependant nous avons effectué quelques modifications.

La première de ces modifications se trouve dans l'étape d'initialisation d'un état, sur la méthode de sélection de la position dans le modèle ϕ. En effet celle-ci est biaisée afin de préférer un choix des valeurs basses, cette méthode considère donc que nous commençons toujours les modèles par le début avec une certaine marge. Dans notre problématique, le point d'entrée dans le modèle n'est pas toujours situé au début de celui-ci. L'initialisation de ϕ s'effectue donc aléatoirement dans $[0, \phi_{\max}]$.

La deuxième modification se trouve sur le calcul de vraisemblance (formule 5.2). En effet la formule utilisée par les auteurs provient de [10]. Cependant cette méthode de calcul présente deux inconvénients. Le premier se situe au niveau de la plage de sortie de cette équation qui se situe dans $p \in [0.24, 0.39]$ et non dans $[0, 1]$ ce qui requière une normalisation du calcul de vraisemblance. Le second inconvénient est que l'utilisation d'une exponentielle n'est pas assez discriminante car beaucoup de nos probabilités sont élevées. De plus ce système est relativement gourmand en temps de calcul. La méthode choisie pour le calcul de vraisemblance $p(\mathbf{z}_t|s_t^{(n)})$ est donnée par les équations suivantes :

$$p(\mathbf{z}_t|s_t^{(n)}) = \prod_{i=1}^{D} p(Z_{t,i}|s_t^{(n)}),$$

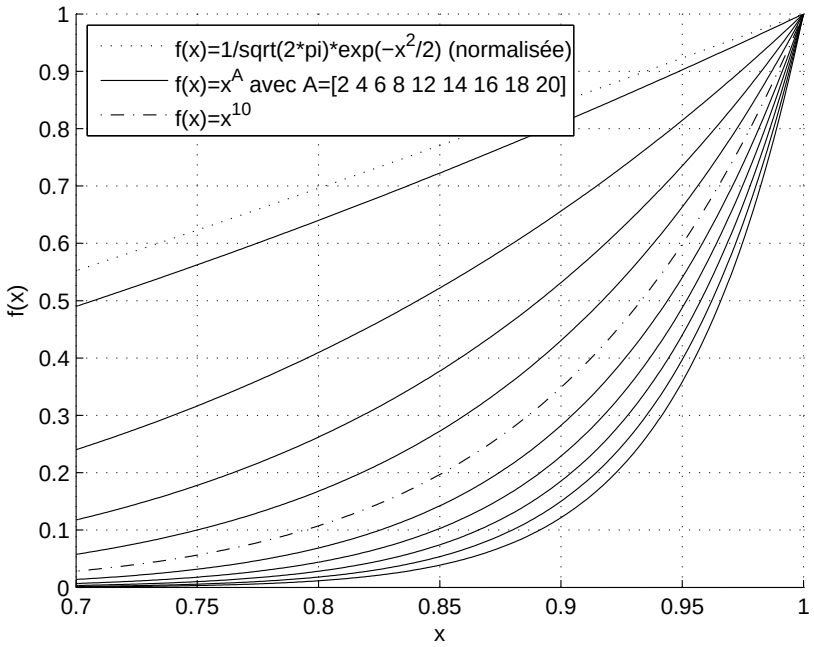


FIG. 5.7: Variations des différentes fonctions de puissance($f(x) = x^A$) et de la fonction donnée dans [10].

où

$$p(Z_{t,i}|s_t) = \left(1 - \left(\frac{\sum_{j=0}^{w-1}\left(z_{(i-j),i} - \alpha m^{(\mu)}_{(\phi-\rho j),i}\right)^2}{w-1}\right)\right)^A,$$

avec $A = 10$. La valeur de A a été déterminée empiriquement à l'aide de Matlab. La constante A sert à améliorer les résultats de la reconnaissance en étant beaucoup plus discriminante avec les probabilités de valeurs élevées comme le montre la figure 5.7 et ce avec un coût de calcul peu élevé.

5.5 Classification par algorithme CONDENSATION

Comme dans le cas de MMC, les tests de classification sont les préliminaires à l'utilisation de cet algorithme pour la reconnaissance de trajet dans le projet VAHM. Pour cela cet algorithme a été implanté dans un premier temps sous Matlab pour mesurer son efficacité. Les 23 trajets de test de 4 types différents décrits au chapitre 3.4 ont été utilisés pour réaliser un test de classification. Les vecteurs d'observations utilisés, choisis suite au résultats obtenus avec l'utilisation des MMC du chapitre 3.4, sont la vitesse angulaire et les rétrécissements avant et arrière comme représentés dans la figure 5.8.

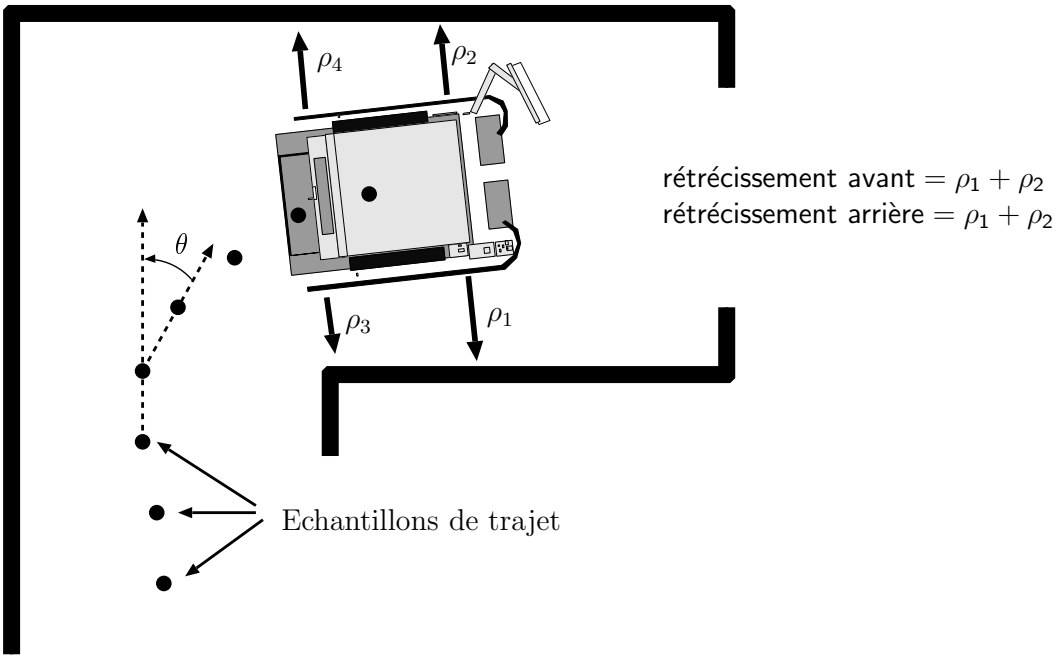


FIG. 5.8: Vecteurs d'observations choisis pour tester l'algorithme CONDENSATION.

Le test de classification requièrt un modèle pour chaque type de trajet. Ces tests étant des tests préliminaires à l'utilisation de cet algorithme et surtout de déverminage de l'implantation de l'algorithme, c'est le premier trajet de chaque groupe qui est choisi comme modèle. Les autres trajets sont utilisés comme trajets à classer. Les paramètres utilisés sont calqués sur ceux de [10] qui sont : $N = 500$ états, $W = [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]$ observations, $\alpha \in [0.7\,1.3]$ (soit $\pm 30\%$ de déformation), $\beta \in [0.7\,1.3]$ (soit $\pm 30\%$ de déformation), $\varepsilon = 0.2$, 10 prédiction avant la ré-initialisation de l'état, $\sigma_i = 1$ (pas d'estimation de la déviation standard), $\sigma_\phi = 0.5$, $\sigma_\alpha = \sigma_\rho = 0.1$. Le nombre d'états N a été réduit pour limiter le temps de calcul soutenu par le fait que nous ayons également moins de modèles à différencier. Un exemple de déformations de $\pm 20\%$ pour une fenêtre de 40 observations est donné dans la figure 5.5.

La taille de la fenêtre temporelle est importante car si elle est trop petite, il existe un risque de confusion entre des parties de trajets ayant des zones semblables. D'un autre coté si la fenêtre est trop grande elle demandera des temps de calculs supplémentaires. Les essais, consistant à trouver quelle est la taille de la fenêtre temporelle à utiliser ainsi qu'à mesurer l'efficacité globale de l'algorithme, se sont déroulés comme suit :

1. Le premier trajet de chaque type est pris comme modèle.
2. Une valeur de fenêtre temporelle W est choisie (de 10 à 60 avec un pas

de 5).

3. Tous les trajets sont parcourus en partant de $t = W$ jusqu'à la fin du trajet.
4. Les résultats de la classification sont relevés à $t = \{W + 1, W + 5, W + 10, W + 20, W + 30, W + 40, W + 50\}$.
5. Retour à l'étape 2 en changeant la taille de la fenêtre temporelle W suivant un incrément de 5.

La figure 5.9 montre le modèle du premier type de trajet ainsi qu'un autre trajet de la même classe. Le but étant de reconnaître le trajet actuel et la position dans ce trajet, les W dernières observations sont utilisées pour déterminer le modèle le plus probable ainsi que la position dans le modèle et ses déformations. Un exemple de reconnaissance est donné en figure 5.10. Cette figure montre une reconnaissance entre le premier et le troisième trajet du premier type. La zone du premier trajet, utilisé comme modèle, est déformée de 19% sur l'échelle temporelle pour correspondre, avec une probabilité de vraisemblance de 0.968, à la zone du troisième trajet présentée dans la figure. Cet exemple montre l'efficacité et l'utilité de la déformation temporelle du modèle.

Les résultats donnés dans la table 5.2 correspondent aux pourcentages de réussite de classification. Ces premiers tests nous renseignent sur l'efficacité de l'algorithme puisque nous pouvons y voir qu'une reconnaissance sans erreur est possible en utilisant une fenêtre de 35 observations et après avoir parcouru 20 échantillons. L'utilisation de modèles de trajets réalisés à partir de plusieurs trajets d'un même type ne s'est pas révélée nécessaire mais leurs utilisations restent dans les perspectives de cette étude. Les calculs ont été réalisés en utilisant l'algorithme original, issu de [10], implémenté sous Matlab. Le temps de calcul moyen, relevé sur un Pentium4 cadencé à 2.4Ghz et équipé de 512Mo de RAM, est de $0.9547s/trajet$.

En ce qui concerne l'erreur de localisation, celle-ci est difficile à évaluer. Il n'existe en effet aucune référence fiable dans les trajets que nous avons utilisés pour tester l'algorithme. Ces trajets, bien qu'effectués dans le même environnement, n'ont pas exactement les mêmes points de départ et d'arrivée ni la même orientation de départ. Le but du système est justement de ne pas utiliser de données géographiques donc aucune attention particulière n'a été prise sur la position initiale lors de la réalisation des trajets. De plus il existe des déformations locales des trajets dues aux glissements

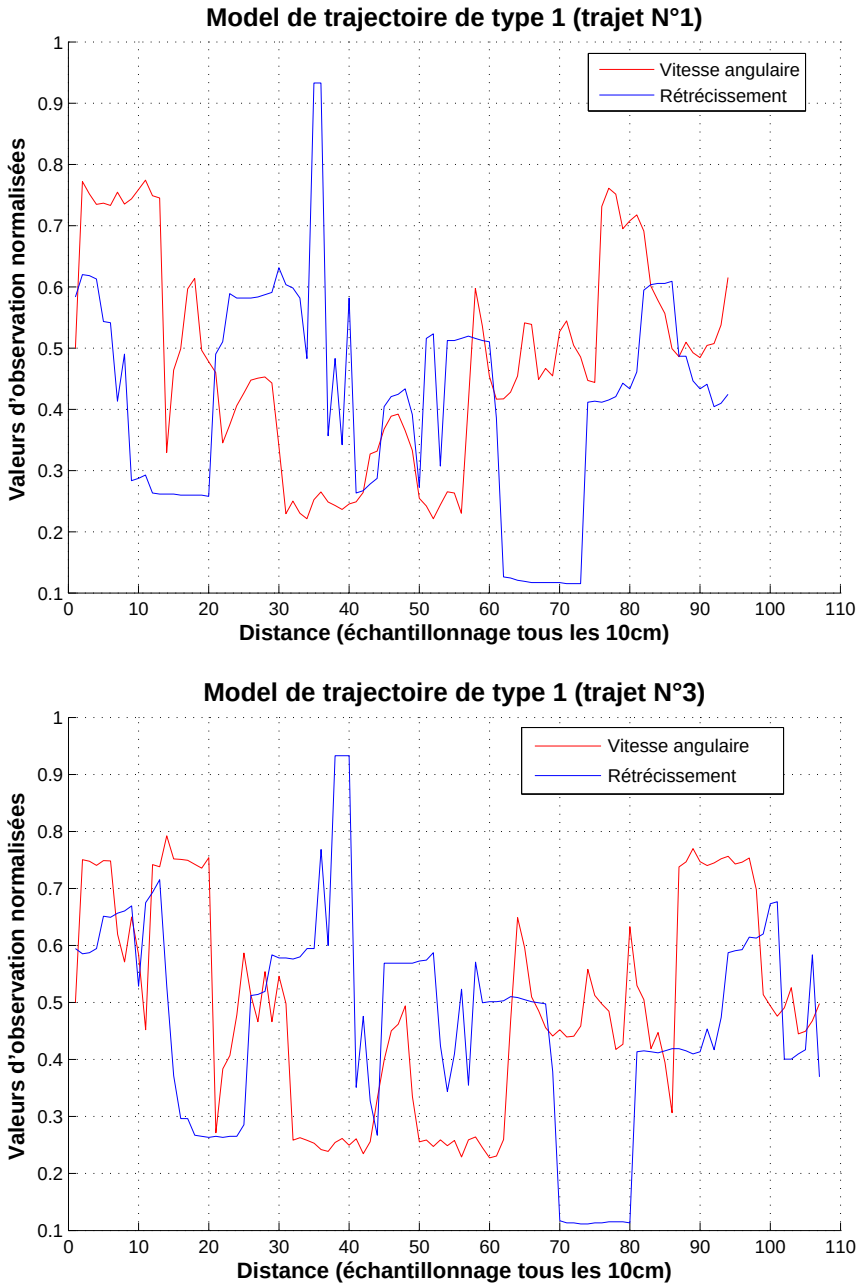


FIG. 5.9: Trajet #1 (utilisé comme modèle) et trajet #3. Ces deux trajets sont du même type.

