REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Ecole Nationale Polytechnique

Département d'Automatique

Projet de fin d'études en vue de l'obtention du diplôme d'ingénieur en Automatique

Thème :

Détection et suivi d'objets dynamiques dédiés aux véhicules autonomes : Approche bayésienne

Dirigé par :
Dr N. Ouadah (CDTA)
Pr H. Chekireb (ENP)

Etudié par :
DJEMA Walid

Juin 2014

- *ENP – Ecole Nationale Polytechnique – 10 Avenue Hassen Badi, 16200 El Harrach, ALGER*
- *CDTA : Centre de développement des technologies avancées – Baba Hassen, ALGER*

Remerciements :

En premier lieu, je remercie dieu tout puissant de m'avoir donné le courage, la volonté et la patience pour réaliser ce travail.

Je remercie mes parents, ma sœur et Nadine, qui m'ont apporté l'aide et le soutien tout au long de mes études.

Je tiens à exprimer mes vifs remerciements à Monsieur Chekireb, mon promoteur à l'ENP et à Monsieur Ouadah, mon encadrant au CDTA. Sans oublier Monsieur Tiar pour son aide et sa disponibilité.

Je remercie également mes enseignants à l'Ecole Nationale Polytechnique et à l'Ecole Préparatoire d'Alger.

Enfin, je remercie les membres du jury : Monsieur Illoul chef de département d'Automatique et Monsieur Stihi président du jury, d'avoir accepté d'évaluer ce travail.

Dédicace :

Je dédie ce travail à mes très chers parents qui m'ont toujours apporté leur amour et leur soutien pour affronter les difficultés de la vie. Qu'ils trouvent ici le témoignage de ma reconnaissance la plus dévouée.

A ma sœur Manel et à son mari Madjid.

A Nadine.

A tous les membres de ma famille : oncles, tantes, cousins et cousines.

A tous mes enseignants du primaire jusqu'à l'ENP

A tous mes amis à Polytechnique, Oxyjeunes et à tous les Dellysiens.

w@lid.

Table des matières

Table des figures

Introduction

La problématique de la détection et le suivi d'objets dynamiques pour un robot mobile n'est pas simple à résoudre et jusqu'à présent il n'existe aucune méthode systématique pour un résultat parfait.

Beaucoup de questions restent encore ouvertes dans le domaine de l'intelligence artificielle, des véhicules autonomes, des systèmes ADAS *"Advanced Driver Assistance Systems"* et de la théorie de navigation sans collision. Les défis sont importants, d'abord sur le plan scientifique mais aussi sur le plan économique, alors qu'aujourd'hui les spécialistes estiment que la première voiture complètement autonome sera commercialisée en 2020.

Lorsque l'on parcourt la documentation sur ce thème, on découvre que le DATMO *"Detection And Tracking of Moving Objects"* est étroitement lié au SLAM *"Simultanous Localization And Mapping"*. Le DATMO est le noyau de la détection et du suivi des objets dynamiques alors qu'en SLAM on cherche à localiser le robot mobile durant sa navigation et à retracer des cartes de l'environnement qui l'entoure. Ces deux techniques sont implémentées sur un même robot et utilisent les mêmes outils de traitement et de perception de l'univers externe.

L'utilité du module DATMO s'avère incontournable. En effet, un robot mobile autonome doit pouvoir détecter les obstacles dynamiques qui l'entourent afin d'éviter toute collision. Ainsi, pour répondre aux applications exigeantes et au fort déploiement des robots aussi bien dans le domaine industriel que celui de la recherche, on a besoin de développer des systèmes robotisés toujours plus performants, plus puissants et plus flexibles avec des degrés d'autonomie élevés et des capacités accrues en perception, planification et évolution dans des endroits non-structurés [10]. De plus, un module de SLAM déployé seul sur un robot ne peut être efficace d'abord pour les raisons de sécurité évoquées précédemment mais aussi pour la fidélité et la qualité des cartes retracées. En effet, sans le DATMO, on ne peut filtrer les objets dynamiques.