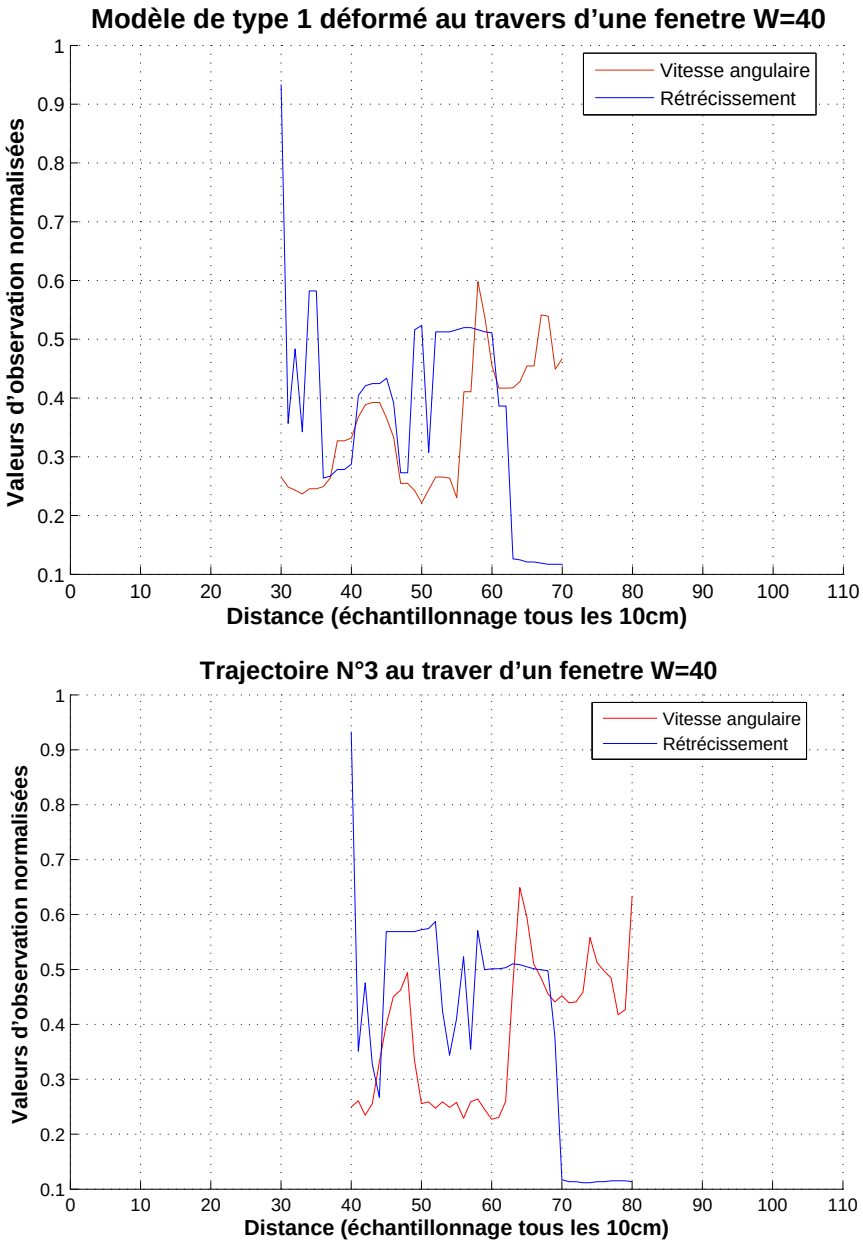


FIG. 5.10: Fenêtre du modèle (trajet #1) avec déformations ($\rho = 0.81$ et $\alpha = 0.95$) et la fenêtre correspondante sur la trajet #3 qui donne une vraisemblance de $p = 0.968$

Num. éch. \ W	10	15	20	25	30	35	40	45	50	55	60
1	61%	26%	26%	22%	35%	35%	30%	35%	39%	39%	30%
5	70%	65%	61%	61%	65%	83%	70%	65%	57%	43%	43%
10	83%	74%	70%	74%	74%	87%	87%	70%	70%	65%	48%
20	74%	70%	78%	78%	87%	100%	100%	96%	91%	96%	96%
30	57%	70%	74%	87%	91%	96%	91%	96%	91%	96%	100%
40	43%	70%	89%	57%	83%	83%	87%	91%	96%	100%	100%
50	48%	83%	70%	61%	74%	83%	83%	96%	91%	N/A	N/A

TAB. 5.2: Pourcentage de réussite de classification en fonction des différentes tailles de fenêtre W pour les 23 trajets tests. Ces tests montrent que l'algorithme CONDENSATION est capable de classer des trajets efficacement pour une taille de fenêtre temporelle W d'environ 35-40 éléments. Il est à noter que la probabilité du meilleur état n'a pas été utilisée pour discriminer les tests où aucun modèle n'aurait été réellement reconnu.

des roues sur le sol et aux différences de trajets causées par les virages plus ou moins serrés. Tout ceci a pour effet de rendre les trajets difficiles à faire correspondre et explique que ces résultats ne donnent pas l'erreur de position dans les modèles. C'est en réalisant un alignement manuel, sur des exemples où le système a reconnu sa position dans le modèle, que nous avons pu voir que la précision en position est de l'ordre du mètres (± 5 échantillons avec 10cm par échantillon).

Ces résultats préliminaires se sont montrés encourageant face aux MMC-MD dont les résultats sont présentés dans le chapitre 4.4. Les résultats des MMC-MD montrent certes des temps de parcours plus cours (environ $0.109s/trajet/modèle$) mais ils requièrent un nombre important de trajets d'apprentissage ainsi que des temps d'apprentissage importants (7 min 31s pour la méthode ayant donné les meilleurs résultats, cf tableau 4.2).

Fort de ces résultats, l'implémentation de ce système de reconnaissance sur la plate-forme VAHM a été réalisée. Le but est de mettre en oeuvre une reconnaissance en temps réel afin de réaliser des tests plus approfondis de reconnaissance dans une utilisation en temps réel.

5.6 Conclusion sur les tests de classification

Les résultats mitigés de la reconnaissance de trajet à l'aide des modèles de Markov nous ont tout de même appris qu'il était plus efficace, dans notre

contexte d'étude, d'utiliser les données issues des capteurs que de se baser uniquement sur les enchaînements de comportements . Les premiers résultats, utilisant les moindres carrés, nous ont montré qu'il était possible de différencier les trajets en faisant un calcul de différence directe entre courbes. Cependant nous nous sommes rapidement rendu compte qu'il existait quelques déformations dans les courbes. Ils nous ont également montré un avantage par rapport aux MMC qui est de donner la position de la reconnaissance dans le modèle alors que les MMC ne donnaient que le modèle.

L'algorithme CONDENSATION a alors été introduit et nous l'avons utilisé dans différents tests qui se sont montrés concluants. Ces test nous ont permis de trouver les paramètres de l'algorithme. Il nous ont également montré qu'il était possible d'utiliser un trajet comme modèle pour la reconnaissance. Ce qui fait qu'il ne requière pas d'étape de modélisation et la taille des trajets n'a pas d'importance contrairement aux MMC qui y sont sensibles.

La méthode de reconnaissance de trajet étant choisie, nous pouvons la mettre en œuvre sur le fauteuil VAHM.

6 Mise en œuvre de la reconnaissance de trajets sur le VAHM

6.1 Introduction

LE système VAHM est un système multi-agents composé d'une mémoire commune autour de laquelle gravitent différents agents ayant pour but de lire les capteurs, interagir avec l'utilisateur ou encore de réagir aux variations de l'environnement du fauteuil. L'agent réalisant la reconnaissance de trajet a pour but d'aider l'utilisateur en lui évitant de devoir ré-émettre les commandes de direction lors de son déplacement dans des environnements connus. Cet agent, que nous avons appelé « *suivi de trajet* », s'intègre parmi les agents cognitifs. Cet agent, lorsqu'il a la certitude de connaître sa position, remplace la commande de l'utilisateur par celle qu'il a enregistrée dans son modèle de trajet. La certitude de connaissance de la position est validée lorsque la probabilité de vraisemblance est suffisamment élevée et que la progression dans le modèle est linéaire. De cette manière, il utilise les agents comportements pour affiner les trajets du VAHM comme le ferait un utilisateur. Ce dernier peut à tout moment arrêter le mouvement et reprendre le contrôle du fauteuil. De plus l'agent « *suivi de trajet* » doit prévenir l'utilisateur qu'il va prendre le contrôle et changer de direction avant que cela ne se produise afin de laisser suffisamment de temps à l'utilisateur pour réagir. L'absence de réaction étant prise comme une validation du choix de direction prise par l'agent.

Les essais de reconnaissance ont été effectués sur un simulateur afin de pouvoir créer des environnements ayant pour but le déverminage et le test de l'algorithme. Par la suite nous avons utilisé des environnements réalistes dans le simulateur.

La partie suivante présente en détail le nouvel agent « *suivi de trajet* ».

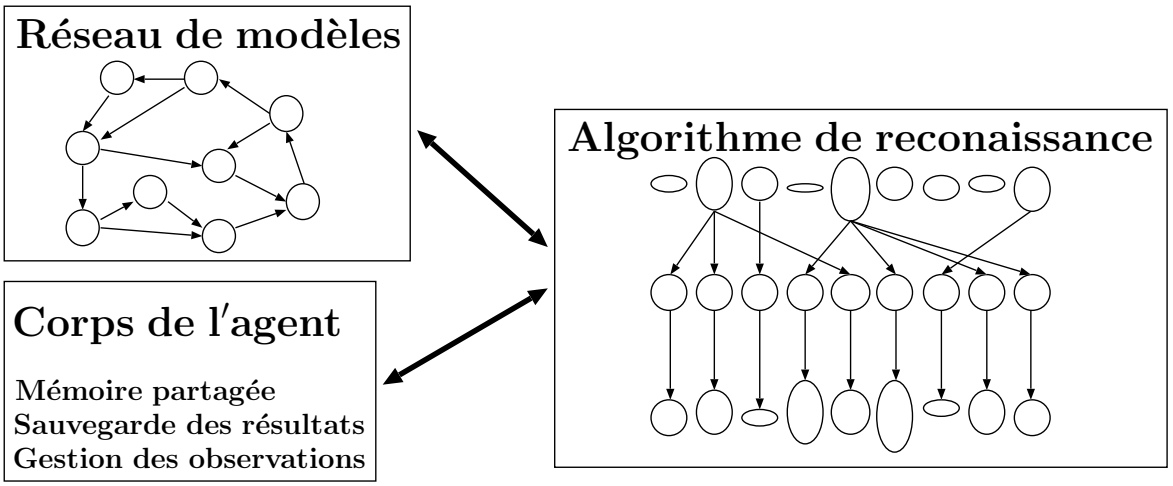


FIG. 6.1: Structure de l'agent suivi de trajet.

6.2 Agent suivi de trajet

L'agent « *suivi de trajet* », comme tous les agents du système VAHM, est codé en C++ et possède un accès à une mémoire centrale partagée. C'est à partir de cette mémoire qu'il peut interagir avec les autres agents et c'est aussi là qu'il trouve les valeurs lues par les capteurs ou encore la commande de direction demandée par l'utilisateur. Il se divise en trois parties qui sont le corps de l'agent, l'algorithme de reconnaissance et un réseau de modèles comme le montre la figure 6.1.

6.2.1 Le corps de l'agent

C'est lui qui accède à la mémoire commune et intègre les valeurs des capteurs dans un tableau d'observations représentant l'ensemble des vecteurs d'observations disponibles. La mémoire étant limitée, ce tableau de stockage est composé de 3000 éléments, soit 300m de parcours.

Le corps de l'agent exécute une reconnaissance à chaque échantillonnage et stocke les résultats dans un tableau composé également de 3000 éléments dont l'index varie en même temps que celui des observations. Il permet également d'afficher des courbes représentant la probabilité de l'état vainqueur ainsi que la position dans le modèle de cet état comme le montre la figure 6.2 .

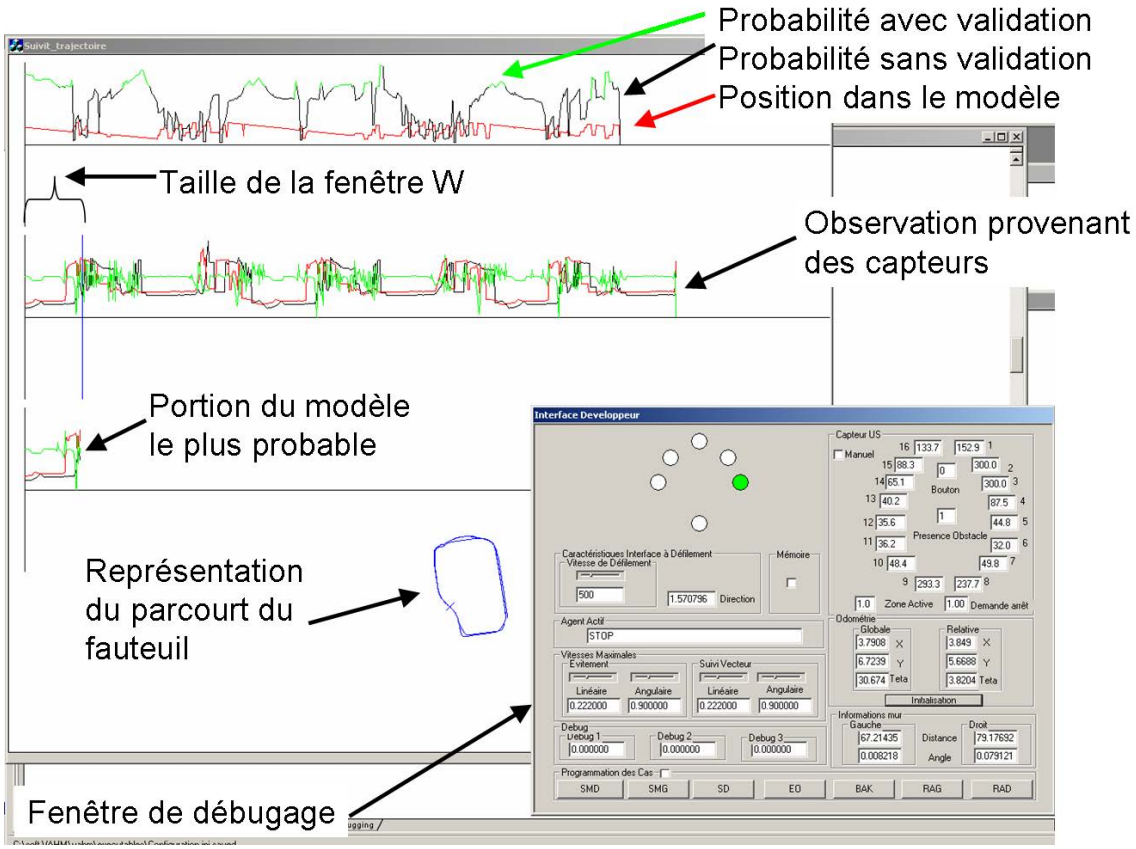


FIG. 6.2: Affichages de l'agent suivi de trajet. La courbe du haut présente la probabilité de reconnaissance avec la position dans le modèle. La deuxième courbe représente les observations des capteurs. La troisième représente la fenêtre de reconnaissance du modèle, dans notre cas elle correspond à la courbe du dessus avec la probabilité de la courbe du haut. La forme au milieu de la figure représente les déplacements du robot.

6.2.2 Le réseau de modèles

Les modèles sont placés dans un réseau afin de pouvoir réaliser des enchaînements de modèles. Ces enchaînements sont très importants car cela évite de devoir repartir de valeurs initiales lorsque le passage d'un modèle à un autre est connu. Pour cela les modèles sont placés dans un réseau chaîné, composé d'étapes et de transitions, à l'image d'une chaîne de Markov dans laquelle chaque étape correspond à un modèle et la probabilité de passer d'un état à un autre est donnée dans chaque transition. Mais le comparatif s'arrête ici car, contrairement aux chaînes de Markov, les transitions aussi possèdent un modèle de petite taille. Ceci est la solution que nous avons adoptée pour éviter de surcharger le réseau avec des états qui ne serviraient que de transition entre deux états comprenant des trajets complets. Un exemple d'enchaînement de modèles lors d'un croisement de couloir est donné dans la figure 6.3. Dans cette figure nous pouvons voir que nous utilisons un modèle par sens de parcours. Ceci est dû au fait que le fauteuil ne réagit pas de la même manière selon le sens dans lequel il prend le virage car les agents qui régissent le fauteuil sont purement réactifs. De plus, les capteurs générant les vecteurs d'observations tel que le rétrécissement ne se trouvent pas sur les axes de rotation du fauteuil ce qui entraîne des différences dans les observations.