Dans ce mémoire, il est question de réaliser un module de DATMO pour le Robucar, véhicule d'essai au CDTA, qui réponde d'une part aux exigences de SLAM, et plus particulièrement à l'ICP-SLAM (ICP : *"Iterative Closest Points"*) et d'autre part aux besoins de planification du même robot.

Dans le premier chapitre, on s'intéresse aux notions de SLAM, DATMO et systèmes ADAS, les trois domaines phares des voitures autonomes de demain. On mettera en relief les concepts qui les relient et qui les différencient en même temps. De par leur implémentation sur un même robot et leur interaction, ils utilisent le même équipement sensoriel extéroceptif et proprioceptif qui sera aussi abordé dans ce chapitre.

Le chapitre deux est consacré à la programmation descriptive des robots et en particulier à la formulation bayésienne du DATMO.

La partie expérimentale s'étale sur les chapitres trois et quatre : le premier justifie le choix de programmation bayésienne du Robucar et pointe le besoin de doter ce dernier d'un module de DATMO qui interagit avec le module de ICP-SLAM. Enfin, au dernier chapitre, on opte pour l'approche par grille d'occupation dans la résolution du module de DATMO, en mettant en relief ses interactions avec la planification de trajectoire et le SLAM.

Chapitre 1

Généralités

Vue de nos jours, l'histoire des robots peut clairement se répartir en trois grandes phases. La première s'étend de l'Antiquité à la fin de la Seconde Guerre mondiale, longue période durant laquelle le concept reste du domaine des mythes et des idées. La seconde s'achève dans la décennie 1980-1990. Elle correspond à l'émergence de la robotique scientifique et à la maîtrise des robots industriels à poste fixe. La troisième phase dont nous sommes les témoins, très encouragée par l'extension extraordinaire de l'informatique, est à la recherche de la maîtrise du robot de service mobile, au comportement autonome et à son insertion conviviale au sein de la société humaine [3]. Ces nouveaux systèmes ont soulevé un grand nombre de problèmes difficiles. Nombre de ceux-ci ne sont d'ailleurs toujours pas résolus. Ainsi, alors que les robots manipulateurs se sont aujourd'hui généralisés dans l'industrie, rares sont les applications industrielles qui utilisent des robots mobiles [6].

1.1 Définition et utilité de la robotique mobile

Le plus souvent, lorsqu'on parle de robotique mobile, on désigne les robots à roues. Les autres types de robots mobiles sont plus connus par leurs moyens de locomotion. On distingue en tout :

- Les robots mobiles à roues.
- Les robots mobiles à pattes.
- Les robots mobiles à chenilles.
- Les robots mobiles volants / aériens.
- les robots mobiles marins et navigants.

Les robots aériens sont souvent connus sous le nom de AUAV (autonomous unmanned aerial vehicule). L'exemple le plus connu étant le drone.

Dans [7], on définit un robot mobile comme étant : « Un robot doté de moyens de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a ».