Prenons le cas d'un couloir faisant un angle droit. Lorsque le fauteuil arrive à ce virage, les capteurs placés à l'avant du fauteuil verront le vide avant de tourner alors que dans l'autre sens le fauteuil devrait tourner avant de voir le vide pour réaliser les mêmes courbes. Or le système étant uniquement réactif un tel comportement n'a que très peu de chance d'apparaître. La figure 6.4 montre le passage d'un angle droit et on peut y voir que les réactions du fauteuil n'étant pas symétriques, les deux sens de passages ne se ressemblent pas.

A chaque arrêt du fauteuil, le corps de l'agent envoie au réseau de modèles les résultats des reconnaissances. Ces résultats sont alors parcourus pour y déceler les transitions effectuées avec succès. A chaque fois qu'une transition est parcourue sa probabilité augmente. Cette probabilité est en fait un compteur du nombre de passages réussi par cette transition. Lors de la recherche de la transition la plus probable, pour un éventuel changement de direction automatique, c'est la transition ayant le compteur le plus élevé qui est choisie.

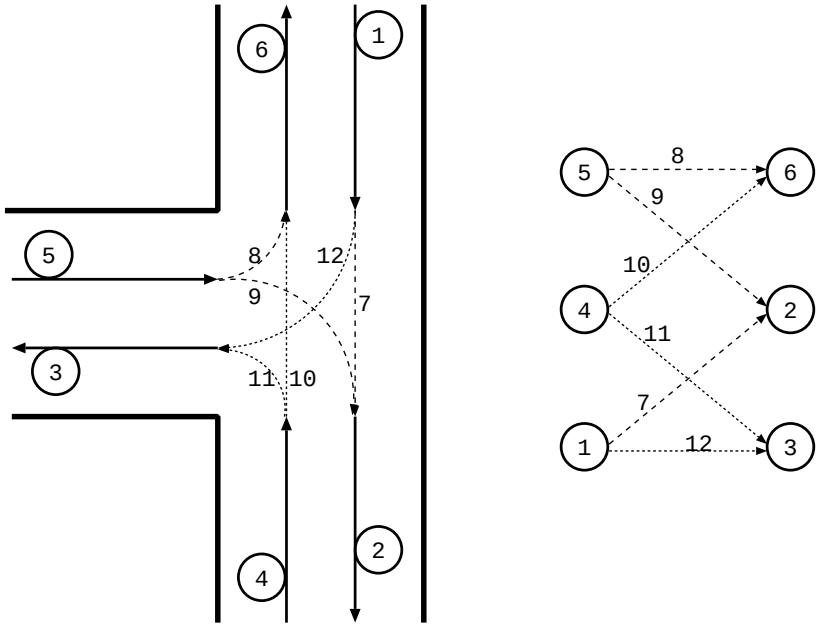


FIG. 6.3: Exemple d'intersection d'un couloir ou d'un passage de porte avec la partie du réseau associée. Le fait que le VAHM réagisse aux obstacles de manière non symétrique et que les vecteurs d'observations ne soient pas tous sur les axes de symétrie du fauteuil nous oblige à créer un modèle par sens de parcours.

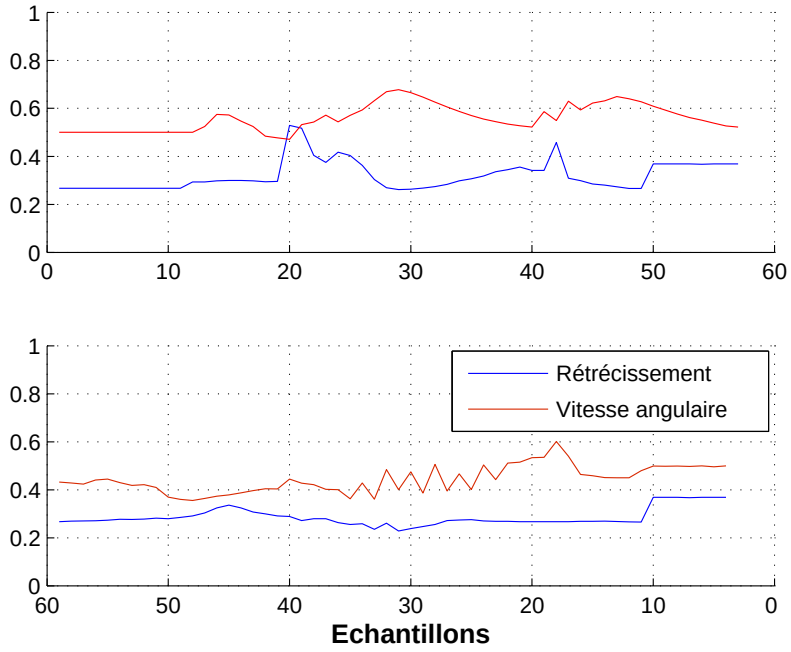


FIG. 6.4: Passage dans un couloir réalisant un angle droit. Les courbes représentes les passages d'un couloir à angle droit dans les sens opposés. Elles montrent clairement que les capteurs ne se ressemblent pas.

6.2.3 L'algorithme de reconnaissance

L'algorithme de reconnaissance utilise l'algorithme CONDENSATION décrit dans le chapitre 5.4. Il reçoit les vecteurs d'observations venant du corps de l'agent et possède également un accès au réseau de modèles. Lors de son initialisation, il lit dans un fichier de configuration une liste des vecteurs d'observation à utiliser lors de la reconnaissance. Ce système permet de modifier les vecteurs d'observations utilisés sans avoir à modifier le code. Après chaque reconnaissance, il retourne les propriétés de l'état vainqueur qui sont le numéro du modèle, la position dans le modèle, les déformations du modèle ainsi que la probabilité de cet état. Lorsque la probabilité passe en dessous d'un certain seuil, l'état est réinitialisé.

6.3 Simulation d'environnements et reconnaissance de trajets

Pour des raisons pratiques évidentes le système VAHM a été testé, dans un premier temps, dans un environnement simulé à l'aide du simulateur décrit dans le chapitre 3.3.4. Ces tests ont été réalisés au fur et à mesure des implantations des fonctionnalités de la reconnaissance de trajets. L'environnement créé dans ce but est visible dans la figure 6.5. Cet environnement a été créé de manière à pouvoir en faire un tour presque complet sans avoir à relâcher le bouton de commande. Le test présenté dans le paragraphe suivant est utilisé pour vérifier la fonctionnalité du système.

6.3.1 Validation de l'implantation de l'algorithme de reconnaissance

Afin de valider la mise en oeuvre de l'algorithme mais également de vérifier l'aspect temps-réel de la reconnaissance et de choisir les paramètres associés à l'algorithme CONDENSATION, nous avons utilisé le simulateur avec l'environnement de la figure 6.5. Le test se déroule en deux étapes. La première étant la création d'un modèle de trajet et la seconde la reconnaissance. Le modèle créé est donné dans la figure 6.6. C'est une boucle presque complète dans l'environnement. En fait elle représente la quasi totalité du parcours qu'il est possible de réaliser avec le VAHM sans avoir à relâcher le bouton. Ainsi nous pourrons vérifier si la substitution de la commande de trajectoire fonctionne en faisant passer le fauteuil dans la partie

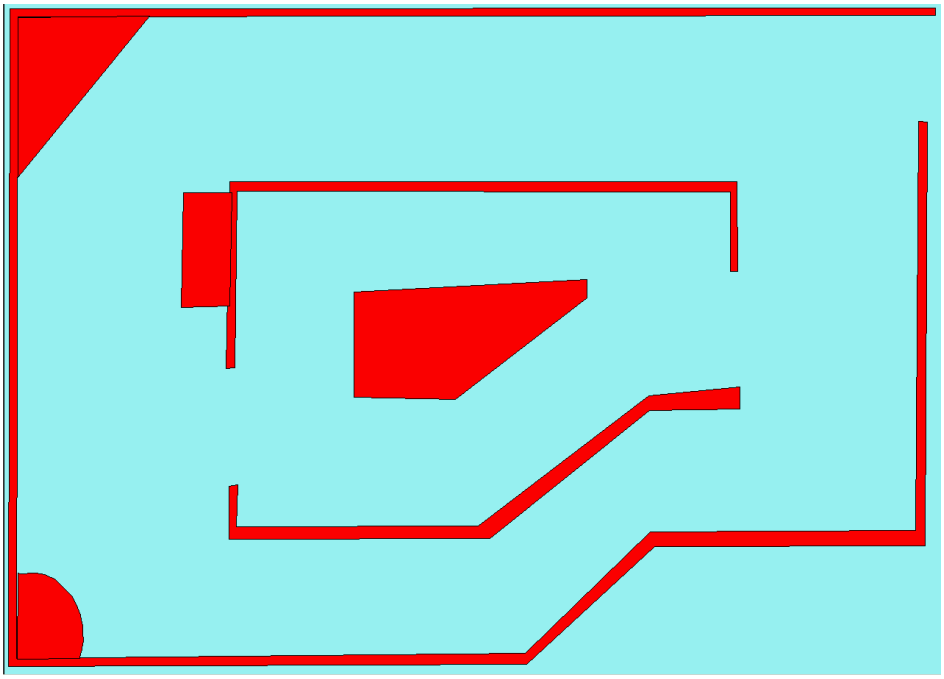


FIG. 6.5: Représentation de l'environnement de simulation utilisé pour la validation du système de reconnaissance automatique. L'espace libre est représenté en bleu et les murs en rouge.

centrale de l'environnement.

Ce premier trajet représente le modèle. Nous procédons alors au test de reconnaissance. Il consiste à se déplacer dans l'environnement en réalisant un trajet similaire ou non au modèle. Les réponses de l'agent « *suivi de trajet* » nous permettent d'évaluer le taux de reconnaissance grâce à la probabilité de reconnaissance et la variation linéaire de la position dans le modèle. Cette dernière signifiant que le système connait sa position dans l'environnement. Ces résultats, visibles dans la figure 6.7, nous montrent que la reconnaissance est bonne malgré les petites différences de trajets du fauteuil. Cette figure présente trois passages dans l'environnement de test. Le premier passage (échantillons de 0 à 300) suit le modèle avec une bonne reconnaissance visible par la probabilité élevée et la progression linéaire de la position dans le modèle. Le second passage (échantillons de 300 à 570) commence par suivre le modèle puis bifurque dans la zone centrale de l'environnement. On distingue l'endroit où le fauteuil quitte le modèle par la chute de la probabilité peu après le $400^{ième}$ échantillon. Le troisième tour dans l'environnement débute vers le $600^{ième}$ échantillon et suit le modèle.

Ces tests nous ont également permis d'ajuster les paramètres de l'algorithme CONDENSATION et les valeurs retenues sont les suivantes après une recherche empirique : $N = 500$ états, $W = 40$ observations, $\alpha \in [0.9\,1.1]$ (soit

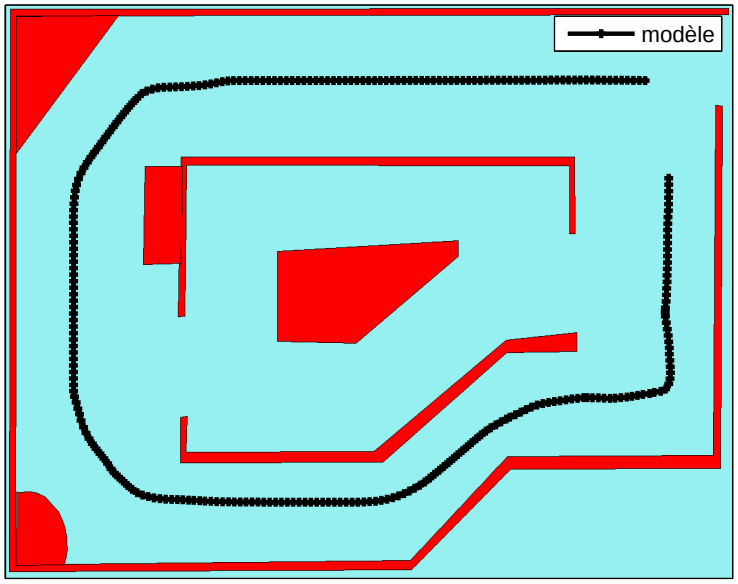


FIG. 6.6: L'environnement de test et le modèle faisant le tour de cet environnement.

$\pm 10\%$ de déformation), $\beta \in [0.8\,1.2]$ (soit $\pm 20\%$ de déformation), $\varepsilon = 0.2$, 1 prédiction avant la ré-initialisation de l'état, $\sigma_i = 1$ (pas d'estimation de la déviation standard), $\sigma_\phi = 0.5$, $\sigma_\alpha = \sigma_\rho = 0.1$.

Une fois l'algorithme CONDENSATION validé et ses paramètres définis, nous avons réalisé l'implantation de la méthode d'enchaînement des modèles et le remplacement de la commande utilisateur.

6.3.2 Enchaînement de modèles de trajets

Le but de cette étude est de réaliser une aide à la navigation en remplaçant les commandes utilisateurs par celles contenues dans le système. L'environnement est stocké par morceaux dans la mémoire mais il est nécessaire de lier les modèles entre eux afin d'obtenir une continuité de la reconnaissance. Afin de mettre en œuvre la méthode d'enchaînement de modèle décrite dans le § 6.2.2, nous utilisons deux modèles enchaînés manuellement dans l'environnement du simulateur. Ces modèles effectuent un trajet qui demande normalement une action de la part de l'utilisateur. Ce test a donc deux fonctions. La première consiste à prouver que le système est capable de suivre des changements de modèle selon une suite connue et définie dans le réseau de modèle et la seconde consiste à vérifier que le

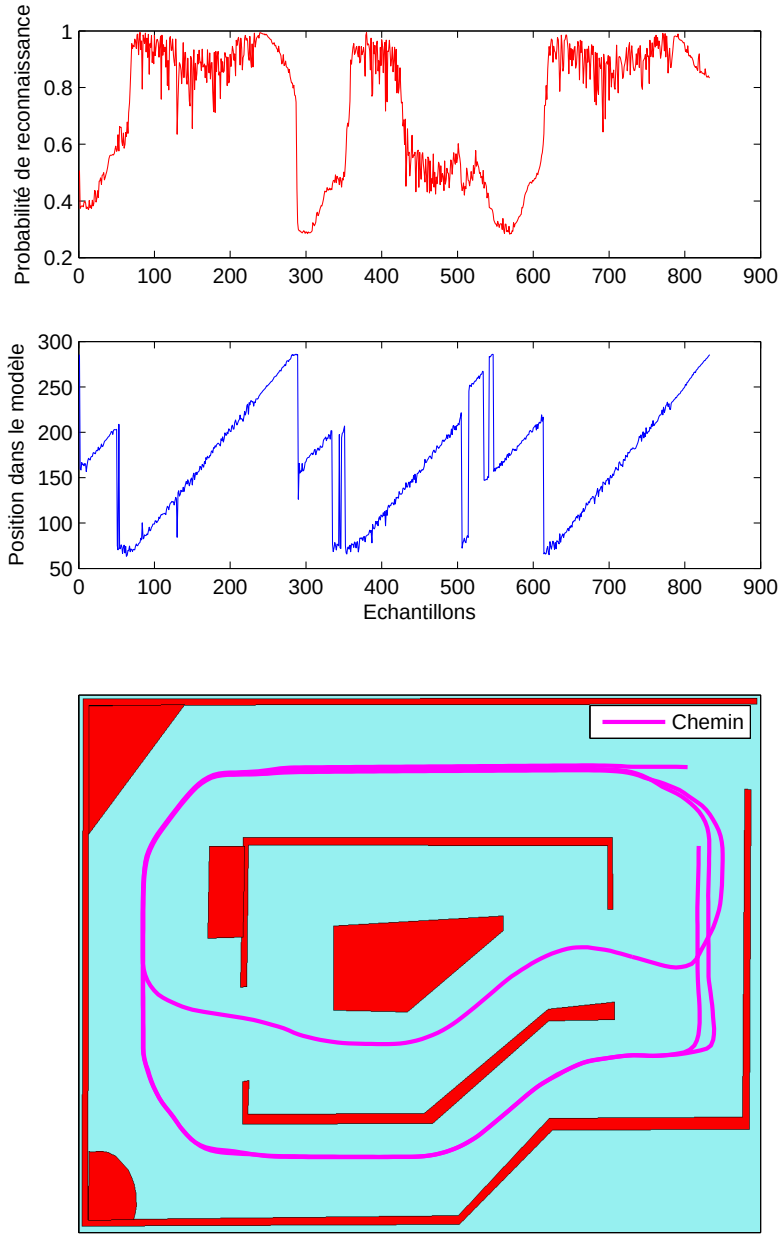


FIG. 6.7: Résultats de la reconnaissance en parcourant plusieurs fois l'environnement. On a réalisé 3 trajets dans l'environnement avec le modèle présenté dans la figure 6.6. Le premier est similaire au modèle, le second est différent et passe par le centre et le troisième est de nouveau similaire au modèle. La reconnaissance du premier trajet (échantillons de 0 à 300) est caractérisée par une probabilité proche de 1 et une progression linéaire de la position dans le modèle. Le second trajet est caractérisé par une chute de la probabilité lors du passage dans la zone centrale après le $400^{ième}$ échantillon. Le troisième trajet démarre après l'échantillon 600 avec une probabilité proche de 1 et une progression linéaire de la position dans le modèle.

système possède la capacité de suivre un changement de direction.

La méthode utilisée pour changer la direction du fauteuil est celle de la substitution de commande. La commande nécessaire à la réalisation du trajet est enregistrée dans le modèle et elle est appliquée en cas de reconnaissance de la position dans le modèle. L'indication de la reconnaissance est calculée à partir de la probabilité de reconnaissance et de la linéarité de la progression. Leurs paramètres ont été définis en testant différentes valeurs sous Matlab. On utilise pour cela :

- la moyenne sur les 7 derniers échantillons de la probabilité de reconnaissance (P_{moyen}), donnée par la formule 6.1 afin de lisser la courbe de probabilité et de pallier les zones locales de faible reconnaissance

$$p_{moyen} = \frac{\sum_{i=1}^{7} p_i}{7}. \tag{6.1}$$

- l'erreur de linéarité de la progression de la position ϕ dans le modèle sur les 7 derniers échantillons (L_ϕ). Cette erreur est calculée par la moyenne des erreurs de progression au carré par la formule

$$L_\phi = \frac{(\sum_{i=1}^{7} \phi_i - \phi_{i+1} - \rho_i)^2}{7}$$

  dans laquelle on calcule la différence entre les positions ϕ_i et ϕ_{i+1} de reconnaissance à laquelle on soustrait le pas de progression ρ. Ceci donne l'erreur de progression dans le modèle. Le calcul de la distance entre ϕ_i et ϕ_{i+1} tient compte du modèle dans lequel ils ont été pris afin de pallier les problèmes de changement de modèles.

La reconnaissance est validée lorsque :

- $P_{moyen} > 0.98$ et $L_\phi < 300$
- $P_{moyen} > 0.80$ et $L_\phi < 40$
- $P_{moyen} > 0.90$ et $L_\phi < 100$
- La reconnaissance à l'instant précédant était validée et $P_{moyen} > 0.70$ et $L_\phi < 200$

La dernière condition de validation réalise une hystérèse afin de garantir une plus grande stabilité face aux petites variations de l'environnement et de continuité de la validation de la reconnaissance. Sans cette validation de la reconnaissance, le fauteuil ne peut pas savoir quand appliquer les changements de direction. Les valeurs des conditions de validation de la reconnaissance ont été déterminées de manière empirique.

Les modèles utilisés ainsi que le trajet effectué sont présentés dans la

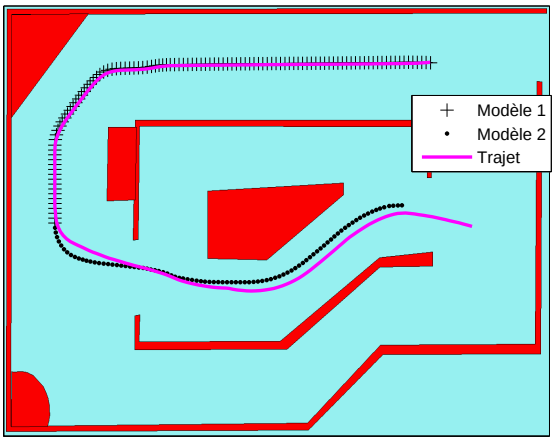
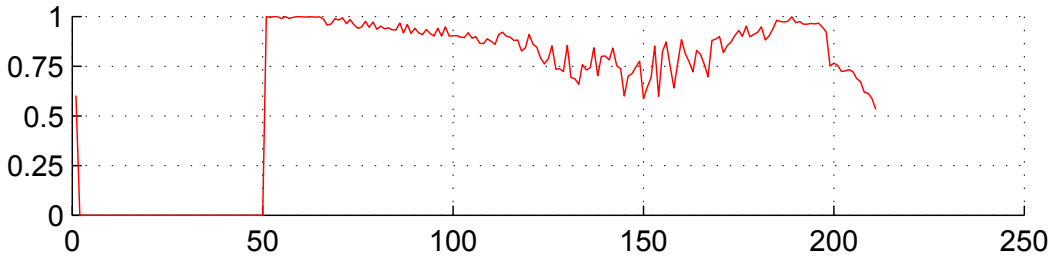


FIG. 6.8: Représentation des deux modèles et d'un trajet dans l'environnement. Le trajet est effectué sans définition de direction après le départ.

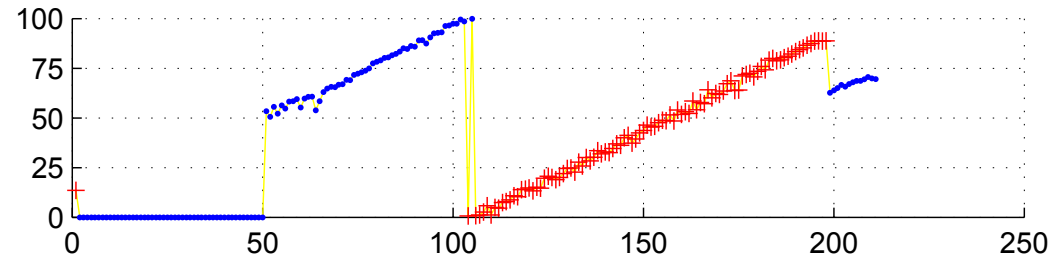
figure 6.8. On remarque que le fauteuil suit bien le changement de trajet alors que l'utilisateur n'est pas sollicité pour effectuer la définition de direction. En effet, le comportement par défaut du fauteuil est un suivi de direction « vers l'avant », comme le montre la figure 6.6 réalisée sans le suivi de trajet, et non de tourner. Les résultats présentés dans la figure 6.9 comprennent la probabilité de vraisemblance, la position dans le modèle ainsi que deux méthodes de la validation de la reconnaissance. On remarque que la validation basée uniquement sur la probabilité n'est pas constante car les trajets ne sont pas identiques alors que la validation utilisant la probabilité et la linéarité de progression valide la reconnaissance sur l'intégralité du trajet. Il est difficile de prouver que l'utilisateur n'a pas été sollicité car, l'échantillonnage s'effectuant sur la distance parcourue, il n'existe aucune trace des arrêts du fauteuil.

Il est intéressant de noter que lors de la transition entre les modèles, aucune perte de reconnaissance n'a lieu bien que l'état vainqueur soit alternativement dans la fin du modèle 1 et au début du modèle 2 comme décrit dans la figure 6.9(b) aux alentours du $110^{ième}$ échantillon. Ceci prouve la prise en compte du chaînage des modèles lors du calcul de la distance entre deux échantillons.

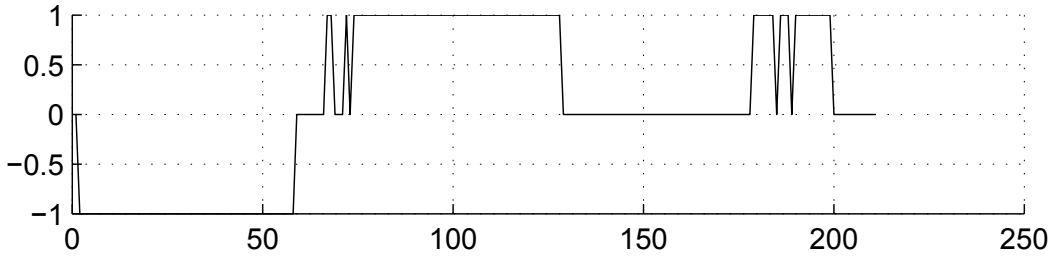
Ces différentes simulations nous permettent de valider l'approche proposée et de passer aux conditions réelles sur la fauteuil VAHM 3.



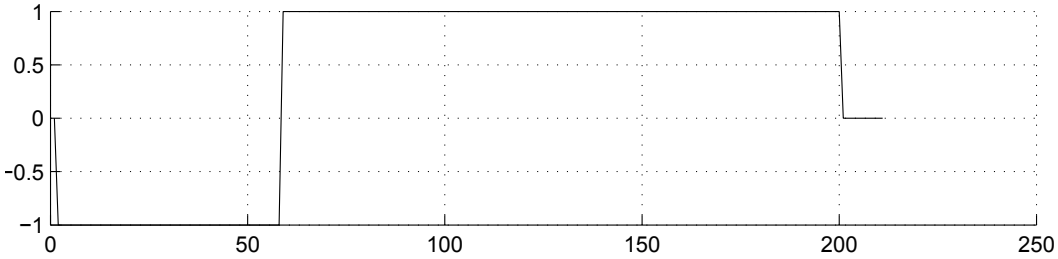
(a) Évolution de la probabilité de vraisemblance



(b) Évolution de la position dans les modèles. Le modèle 1 est reconnu de 50 á environ 110 et le modèle 2 de 110 à 200.



(c) Évolution du résultat de la reconnaissance en utilisant uniquement la probabilité de vraisemblance



(d) Évolution de la validation de la reconnaissance en utilisant la probabilité et la linéarité de la progression.

FIG. 6.9: Probabilité de vraisemblance, position dans le modèle et validation de reconnaissance lors de l'enchaînement de deux modèles. La direction n'est changée que lorsque la validation de la reconnaissance est à 1. Lorsque celle-ci est à -1 cela veut dire que la reconnaissance n'a pas été effectuée car il n'y a pas assez d'observations. Les différences entre les figures 6.9(c) et 6.9(d) montrent l'utilité du calcul de validation des résultats.

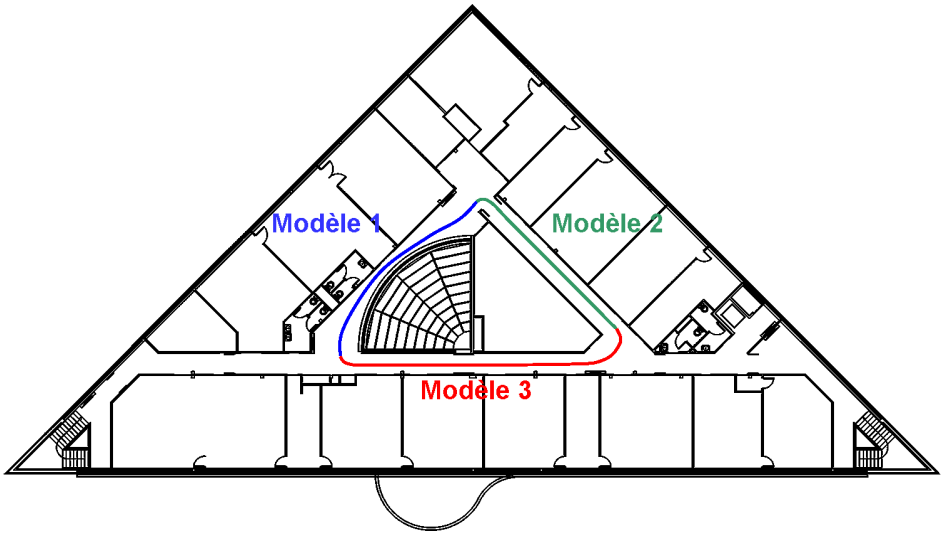


FIG. 6.10: Environnement utilisé dans pour les essais réels.

6.4 Tests en conditions réelles

Ces tests se sont déroulés en deux étapes. Dans un premier temps nous avons mis en oeuvre la reconnaissance de trajets en utilisant le joystick pour contrôler le fauteuil et non l'interface à défilement couplée au contacteur TOR. Ce type d'essai nous permet de vérifier et d'ajuster les paramètres de la reconnaissance par rapport à ceux du simulateur. Dans un second temps nous utilisons le système de commande du fauteuil pour générer les commandes de trajectoires et pour faire fonctionner l'aide à la planification de trajet.

L'environnement utilisé est celui du laboratoire. Ces essais consistent à se déplacer dans les couloirs. Ils se déroulent tous de la même manière :

- dans un premier temps l'environnement est appris en le parcourant une fois,
- puis la reconnaissance est utilisée lors des passages suivants.

La figure 6.10 montre le plan du bâtiment dans lequel les tests sont effectués.

6.4.1 Utilisation du joystick de commande

Lors de cette série d'essais nous utilisons le joystick de commande pour déplacer le fauteuil dans l'environnement. Pour cela nous lançons le programme de contrôle du VAHM en activant l'agent de reconnaissance de trajectoire. Plusieurs trajets sont réalisés en modifiant la taille de fenêtre variable et la puissance de discrimination de la probabilité de vraisemblance

(cf §5.4.2).

Ces essais montrent que des éléments comme des pieds de table ont pour effet de fournir un pic très localisé dans les valeurs des distances retournées par les capteurs. De plus un environnement complexe peut générer des zones dans lesquelles un des capteurs US n'a pas de signal de retour. Par exemple, cette erreur de mesure se produit généralement lors de la rotation du fauteuil VAHM, ce qui entraîne un angle d'incidence empêchant la mesure. Malheureusement, l'algorithme CONDENSATION n'est pas un algorithme exact et il existe un déphasage entre le modèle et les observations, de ce fait ces pics entraînent des différences importantes lors du calcul de vraisemblance. Pour limiter ce problème nous avons réalisé un lissage sur les observations et donc sur le modèle lors de son apprentissage.