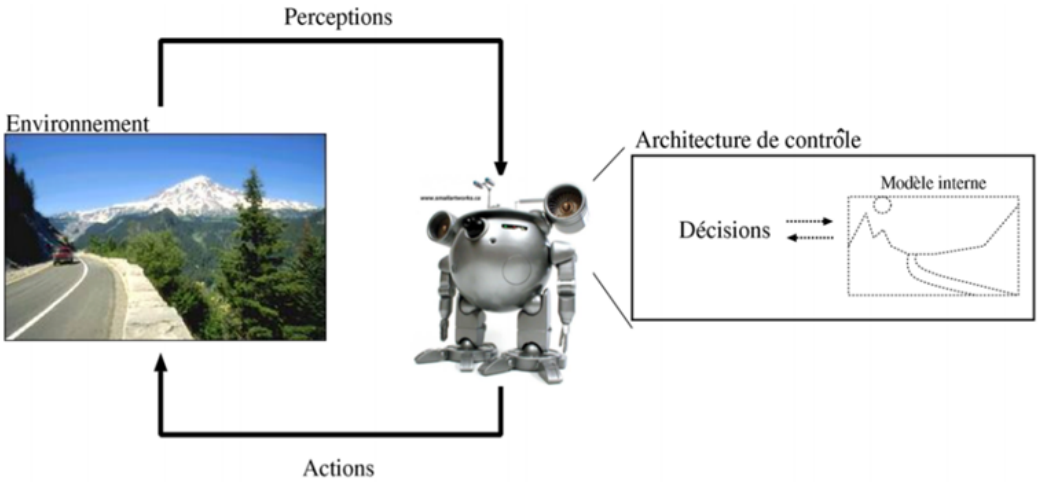


FIGURE 1.1 – Schéma des interactions d'un robot avec son environnement

De nos jours, le rôle des robots ne se limite plus à la simplification de l'existence humaine mais s'étend aussi à la protection de leurs « maîtres ». Ainsi les militaires ont développé des systèmes automatisés tels que les drones ou les chars qui peuvent détecter les mines et explosifs et stopper l'avancement du véhicule en cas de danger. D'autres robots mobiles apportent de l'aide aux soldats blessés. Dans les aéroports des robots existent pour surveiller et fouiller des bagages suspects et désactivent les bombes et les explosifs. C'étaient là les premières applications de la robotique mobile avant même de penser à rendre les voitures complètement autonomes. Les voitures autonomes peuvent assister les personnes physiquement malades. On trouve aussi de nouvelles générations de robots mobiles qui servent en exploration spatiale, le plus performant et le plus connu *«Mars Rover»*, petit bijou de la NASA destiné à l'exploration de la planète rouge. D'autres robots mobiles servent à l'exploration des océans ou sont envoyés dans des zones à haut-risque ou radioactives.

FIGURE 1.2 – Exemples de robots mobiles d'utilité militaire, civile ou de recherche

Mis à part les domaines d'activité cités ci-dessus, le marché commercial de la robotique mobile est toujours relativement restreint en dehors des robots aspirateurs vendus à plusieurs millions d'exemplaires [7]. Cependant, il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. La voiture complètement autonome qui sera vendue au grand public est attendue pour 2020. Entre temps, des travaux très importants se réalisent en recherche pour préparer ce futur marché mondial. L'équipe de la division de robotique du CDTA travaille dans ce sens en voulant doter son véhicule d'un maximum d'autonomie possible. La navigation en toute sécurité dans des environnements dynamiques reste un défi majeur pour la voiture autonome de demain.