La figure 6.11 représente les valeurs lissées et non lissées d'une même zone lors de différents passages. Elle montre aussi les moyennes des carrés de la différence entre deux courbes identiques de rétrécissement en fonction du déphasage. Ces valeurs sont calculées sur les données des rétrécissements avant et arrière, avec et sans lissage. On utilise pour cela une interpolation linéaire entre les valeurs, comme le fait l'algorithme CONDENSATION. Dans cette figure, on peut remarquer que la différence est bien plus grande sans lissage que avec lissage. De plus ce déphasage existe obligatoirement du fait du fonctionnement probabiliste de l'algorithme. Ceci tant à prouver que le lissage peut améliorer le fonctionnement du système.

On remarque que pour un filtrage moyenneur sur 5 éléments une amélioration de la reconnaissance est notable. Les modifications des autres paramètres n'apportent pas d'améliorations.

Une analyse des résultats montre que, lors de la « Condensation », le renouvellement des états (particules) ayant des valeurs de probabilité faible n'est pas assez important. Ils se propagent malgré une valeur moyenne. Le seuil de renouvellement a donc été modifié. De valeur initiale $\epsilon = 0.2$, il est augmenté à $\epsilon = 0.5$ afin de favoriser le renouvellement des états.

Ce changement a largement amélioré les résultats de reconnaissance. Les nouveaux paramètres de l'algorithme CONDENSATION sont alors :

- Taille de la fenêtre d'observations : $W = 60$
- Nombre d'états : $N = 800$
- Variation de l'amplitude : $\alpha = [0.9\ 1.1]$
- Variation de l'échantillonnage : $\beta = [0.9\ 1.1]$
- Seuil de la probabilité de vraisemblance avant renouvellement de l'état : $\epsilon = 0.5$ et exécution d'une prédiction avant le renouvllement.

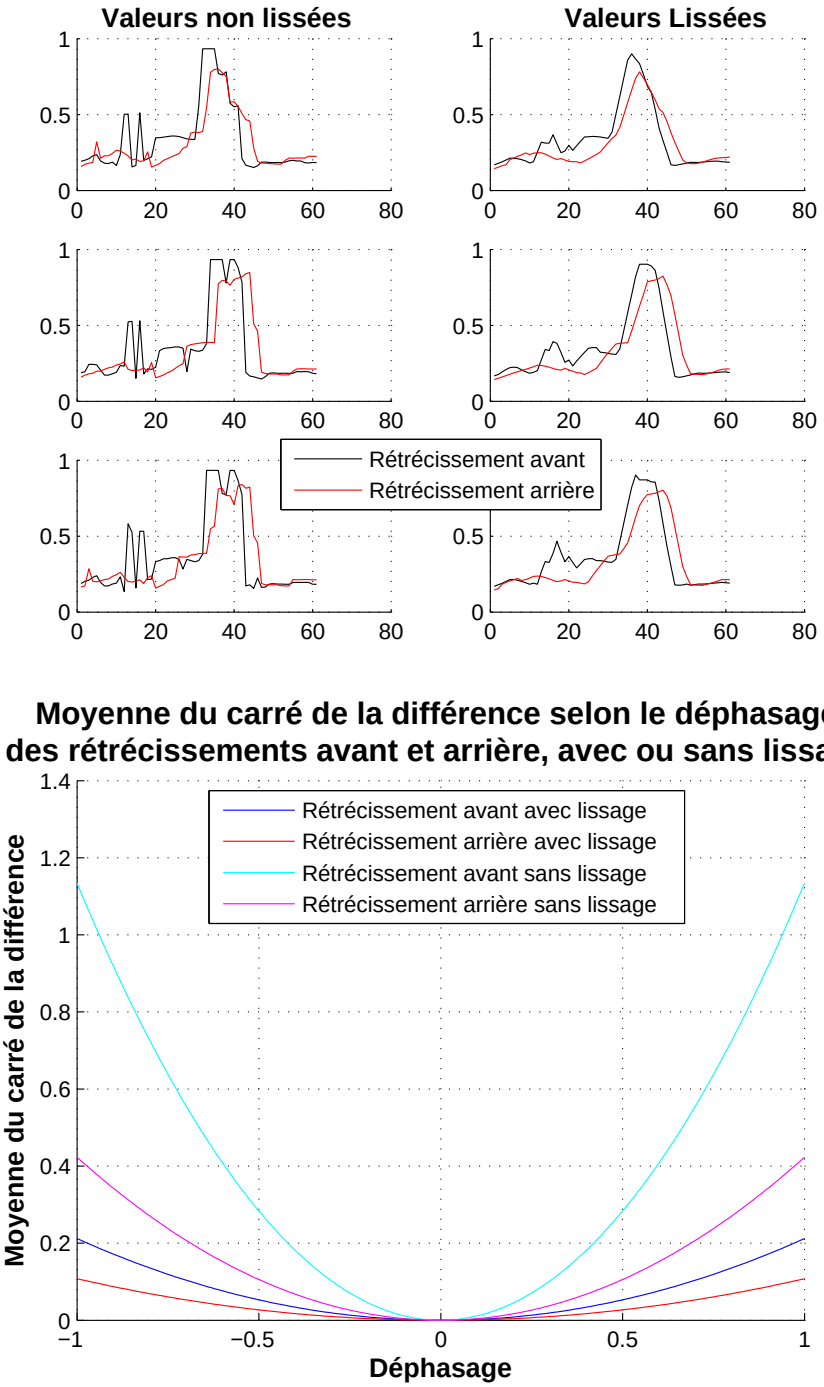


FIG. 6.11: Lissage des valeurs. Ces figures représentes le lissage des valeurs d'une même zone d'un trajet ainsi que la valeur de la différence entre un morceau de trajet sur une des zones montrées dans la figure du haut et ce même trajet mais déphasé. Les résultats montre que lors de déphasages, les différences sont plus faibles lorsque les données sont lissées.

- Utilisation de 10% de nouveaux états.
- Puissance utilisée pour la discrimination de la vraisemblance : $A = 16$
- Incertitudes de prédiction : $\delta_\rho = 0.02$, $\delta_\phi = 0.05$ et $\delta_\alpha = 0.02$

Les résultats de la reconnaissance sont décrits dans la figure 6.12. Cette figure présente les 3 modèles appris lors d'un premier passage dans l'environnement. Puis les résultats donnés par deux passages consécutifs dans ce même environnement. Elle montre que le fauteuil VAHM a réussi à se localiser tout au long des trois modèles appris. Il est à noter que la liaison entre le modèle 3 et le modèle 1 n'a pas été modélisée, ce qui explique que la reconnaissance échoue lors de cette transition. On remarque que le fauteuil est de nouveau localisé lorsque la fenêtre W se trouve complètement sur une zone du modèle 1.

6.4.2 Utilisation de l'interface TOR

L'utilisation de l'interface TOR avec la reconnaissance de trajet représente la finalité de notre travail. Ce test consiste à valider la capacité du système à apprendre un trajet, le reconnaître et appliquer les changements de direction contenus dans le modèle. La figure 6.13 montre les résultats de l'utilisation du système dans le même environnement réel. Dans ce cas, nous utilisons un seul modèle regroupant les 3 modèles présentés dans le paragraphe précédent, les changements de direction étant contenus dans le modèle. L'environnement est parcouru 3 fois : le premier trajet permet d'obtenir le modèle et les deux autres doivent reproduire le trajet sans que l'utilisateur n'ait à réaliser une commande de changement de direction. La figure 6.13 montre un déplacement automatique en reproduisant le modèle appris lors du premier passage sans action de l'utilisateur.

On remarque cependant qu'il existe des pertes de localisation. Elles sont dues principalement aux erreurs de mesure (technologies des capteurs US, glissement des roues sur le revêtement). Ces différences sont causées aussi par des problèmes liés aux comportements du fauteuil. Le problème vient du fait que le VAHM 3, sur lequel ont été réalisé ces essais, est un fauteuil à propulsion ce qui entraîne que le centre de rotation se retrouve très en arrière par rapport au VAHM 2 qui est à traction les différences de trajectoire sont présentées dans la figure 6.14. Ce problème implique une modification des paramètres des agents notamment lors des intersections de couloirs. Afin d'éviter cette lourde tâche, seuls les comportements d'évitement d'obstacles et de suivi de direction ont été utilisés, ce qui entraîne un fonc-

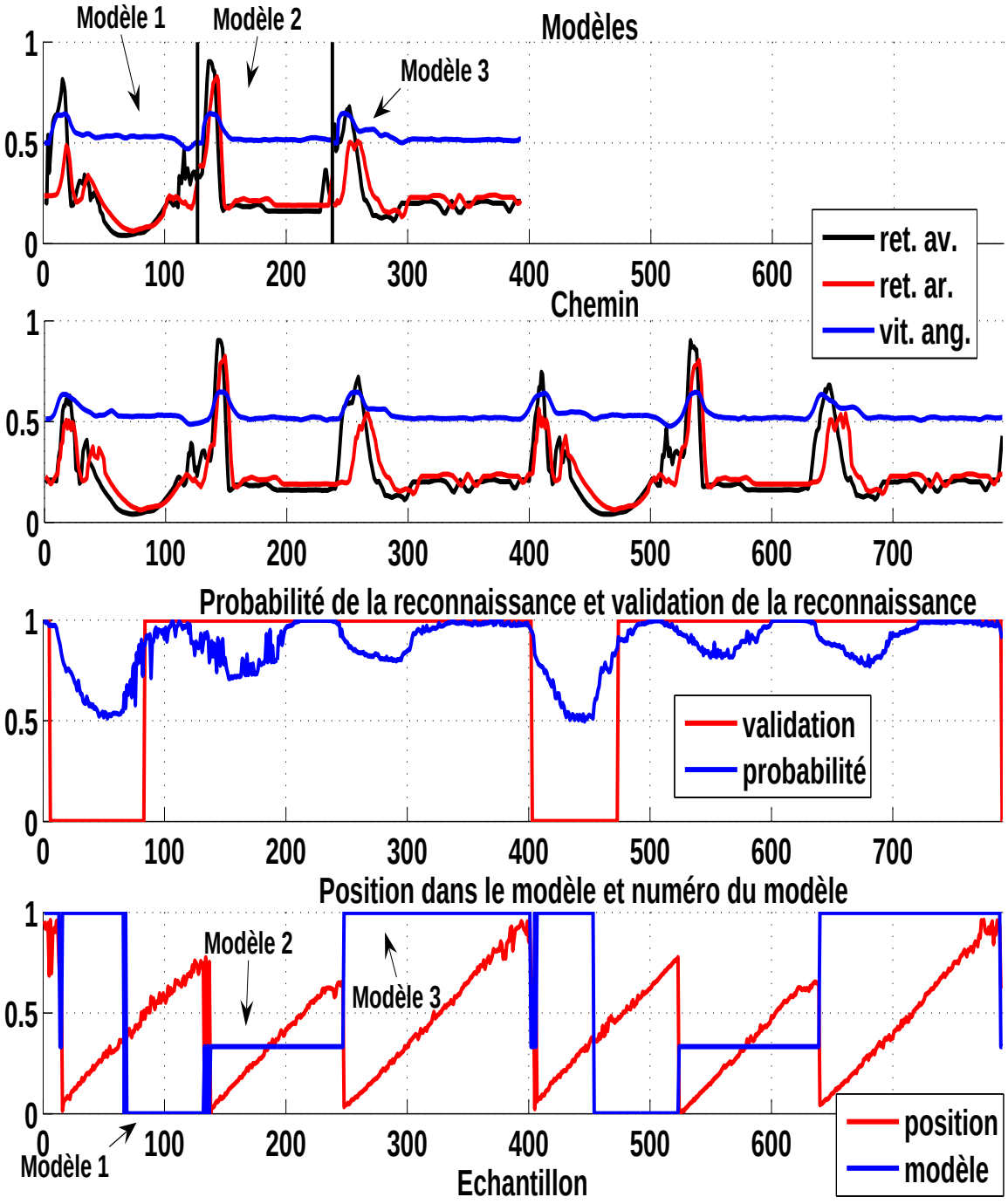


FIG. 6.12: Résultats des tests réels avec une commande du fauteuil par le joystick. Cette figure présente les 3 modèles, les trajets parcourus. Les résultats montrent que la reconnaissance est validée lorsque le fauteuil suit le même trajet que les modèles (avec enchaînement).

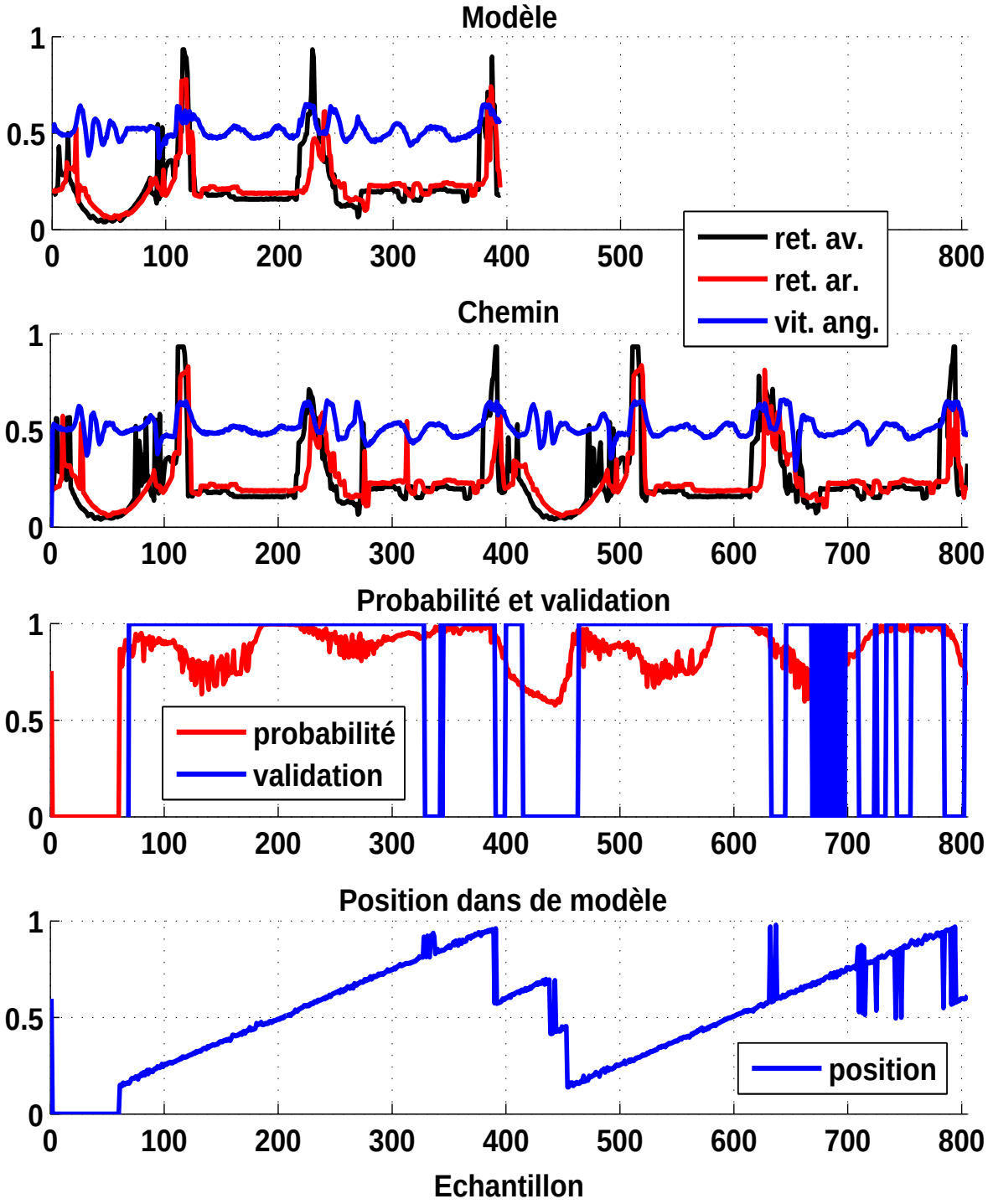


FIG. 6.13: Résultats des tests réels avec une commande par l'interface TOR. Cette figure présente le modèle, le chemin parcouru et les résultats. Le modèle représente un tour complet dans l'environnement et le trajet représente deux autres tours dans l'environnement. Le trajet complet est réalisé avec une seule commande de direction au départ. Le fauteuil réalise automatiquement les deux passages dans l'environnement.

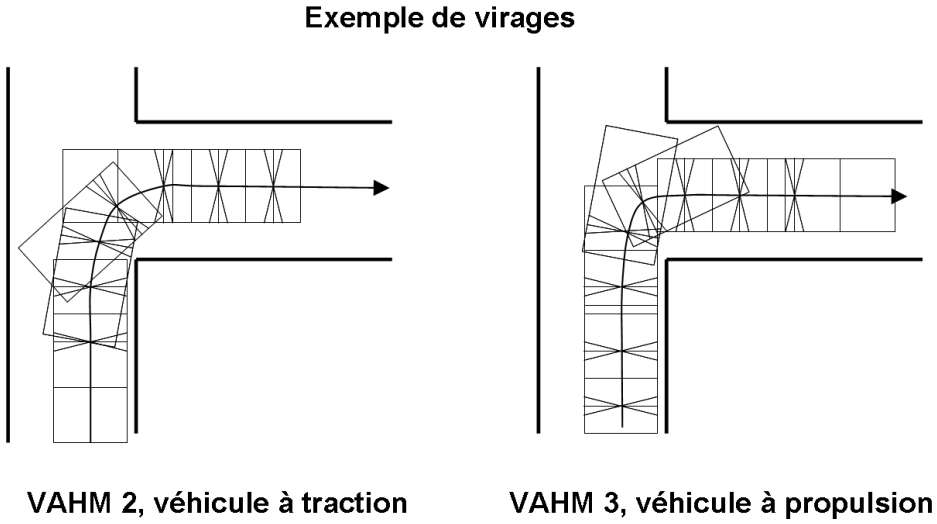


FIG. 6.14: Différences de trajectoire entre le fauteuil VAHM 2 qui est à traction et le VAHM 3 est à propulsion.

tionnement dégradé du fauteuil et des oscillations dans les couloirs. Mais le but étant ici uniquement de valider la possibilité de l'aide à la définition de trajet, ces problèmes ne sont pas trop dérangeants bien qu'ils entraînent des erreurs de localisation dans le second passage de test aux environs des échantillons 660 à 700 à cause d'une différence dans ces oscillations.

Ces problèmes nous ont également amené à modifier légèrement la base de cas en ajoutant la prise en compte de la différence entre les capteurs sur les côtés au milieu et à l'arrière afin de ne faire tourner le fauteuil seulement quand l'avant est suffisament dégagé des obstacles.

6.5 Conclusion sur la mise en oeuvre

L'agent de suivi de trajet est un agent particulier au sein du système VAHM. Il n'agit pas comme un agent d'interaction avec l'extérieur ou comme un agent de comportement réflexe. Il doit anticiper les actions de l'utilisateur et apprendre ses habitudes de déplacement. Cette fonctionnalité nécessite une méthode de localisation topologique afin de connaître la position du fauteuil. La méthode développée utilise la reconnaissance de trajet pour réaliser cette localisation. L'algorithme de reconnaissance utilisé est l'algorithme CONDENSATION. Il permet de mettre en correspondance un modèle avec les dernières observations provenant des capteurs installés sur le fauteuil. Les modèles sont placés dans un réseau Bayésien qui possède des similarités avec une chaîne de Markov. Chaque noeud représente un modèle

et la probabilité de transition d'un noeud à un autre permet de déterminer la suite la plus probable du déplacement.

La mise en oeuvre de cet agent au sein du système a été validée étape par étape afin d'affiner les différents paramètres de l'algorithme CONDENSATION mais également les seuils de détection permettant de certifier la reconnaissance. Ce seuil est défini par la valeur de la probabilité et la linéarité de la progression de la position dans le modèle. Ce système permet d'avoir une reconnaissance fiable et de pouvoir apprendre automatiquement les modèles dans le réseau.

6.6 Conclusion

Ce chapitre part d'un constat simple : « il n'est viable dans notre cas d'utiliser les MMC pour modéliser un trajet à partir de l'enchaînement des comportements mais que l'utilisation des données provenant des capteurs est bien plus pertinente ». Nous avons utilisé la mise en correspondance de courbes.

Nous avons donc commencé par utiliser l'algorithme des moindres carrés sur une fenêtre glissante pour essayer de réaliser un appariement entre des courbes d'un même type. Cette méthode nous a montré qu'il semblait possible de réaliser la localisation topologique à partir des courbes formées par l'évolution des données provenant des capteurs mais qu'un problème de déformation de ces courbes existe. Ces déformations proviennent des erreurs d'échantillonnage liées au glissement des roues sur le sol mais aussi des différences de trajet qui existent lorsque le fauteuil effectue des virages plus ou moins serrés (il ne passe pas toujours au même endroit).

Après avoir réalisé un choix rapide des algorithmes d'appariement de valeurs continues, nous nous sommes intéressés à l'algorithme CONDENSATION qui est issu des filtres particulaires.

L'intérêt de l'algorithme CONDENSATION pour réaliser la reconnaissance de trajets est de permettre d'intégrer plusieurs sources d'observations et également de connaître la position des observations dans le modèle. Alors que les MMC ne retournent que le modèle correspondant aux observations. De plus les résultats préliminaires ont prouvé qu'il était possible de réaliser une reconnaissance à partir d'un seul trajet par type de trajets. Les temps de calculs sont acceptables et fixes car ils sont proportionnels au nombre d'états utilisé.

Une fois cet algorithme validé, nous l'avons implanté dans le système

VAHM en ajoutant un nouvel agent appelé « *suivi de trajets* ». Cet agent a pour but d'agir comme l'utilisateur l'aurait fait en remplaçant sa commande par celle contenue dans le modèle du trajet. Les modèles sont intégrés dans un réseau permettant de les chaîner. Ceci permet, à la fin d'un trajet, de choisir automatiquement le trajet suivant le plus usité. L'utilisateur peut à tout moment interrompre le déplacement et sélectionner manuellement une autre direction. Il est également prévu qu'il soit prévenu à l'avance des changements de direction.

Les tests réalisés avec le simulateur du système VAHM montrent que le fauteuil peut suivre un trajet mais également enchaîner deux modèles de trajets. La validation de la reconnaissance est basée sur la probabilité de vraisemblance retournée par l'état vainqueur et la linéarité de la progression au sein du modèle.

7 Conclusion et perspective

7.1 Conclusion

LE projet présenté dans ce mémoire s'inscrit dans le cadre du projet de fauteuil intelligent VAHM qui a pour but de réaliser un véhicule pouvant être commandé à partir d'une interface TOR. Le but de ce projet est d'intégrer une méthode de localisation topologique (cf §2.4) afin de permettre au fauteuil, une fois localisé dans son environnement, de pouvoir assister l'utilisateur dans son déplacement. Cette aide doit permettre de limiter le nombre de commandes de direction nécessaire à la navigation. Car, en fonction des pathologies dont souffre l'utilisateur, il lui est parfois difficile et fatigant de générer ces commandes de direction.

L'originalité de ce travail réside dans la méthode de localisation topologique. L'idée de départ était de profiter du fait que le VAHM utilise un système multi-agents pour se déplacer et plus spécifiquement des agents comportementaux (évitement d'obstacles, suivi de mur droit ou gauche, suivi de direction et retour arrière). Les enchaînements de ces comportements lors de la réalisation de trajets peuvent être appris à l'aide de Modèles de Markov Cachés. Des tests ont été réalisés sur 24 trajets réels provenant de 4 types différents. Malheureusement les résultats montrent que cette méthode possède une efficacité limitée lorsqu'il s'agit de reconnaître des trajets n'ayant pas été appris (tests de généralisation). Ceci peut s'expliquer par la faible quantité de trajets disponibles pour l'apprentissage. Cependant il était important de savoir s'il était possible de généraliser à partir d'un faible nombre d'apprentissage car il n'est pas concevable que l'utilisateur doive effectuer un grand nombre de fois un parcours avant que celui-ci puisse être appris convenablement.

Ce travail a été étendu en intégrant d'autres données dans les modèles de trajet. Ces informations supplémentaires sont le rétrécissement et la vitesse angulaire. Ces données ont apporté une légère amélioration des résultats mais le nombre de calculs nécessaire à l'apprentissage a grandement augmenté les temps d'apprentissage. De plus cela a clairement démontré que

l'utilisation des comportements comme source d'observation n'est pas utile puisque les résultats sont proches voire meilleurs en n'utilisant que les données numériques provenant des capteurs. Cette dernière remarque nous a poussés à n'utiliser que les données numériques provenant des capteurs pour réaliser la reconnaissance. Nous avons alors abandonné les MMC au profit d'un système comparant directement les courbes des informations capteurs.

Cependant deux problèmes majeurs sont apparus. Le premier est les déformations des courbes dues aux glissements des roues et aux différences de trajets (virages plus ou moins serrés...). Le second vient du fait que l'on ne peut pas tester toutes les possibilités de position du fauteuil en condition et encore moins toutes les possibilités de déformation pour chaque position. La solution choisie utilise un outil provenant du filtrage particulaire : l'algorithme CONDENSATION. Cet algorithme regroupe en effet les solutions aux deux problèmes liés à la comparaison directe des courbes des données des capteurs.

L'algorithme CONDENSATION a été implémenté sous Matlab pour une première évaluation. Un premier test de validation de l'algorithme de classification, utilisant uniquement le premier chemin de chaque type, a prouvé non seulement la validité mais également son efficacité puisqu'il permet une classification sans erreur. Son intégration dans le système VAHM a alors été réalisée en C++ en ajoutant un agent spécifique qui réalise la reconnaissance de trajets. Cet agent utilise comme modèle le premier passage sur un trajet appris manuellement avec un découpage manuel. Lorsque cet agent a reconnu le trajet, il remplace la commande de l'utilisateur par celle qui a été enregistrée dans le modèle. L'utilisateur pouvant à tout moment stopper le fauteuil pour changer de direction. Les modèles sont enregistrés dans un réseau apparenté à une chaîne de Markov où chaque noeud correspond à un modèle et la transition d'un noeud à un autre, et donc d'un modèle à un autre, se fait par rapport à la probabilité de transition qui représente la transition la plus utilisée parmi les transitions possibles.

7.2 Perspectives

La suite de ce travail se situe principalement au niveau de la gestion du réseau de modèles et de l'apprentissage automatique des modèles. En effet, bien que le réseau de modèles existe et soit fonctionnel, son peuplement se fait à la main et aucun automatisme de gestion n'existe pour l'instant.

Un autre point important est la gestion du réseau de modèles car si l'on ajoute indéfiniment des modèles, il y a un risque d'alourdissement du système. Un troisième point discutable est celui de la gestion des probabilités de transition car il serait intéressant d'inclure des plages horaires où l'utilisateur prend un trajet plus qu'un autre : par exemple à l'heure du repas, il va plutôt aller dans la cuisine ou la salle à manger que dans la chambre. Des événements peuvent également influencer les déplacements comme le téléphone, ou quelqu'un qui sonne à la porte. Une autre partie de ce travail qui mérite un approfondissement est la gestion des passages d'un modèle à un autre et plus globalement de l'utilisation d'un nombre variable d'états dans l'algorithme CONDENSATION.

Apprentissage automatique

L'apprentissage automatique des modèles est une tâche complexe. En effet il n'est possible de considérer comme modèle qu'un trajet ayant un début et une fin. Ceci implique que pour apprendre un modèle il faut savoir, dans le réseau, où il commence et où il finit. Un nouveau modèle commence après une perte de reconnaissance et se finit à la reprise de la reconnaissance. Mais avant d'intégrer ce nouveau modèle, il faut répondre à plusieurs questions.

Le point de reprise de reconnaissance est-il bon ?

Il se peut en effet que le système n'ait pas reconnu immédiatement la position du fauteuil. Il convient donc de vérifier où le fauteuil a "rattrapé" le modèle. Une solution est de revenir en arrière depuis le point de reconnaissance en utilisant les paramètres de l'état ayant effectué la reconnaissance.

A t on affaire à un nouveau modèle ou est-ce juste une perte locale de reconnaissance ?

Il se peut en effet que la modification de l'environnement, comme le déplacement d'un meuble par exemple, ait engendré une perte locale de reconnaissance. La taille de la perte de reconnaissance ainsi que ses positions de début et de fin sont importantes. En effet nous n'avons pas de problème dans le cas d'une perte de reconnaissance partant et arrivant dans le même modèle, avec une différence de position dans le modèle approximativement égale à la taille de la perte de reconnaissance. Cependant, le cas où celle-ci

part et arrive dans des modèles différents devient plus particulier car il faut vérifier si cette perte n'a pas eu lieu sur une transition existante ou non entre ces modèles.

Comment traiter une perte locale de reconnaissance ?

Il est en effet difficile de savoir si la différence entre le modèle et les observations vient d'une situation temporaire (présence de personne ou d'objets présents temporairement) ou d'une situation permanente (ajout, déplacement ou suppression d'un meuble). Et en cas de situation temporaire, vient-elle du modèle ou des observations ? Plusieurs solutions existent comme par exemple le remplacement des valeurs du modèle par celles des observations. Cette solution ne traite cependant pas les situations temporaires si elle est effectuée à chaque fois qu'une erreur survient . Mais comment intégrer le déphasage entre les observations et le modèle ainsi que les déformations qui ont été utilisées lors de la reconnaissance ?

Une autre solution d'intégration des valeurs est de réaliser un méta-modèle en combinant le modèle existant, les observations, le déphasage et les déformations locales. Mais le problème de la stabilité des modèles sur le moyen et long terme se pose. En effet si le problème est occasionnel mais récurrent, il est possible que cette méthode se stabilise sur une moyenne qui rendra la détection défectueuse dans les deux cas de présence ou non de l'objet.

Comment intégrer un nouveau modèle ?

L'intégration d'un nouveau modèle dépend de la position où il s'intègre parmi les modèles existants. On peut distinguer 4 cas différents pour les points d'entrée ou de sortie. Ces points sont au début d'un modèle, à la fin d'un modèle, dans le reste du modèle ou dans une transition. Les actions à réaliser ne sont cependant pas les mêmes s'il s'agit du point d'entrée ou de sortie du nouveau modèle. La taille de ce nouveau modèle est également importante car, si celui-ci est petit, il peut s'agir d'une transition.

Les solutions sont nombreuses et certains paramètres sont encore à définir comme la zone correspondant au début et à la fin des modèles ou encore à partir de quelle taille définit on un modèle comme suffisamment petit pour être considéré comme une transition. Les tableaux 7.1 et 7.2 définissent, respectivement pour des petits et des grands modèles, les actions à mener en fonction des positions de perte et de reprise de la reconnaissance

dans les modèles qui composent le réseau.

Le nombre de possibilités lors de l'apprentissage rend ce dernier difficile et onéreux en terme de temps de calcul. De plus, il requiert souvent de connaître les noeuds et transitions, suivants et précédents, utilisés lors du déplacement. De ce fait il est préférable de réaliser cette tâche lorsque le VAHM est à l'arrêt car il est peu sollicité.

Les méthodes que nous venons de décrire ne font qu'ajouter des modèles dans le réseau mais il serait profitable de pouvoir également retirer des modèles qui ne sont plus utilisés.

Gestion du réseau de modèles

Lors de l'ajout des modèles il arrive que des morceaux de modèles soient déplacés dans les transitions mais rien n'est prévu pour les cas où celles-ci deviennent trop grandes et doivent donc être considérées comme des noeuds (modèles). Une autre fonction devrait être présente dans la gestion du réseau de modèles pour permettre de supprimer les modèles après un certain temps d'inutilisation.

Gestion des probabilités de transition

Actuellement la gestion des probabilités de transition se fait en fonction du nombre de passages dans les transitions. La transition la plus probable parmi les transitions possibles à la sortie d'un noeud est celle qui est la plus utilisée (compteur le plus grand). Cependant l'heure du déplacement peut être un critère supplémentaire dans le choix des directions. En effet aux heures du repas, l'utilisateur aura plutôt tendance à se déplacer vers le lieu où il prend son repas même s'il emprunte des directions qui ne sont pas celles qui sont le plus souvent utilisées. De même, un événement extérieur comme le téléphone où quelqu'un qui sonne à la porte peut grandement influencer les directions prises par l'utilisateur. Il est alors possible d'ajouter un apprentissage de l'heure ou d'événements extérieurs dans les transitions et d'adapter la fonction de calcul de probabilités en conséquence.

Un autre aspect dans la gestion des probabilités de transitions est d'utiliser une mémoire de la provenance sur plusieurs modèles consécutifs au lieu de n'utiliser que le modèle précédant. Ceci permettrait d'avoir une meilleure mémoire de la provenance afin d'affiner la sélection des enchaînements des modèles. Cette mémoire peut, par exemple, être réalisée en comptabilisant,

Si la position	du début de la perte de reconnaissance est	de la fin de la perte de reconnaissance est
au début d'un modèle existant.	Ce cas est ambigu. Normalement on ajoute au début du nouveau modèle, le début du modèle existant ainsi que la transition parcourue auparavant. On considère également comme provenance le modèle que l'on a quitté. Cependant en ajoutant toutes ces valeurs, le modèle peut ne plus être considéré comme petit.	On déplace les valeurs du modèle existant dans toutes transitions précédentes et on considère le modèle existant comme destination.
au milieu d'un modèle existant.	Découpage du modèle existant. La première partie de l'ancien modèle est considérée comme la provenance de la nouvelle transition.	Découpage du modèle existant. La seconde partie de l'ancien modèle est considérée comme la destination de la nouvelle transition.
à la fin d'un modèle existant.	On déplace les valeurs du modèle existant au début de toutes les transitions le suivant et on considère le modèle existant comme provenance de la transition.	Ce cas est ambigu. Normalement on ajoute à la fin du nouveau modèle, la fin du modèle existant et la transition utilisée après celui-ci . La destination de cette transition est alors considérée comme destination pour la nouvelle transition. Cependant en ajoutant toutes ces valeurs, le modèle peut ne plus être considéré comme petit.
dans une transition existante.	Ajout des données du début de la transition au début du modèle. Le modèle précédant de la transition existante est considéré comme la provenance de la nouvelle transition.	Ajout des données de la fin de la transition à la fin du modèle. Le modèle suivant la transition existante est considéré comme la destination de la nouvelle transition.

TAB. 7.1: Liste des modifications du réseau pour un petit modèle en fonction des points d'entrée et de sortie. Les petits modèles sont considérés comme des transitions, ils servent donc de liaison entre deux noeuds du réseau.

Si la position	du début de la perte de reconnaissance est	de la fin de la perte de reconnaisance est
au début d'un modèle existant.	On déplace les valeurs du début du modèle existant dans toutes les transitions qui le précède et on copie les valeurs et la provenance de la transition parcourue avant d'arriver dans le modèle existant (avec les valeurs qui étaient au début du modèle) dans la nouvelle transition de provenance.	On déplace les données du début du modèle existant dans toutes les transitions qui y mènent et on ajoute une transition vide du nouveau modèle vers le modèle existant.
au milieu d'un modèle existant.	On découpe le modèle existant et on ajoute une transition vide de la première partie du modèle existant vers le nouveau modèle.	On découpe le modèle existant et on ajoute une transistion vide du nouveau modèle vers la deuxième partie du modèle existant.
à la fin d'un modèle existant.	On déplace les valeurs de la fin du modèle vers toutes les transitions suivantes et on ajoute une transition vide provenant de l'ancien modèle et allant vers le nouveau modèle.	On déplace les valeurs de la fin du modèle vers toutes les transitions suivantes et on copie les valeurs et la destination de la transition parcourue après le modèle existant dans la nouvelle transition de destination
dans une transition existante.	On copie la provenance et les valeurs du début de la transistion existante dans la nouvelle transition de provenance.	On copie la destination et la fin des valeurs de la transition dans la nouvelle transition de destination

TAB. 7.2: Liste des modifications du réseau pour un grand modèle en fonction des points d'entrée et de sortie. Dans tous les cas on crée une transition avant le nouveau modèle ayant pour destination le nouveau modèle et une transition après le nouveau modèle ayant ce dernier comme source afin que ce nouveau modèle soit intégré comme un noeud du réseau.

à chaque passage dans une transition, la séquence des X derniers modèles parcourus. Ainsi en fonction des derniers modèles parcourus, nous pouvons affiner la prédiction de l'enchaînement des modèles.

Gestion de CONDENSATION

Actuellement le nombre d'états dans CONDENSATION est fixe. L'ajout d'une méthode permettant de faire varier le nombre d'états permettrait de limiter les temps de calcul lorsque la reconnaissance fonctionne. Cette méthode permettrait également de pouvoir suivre sans difficulté toutes les transitions. En effet actuellement seule la transition la plus probable est sélectionnée lors de l'étape de prédiction. Par exemple, si l'utilisateur choisit une autre direction, la reconnaissance échoue jusqu'à ce que la fenêtre de mise en correspondance passe ce changement de direction forcé. Ce problème peut également être résolu en propageant des états dans toutes les possibilités de transition à chaque fois que l'on arrive à la fin d'un modèle. Ceci entraînera alors qu'il y aura moins d'états disponibles pour les autres propagations. L'impact d'un tel système sur l'efficacité devrait cependant être limité car le nombre de possibilités à la fin d'un modèle est relativement faible. Il est effectivement rare de voir plus de 3 ou 4 possibilités de direction comme c'est le cas lors de l'intersection de deux couloirs.

L'objectif de ce travail, qui était de réaliser une reconnaissance de trajet sur un fauteuil roulant dans le but de réaliser une aide à la navigation pour les personnes handicapées, est atteint même s'il reste encore des détails à régler pour pouvoir l'intégrer convenablement dans le projet VAHM.

Bibliographie

[1] Y. Adachi, Y. Kuno, N. Shimada, and Y. Shirai. Intelligent wheelchair using visual information on human faces. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robot an Systems (IROS)*, pages 354–359, Victoria, BC, Canada, 13 1998. IEEE.

[2] P. Althaus and E. Christensen. Automatic map acquisition for navigation in domestic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1551–1556, 2003.

[3] Geovany Araujo Broges. *Cartographie de l'environnement et localisation robuste pour la navigation de robots mobiles*. Spécialité : Génie informatique, automatique et traitement du signal, Université de Montpellier II, LIRMM - Université de Montpellier II, 27 2002.

[4] K. Arras, N. Tomaris, B. Jensen, and R. Siegwart. Multisensor on-the-fly localization : Precision and reliability for applications. *Robotics and Autonomous Systems*, 34 :131–143, 2001.

[5] O. Aycard, F. Charpillet, D. Fohr, and J-F Mari. Place Learning and Recognition using Hidden Markov Models. In *Proceedings if the IEEE / IROS*, pages 1741–1746, 1997.

[6] A.C. Balcells and J.A. Gonzalez. TetraNauta : A Wheelchair Controller for User whith Very Severe Mobility Restrictions. In *Proceedings of the CSUN (California State University, Northridge) - Technology and Persons with Disabilites Conference*, Los Angeles, US, mar 1998.

[7] B. Barshan and H.F. Durrant-Whyte. Inertial navigation systems for mobile robots. In *IEEE transation on Robotics and Automation*, volume 11 of *328-342*, jun 1995.

[8] P. Beattie. SENARIO : SENsor Aided intelligent wheelchaiR navigatiOn. In *IEE Colloquium on New Developpements in Electric Vehicules for Disabled Persons*, pages 2/1–2/4, London, UK, 17 1995.

[9] D.A. Bell, J. Borenstein, S.P. Levine, Y. Koren, and L. Jaros. An assis-

tive navigation system for wheelchair based upon mobile robot obstacle avoidance. *IEEE Robotics and Automation*, may 1994. San Diego.

[10] M. J. Black and A. D. Jepson. A probabilistic framework for matching temporal trajectories : Condensation-based recognition of gesture and expressions. In H. Burkhardt and B. Neumann, editors, *Europran Conf. on Computer Vision, ECCV-98*, pages 909–924, Freiburg, Germany, 1998. Springer-Verlag.

[11] G.A. Borges and M.-J. Aldon. Optimal mobile robot pose estimation using geometrical maps. *IEEE Transaction on Robotics and Automation*, 18(1) :87–94, feb 2002.

[12] V. Borgolte, H. Hoyer, C. Buhler, C. Heck, and R. Hoelper. Architectural concept of semi-autonomous wheelchair. *Journal of Intelligent and Robotics System*, (22) :233–253, 1998.

[13] Guy Bourhis, Odile Horn, Olivier Habert, and Alain Pruski. The VAHM project : autonomous vehicle for people with motor disabilities. *IEEE Robotics and Automation Magaazine, Special Issue on 'Research on Autonomous Robotics Wheelchair in Europe'*, 7(1), mar 2001.

[14] Guy Bourhis and Pierre Pino. Mobile robotic and mobility assistance for people with motor impairments : rationnal justification of the VAHM project. *IEEE Transactions on Rehabilitation Engineering*, 4(1) :7–12, 1996.

[15] E.S. Boy, C.L. Teo, and E. Burdet. Collaborative Wheelchair Assistant. In *IEEE/RJS International Conference on Robotics and Intelligent Systems (IROS)*, pages 1511–1516, Lausanne, Suisse, 30 2002.

[16] R. Brooks. How to build complete creatures rather than isolated cognitive simulators. 1991.

[17] T. Brouard, M. Slimane, and J.P. Asselin de Beauville. Modélisastion de processus simultanés indépendants par haînes de Markov cachés multidimentionnelles (CMC-MD/I). Rapport interne, Laboratoire d'Informatique, E3I, Tours, 16 1997.

[18] Thierry Brouard. *Hybridation génétique de Chaînes de Markov cachées : conception d'algorithmes d'apprentissage et applications*. PhD thesis, Université de Tours, 1999.

[19] Markus Bushberger, Klaus-Werner Jörg, and Ewald von Puttkamere. Laserradar and Sonar Based World Modeling and Motion Control for Fast Obstacle Avoidance of the Autonomous Mobile Robot MOBOT-IV.

In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, volume 1, pages 534–540, Atlanta, Gerogia, 2 1993.

[20] CALL Center. Learning through Smart Wheelchair. Technical report, CALL Center, University of Edinburgh, may 1994.

[21] D. W. Cho. Certainty grid representation for robot navigation by a Bayesian method. *Robotica*, 8 :159–165, 1990.

[22] K. S. Chong and L. Kleeman. Sensor based map building for a mobile robot. In *Proceeding of the 1997 IEEE Interenational Conference on Robotics and Automation*, pages 1700–1705, Albuquerque, New Mexico, apr 1997.

[23] H. Chung, L. Ojeda, and J. Borenstein. Accurate mobile robot dead-reckoning with a precisioncalibrated fiberoptic gyroscope, 2001.

[24] J. Connell and P. Viola. Cooperative control of a semi-autonomous mobile robot, 1990.

[25] J.J. Cox. Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transaction on Robotics and Automation*, 7 :193–204, 1991.

[26] J. L. Crowley. World modeling and position estimation for a mobile robot usig ultrasonic ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 674–680, 1989.

[27] F. Dellaert, W. Burgar, D. Fox, and S. Thrun. Using the Condensation algorithm for robust, vision-based mobile robot localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, 1999.

[28] Michael Drumheller. Mobile robot localization using sonar. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 9, pages 325–332. IEEE Computer Society, mar 1987.

[29] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, New York, NY, USA, 2000.

[30] H. Everett. *Sensors for Mobile Robots : Theory and application*. 1995.

[31] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots : : I. a review of localization strategies. *Cognitive Systems Research*, 4(4) :243–282, December 2003.

[32] A. M. Flynn. Combining sonar and infrared sensors for mobile robot navigation. *Int. J. Rob. Res.*, 7(6) :54–64, 1988.

[33] D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11 :391–427, Novembre 1999.

[34] U. Frese, P. Larsson, and T. Duckett. A Multigrid Algorithm for Simultaneous Localization and Mapping. *IEEE Transactions on Robotics*, 21(2) :1–12, 2005.

[35] P. Goel, S. Roumeliotis, and G. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1134–1140, oct 1999.

[36] T. Gomi and A. Griffith. *Developping intelligent wheelchairs for the handicaped*, volume 1458. Springer Verlag, 1998.

[37] Guillermo Del Castillo Del Riego. *Autonaumous, vision-based, pivoting wheelchair with obstacle detection capabitility*. PhD thesis, University of Notre Dame, mar 2004.

[38] O. Habert and A. Pruski. Raisonnement à partir de cas pour le contrôle d'un fauteuil électrique. *JESA*, 34(6–7), septembre 2000.

[39] O. Habert and A. Pruski. Raisonnement à partir de cas pour le contrôle d'un fauteuil électrique. *Handicap 2000, Paris*, 2000.

[40] P. Hoppen, T. Knieriemen, and E. von Puttkamer. Laser-radar based mapping and navigation for an autonomous mobilerobot. In *Proceedings of the 1990 IEEE International Conference on Robotics ans Automation.*, volume 2, pages 948–953, Cincinnati, OH, USA, 13 1990.

[41] H. Hoyer, U. Borgolte, and A. Jocheim. The OMNI-Wheelchair - State of the art -. In *Proceedings of the 14th Annual California State University Northridge (CSNU) Conference on Technoligiy and Persons With Disabilities [Online]*, 1999.

[42] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Eur. Conf. on Computer VisionECCV (1)*, pages 343–356, 1996.

[43] Michael Isard and Andrew Blake. A mixed-state CONDENSATION tracker with automatic model-switching. In *ICCV*, pages 107–112, 1998.

[44] P. Jensfelt, Olle Wijk, D.J. Austin, and M. Andersson. Experiments on augmenting Condensation for mobile robot localization. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 2518–2524, San Francisco, CA, apr 2000.

[45] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME - Journal of the Basic Engineering*, 82 (Series D) :35–45, March 1960.

[46] M. Kass, A. Witkins, and D. Terzopoulos. Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, jan 1988.

[47] M. Kreutner and O. Horn. Co-operation between ultrasound and monocular vision for the localization of a mobile robot. In *Proceedings of the Computing Enginering in Systems Applications*, Ecole Centrale, Université de Lille, France, jul 2003.

[48] Michel Kreutner. *Perception Multisensorielle Pour le Positionnement d'un Véhicule Autonome Dédié aux Personnes Handicapées Moteurs*. Spécialité : Automatique, Université de eMetz, LASC, Université de Metz - Ile du Saulcy - 57000 Metz, 21 2004.

[49] A. Lankenau, T. Röfer, and B. Krieg-Brückner. Self-localization in large-scale environments for the bremen autonomous wheelchair. In C. Freksa, W. Brauer, C. Habel, and K. F. Wender, editors, *Spatial Cognition III*, number 2685 in Lecture Notes in Artificial Intelligence, pages 34–61. Springer ; http ://www.springer.de/, 2003.

[50] Wang-Heun Lee, Kyoung-Sig Roh, and In-So Kweon. Self-localization of a mobile robot without camera calibration using projective invariants. In *Pattern Recognition Letter*, volume 21, pages 45–60, 2000.

[51] J.J. Leonard and F.D. Durrant-whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transation on Robotics and Automation*, 7(3) :376–382, jun 1991.

[52] John J. Leonard and Juan Domingo Tardòs. Mobile navigation and mapping. In *IEEE International Conference on Robotics and Automation : Workshop on mobile robot navigation and mapping*, apr 2000.

[53] S.P. Levine, D.A. Bell, A. Jaros, R.C. Simpsons, Y. Koren, and J. Borenstein. The navchair assistive wheelchair navigation system. *IEEE Transactions on Rehabilitation Engineering*, 7(4), december 1999.

[54] Edmund F. LoPresti, Richard C. Simpson, David Miller, and Illah Nourbakhsh. Evaluation of sensors for a smart wheelchair.

[55] P. Mallet and G. Schöner. WAD Projetc where Attractor Dynamics aids wheelchair Navigation. In *Proceedings of the 2002 IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, pages 690–695, Lausanne, Suisse, oct 2002. IEEE.

[56] Bruno Marhic, El Mustapha Mouaddib, David Fofi, and Eric Brassart. Localisation absolue par capteur omnidirectionnel SYCLOP. *Traitement du Signal*, 17(3-NS) :195–206, 2000.

[57] Y. Matsumoto, T. Ino, and T. Ogsawara. Development of Intelligent wheelchair system with face and gazebased interface. In *Proceedings of the 10th IEEE International Workshop on Robot an Human Intereactive Communication 2001*, pages 262–267, Bordeaux, France, 2001. IEEE.

[58] M. Mazo. An integral system for assisted mobility [automated wheelchair]. *Robotics and Automation Magazine, IEEE*, 8(1) :46–56, mar 2001.

[59] G. Ménier and F. Poirier. Système adaptatif de prédiction de texte.

[60] Jean-Arcady Meyer and David Filliat. Map-based navigation in mobile robots : : Ii. a review of map-learning and path-planning strategies. *Cognitive Systems Research*, 4(4) :283–317, December 2003.

[61] D.P. Miller and M.G. Slack. Increasing access with a low-cost robotic wheeelchair. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World'*, volume 3, pages 1663–1667, Munich, Allemagne, 12 1994.

[62] Hans P Moravec and Albreto Elfes. High Resolution Maps from Wide Angle Sonar. In *Proceedings of the International Conference on Robotics and Automation*, pages 116–121, 1985.

[63] Y. Morère and A. Pruski. Aided navigation for disabled people : Route recognition with hmms - first results. In *7th Conf. for the Advancement of Assistive Technology, AAATE 2003*, Dublin, 1-3 sept 2003 2003.

[64] R. Müller, T. Röfer, A. Lankenau, A. Musto, K. Stein, and A. Eisenkolb. Coarse Qualitative Descriptions in Robot Navigation. In C. Habel C. Freksa, W. Brauer, editor, *Spatial Cognition II*, number 1849 in Lecture Notes in Artificial Intelligence, pages 265–276. Springer, 2000.

[65] P. Nickson. Solid-state tachometry. *Sensors*, pages 23–26, apr 1985.

[66] P.D. Nisbet, I.R. Loudon, and J.P. Odor. The call centre smart wheelchair. In *Proceedings of the 1st International Workshop on Robotic Applications to Medical and Health Care*, pages 9.1–9.10, Ottawa, 1988.

[67] G. Oriolo, G. Ulivi, and M. Vendittelli. Fuzzy maps : a new tool for mobile robot perception and planning. *J. of Robotic Systems*, 14(3) :179–197, 1997.

[68] S. Patel, S.H. Jung, J.P. Ostrowski, R. Rao, and C.J. Taylor. Sensor bseed door navigation for a nonholomic vehicle. In *Proceedings of the 2002 IEEE Intenational Conference on Robotics and Automation (ICRA)*, pages 3081–3086, Washington DC, may 2002. IEEE.

[69] M. Piasecki. Global localization for mobile robots by multiple hypothesis tracking. *Robotics ans Autonomous Systems*, 16(1) :93–104, 11 1995.

[70] J. Picone. Continuous Speech Recognition using HMMs. *IEEE ASSP Magazine*, pages 1–16, Juillet 1990.

[71] G. Pires, R. Aradjo, U. Nunes, and A.T. De Almeida. Robchair - A powered wheelchair using a behaviour-based navigation. In *5th International Workshop on Advenced Motion Control (ACM'98-Coimba)*, pages 536–541, Coimbra, Portugal, 29 1998. IEEE.

[72] Polaroid. Polaroid ultrasonic ranging system user's manual, 1991.

[73] E. Prassler, J. Scholz, and P. Fiorini. A robotic wheelchair roaming in a railway station, 1999.

[74] A. Pruski, M. Ennaji, and Y. Morère. Vahm : A user adapted intelligent wheelchair. In *IEEE Conference on Control Application*, Galsgow, Scotland, UK, September 2002.

[75] A. Pruski and O. Habert. Obstacle avoidance module for the vahm-2 wheelchair. In *5^{th} conference for the advanced of assistive technology*, Düsseldorf, Germany, november 1999. AAATE 1999.

[76] Alain Pruski. *Robotique Mobile, Planification de trajectoire*. Hermes, 1996.

[77] Alain Pruski, Mourad Ennaji, and Yann Morère. VAHM : Un fauteuil intelligent adapté à l'utilisateur. In *Compte-rendu de la Conférences Handicap 2002*, Paris, 13-14 Juin 2002. Handicap 2002.

[78] Alain Pruski, Olivier Habert, and Mourad Ennaji. Symbiotic Man-machine Architecture for a Smart Wheelchair Control. In *6th Conf. for the Advancement of Assistive Technology, AAATE 2001*, pages 216–221, Ljubljana,Slovenia, 6 September 2001.

[79] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE*, 77(2) :257–286, February 1989.

[80] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3 :4–16, 1986.

[81] R. Lawrence Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *IEEE*, pages 257–286, février 1989.

[82] R.S. Rao, K. Conn, S.H. Jung, J. Katupitiya, T. Kientz, V. Kumar, J. Ostrowski, S Patel, and C.J. Taylor. Human Robot Intercation : Application to Smart Wheelchairs. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 3583–3588, Wahington DC., may 2002.

[83] U. Raschke and J. Borenstein. A comparison of grid-type map-building techniques by index of performance. In *Proceeedings of the International Conference on Robotics and Automation*, volume 3, pages 1828–1832, Cincinnati, OH, USA, mar 1990.

[84] B. Rebsament, E. Burdet, Cuntail Guan, Haitong Zhang, Chee Leong Teo, Qiang Zeng, M. Ang, and C. Laugier. A Brain-controlled Whellchair Based on P300 and Path Guidance. In *The first IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1101–1106, 22 2006.

[85] T. Röfer. Routemark-based navigation of a wheelchair. In *Proceedings of the 3rd ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems*, pages 333–338, 1997.

[86] T. Röfer. Building Consistent Laser Scan Maps. In *Proceedings of the 4th European Workshop on Advanced Mobile Robots (Eurobot 2001)*, volume 86, pages 83–90. Proceedings of the 4th European Workshop on Advanced Mobile Robots (Eurobot 2001), 2001.

[87] K. Schilling, H. Roth, R. Lieb, and H. Stützle. Sensors to improve the safety of wheelchair user. *Improving the quality of life for the european citizens IOS Press*, 4 :331–335, 1998.

[88] H. Shatkay and L.P. Kaelbling. Learning Geometrically-Constrained Hidden Markov Models for Robot Navigation : Briding the Topological-Geometrical Gap. *Journal of Artificial Intelligence Research*, 16 :167–207, Mars 2002.

[89] R. Simpson, E. LoPresti, S. Hayashi, S. Guo, R. Frisch, A. Martin, W. Ammer, D. Ding, and R. Cooper. The Smart Power Assistance Module for Manual Wheelchairs. In *Proceedings Of CWUAAT '04*, pages 51–54, Cambridge, UK, 24 2004.

[90] R. Simpson, D. Poirot, and F. Baxter. The hephaestus smart wheelchair system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(2) :122–125, Juin 2002.

[91] R. C. Simpson and S.P. Levine. Automatic Adapatation in the NavChair Assistive Wheelchair Navigation System. In *IEEE transactions on Rehabilitation Engineering*, volume 7, pages 452–463, dec 1999.

[92] Richard Simpson, Edmund LoPresti, Steve Hayashi, Illah Nourbakshs, and David Miller. The smart Wheelchair Component System. *Journal of Rehabilitation Research and Development*, 41(3B) :429–442, may 2004.

[93] Richard C. Simpson. Smart wheelchairs : A literature review. *Journal of Rehabilitation Research and Development*, 42(4) :423–436, jul 2005.

[94] Sebastian Thrun, M. Beetz, Maren Bennewitz, Wolfram Burgard, A.B. Creemers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Chuck Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11) :972–999, November 2000.

[95] Sebastian Thrun and Arno Bucken. Integrating grid-based and topological maps for mobile robot navigation. In *AAAI/IAAI, Vol. 2*, pages 944–950, 1996.

[96] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo localisation for mobile robots. *Artificial Intelligence*, 128 :99–141, 2001.

[97] N. Vlassis, N. Sgouros, G. Efthivoulidis, G. Papakonstantinou, and P. Tsanakas. Global path planning for autonomous qualitative navigation. 1996.

[98] H. Wakaumi, K. Nakamura, and T. Matsumura. Development of an automated wheelchair guided by a magnetic ferrite marker lane. *Journal of rehabilitation research and development.*, 29(1) :27–34, 1992.

[99] G. Weiss and E. von Puttkamer. A map based on laserscans without geometric interpretation. 1995.

[100] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omnidirectional vision for robot navigation. In *Proc. IEEE Workshop on Omnidirectional Vision - Omnivis00*, 2000.

[101] J. Yang, Y. Ku, and C. S. Chen. Hidden markov model approach to skill learning and its application in telerobotics. Technical Report CMU-RI-TR-93-01, Robotic Institute, Carnegie Mellon University, Pittsburg, PA, jan 1993.

[102] S. Young. Large Vocabulary Continuous Speech Recognition : a Review. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Underrstanding*, pages 3–28, Snowbird, Utah, December 1995.

Titre :

Aide à la navigation pour les personnes handicapées : reconnaissance de trajets.

Résumé :

La robotique pour l'aide aux personnes handicapées s'est grandement développée ces deux dernières décennies. Le but de cette discipline est d'aider ces personnes au quotidien dans leurs déplacements et la réalisation de tâches courantes. Le travail présenté dans ce mémoire porte sur l'aide à la navigation pour un fauteuil intelligent. Le but est de réaliser un système de reconnaissance et de suivi de trajets. Ce système doit, à terme, éviter à l'utilisateur de définir tous les changements de direction. En effet, dans les cas de handicap sévères, seuls quelques signaux de commande sont perceptibles et il est généralement très fatiguant pour ces personnes de générer ces signaux.
Ce travail s'inscrit dans le projet VAHM qui utilise des fauteuils électriques du commerce sur lesquels ont été ajoutés une série de capteurs et un calculateur. Le projet VAHM utilise un système multi agents pour contrôler le fauteuil. Différents comportements (suivi de direction, suivi de mur, évitement d'obstacles,...) sont mis en oeuvre pour réagir à l'environnement.
Dans un premier temps, on étudie la possibilité d'utiliser l'enchaînement des comportements du fauteuil pour modéliser et reconnaître un trajet en utilisant les Modèles de Markov Cachés (MMC). Cette méthode montre ses limites en ce qui concerne la généralisation du modèle à des trajets non appris.
Puis l'utilisation des Modèles de Markov Cachés Multi Dimensionnels (MMC-MD) est alors étudiée en intégrant, en plus des comportements, des données provenant des capteurs. Cependant les essais montrent que la reconnaissance est meilleure en utilisant uniquement les données provenant des capteurs.
Ces remarques ont conduit à l'utilisation exclusive des données provenant des capteurs dans la suite du projet pour le choix de la méthode de reconnaissance. L'algorithme CONDENSATION est alors testé pour réaliser la reconnaissance. Ce système possède de nombreux avantages dont celui de ne pas avoir besoin de modèle complexe puisque un seul trajet de référence est nécessaire pour réaliser la reconnaissance. Ces modèles de trajets sont ensuite mis dans un réseau bayésien pour faciliter les transitions d'un modèle à l'autre.
La dernière partie du mémoire présente les résultats de la mise en oeuvre de la reconnaissance en conditions réelles sur le fauteuil VAHM 3.

Mots clés :

Fauteuil roulants intelligents, aide à la navigation, modèles de Markov cachés, algorithme CONDENSATION, reconnaissance de trajets, apprentissage.

Title :

Navigation assistance for disabled people : trajectory recognition

Abstract :

Robotics for disabled people had been largely developed these two last decades. The main goal of this research topic is to help disabled people in their common deplacements and tasks. The work presented in this thesis concerns navigation assistance for a smart wheelchair. We propose a new approach based on this behaviour-based structure, aiming at assisting the user in a global way. The intelligent system relies on a modelisation of the most frequently used routes and assists the user when navigating by suggesting the next movement when the route has been recognised. The user will not have to change the direction, thus sparing him/her an action which can be long and tiring in case of a severe disability.
This work is included in the VAHM project that use commercial electric wheelchair which is enhanced by several sensors and a calculator. The VAHM project is using a multi-agent control system. Different behaviours (direction following, wall following, obstacle avoidance ...) are designed react to the environment.
At first, we tried to to model and recognize trajectories with Hidden Markov Models (HMM) only using wheelchair's behaviour sequences. This method showed its limits to generalize the recognition of unlearned trajectories.
Then we used Multidimensional Hidden Markov Models (M-HMM) (by adding sensor's data to the behaviours) to achieve recognition. However, our tests show that the recognition is better by using only sensor's data.
These observations have conducted us to use only sensor's data for the recognition. A simple test showed that curves discribed by these crisp data for different trajectories were quite similar. So, we tried to find an adequate method to deals with such problems. The CONDENSATION algorithm had been tested to perform this recognition. This system earn numerous advantages including that it doesn't need a complex model since only one reference trajectory is necessary to perform the recognition. Then these trajectory models have been put into a Bayesian network to facilitate transitions between models in order to recognize a long trajectory.
The last part of this thesis presents recognition's results in real conditions on the VAHM 3 wheelchair prototype.

Keywords :

Smart wheelchair, navigation assistance, hidden Markov models, CONDENSATION algorithm, trajectory recognition, learning.