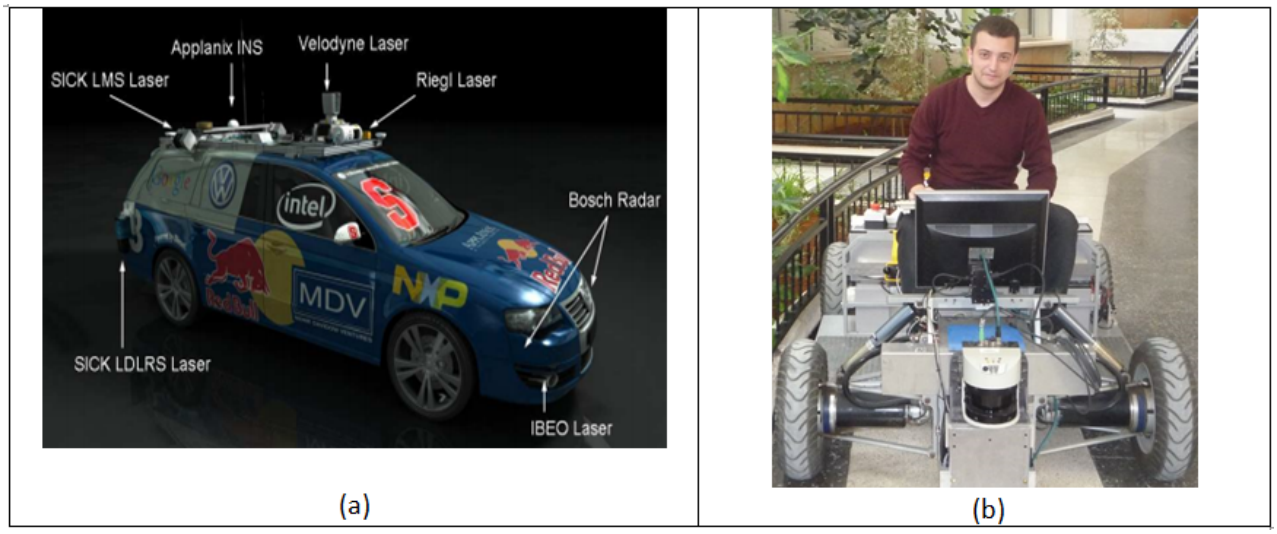


FIGURE 1.3 – (a) Voiture expérimentale de la Stanford University, (b) Le robucar, voiture d'essai au CDTA

1.2 La navigation autonome des robots mobiles

La robotique mobile a permis d'augmenter considérablement nos connaissances sur la localisation et la navigation de systèmes autonomes [6]. Mais qu'est-ce qu'un robot autonome ? Existe-t-il un seul type de robots autonomes ?

Dans sa thèse, Morette précise d'abord que dans ce domaine d'étude on parle d'autonomie au sens décisionnel qui est différente de l'autonomie énergétique, par exemple, où le robot doit gérer de façon autonome ses propres sources d'énergie.
En mode autonome le robot doit prendre ses propres décisions. Cela signifie qu'il doit être capable à la fois de percevoir correctement son environnement, mais également de savoir comment réagir en conséquence, suivant le niveau d'autonomie [8]. C'est à lui de planifier son parcours et de déterminer avec quels mouvements il va atteindre son objectif. Les recherches dans ce domaine portent principalement d'une part sur la localisation du véhicule autonome et la cartographie de son environnement, d'autre part sur le contrôle de tels véhicules (structure de contrôle, stratégies de commande, planification). Les équipes de recherche

dans le domaine de la robotique mobile travaillent dans le but de doter leurs plates-formes d'essai d'un maximum d'autonomie. C'est dans ce sens que travaille l'équipe de robotique du CDTA travaille sur le Robucar.

1.3 La perception

Un robot est d'abord une entité mécanique ou physique avec une carcasse, des moteurs, des actionneurs et des batteries. Quant au fonctionnement il est géré par des cartes électroniques et des calculateurs (unité de traitement et de commande). En robotique mobile ces unités de traitement et de décision sont beaucoup plus performantes que celles utilisées pour la robotique industrielle classique comme les bras manipulateurs aux usines d'assemblage ou de soudage. Mais ce qui creuse d'avantage le fossé entre la robotique industrielle classique et la robotique mobile autonome ce sont les importants moyens de perception mis à disposition du robot pour qu'il localise sa position et celle des obstacles qui l'entourent. On exige d'un robot mobile autonome qu'il puisse « percevoir » correctement l'environnement dans lequel il évolue. C'est la base même de tout système autonome. Sans une bonne perception et interprétation de ce qui l'entoure, un robot ne peut pas prendre de décision correcte [8]. Les robots sont alors équipés d'organes sensoriels qui sont les capteurs qu'on classe traditionnellement en deux catégories selon qu'ils mesurent l'état du robot lui-même ou bien l'état de l'environnement qui l'entoure.

1.3.1 Les capteurs proprioceptifs

Ils sont utilisés pour fournir des informations sur l'état du robot lui-même lorsqu'il se déplace, comme sa position ou ses coordonnées, sa vitesse, son accélération ou encore l'angle de giration. La connaissance du déplacement effectué par le robot est d'une importance capitale. Les odomètres sont généralement utilisés pour estimer le déplacement mais ils présentent le grand problème d'une dérive qui croît au cours du temps.

Pour un robot mobile à roues, ces odomètres sont faciles à mettre en œuvre. Ils sont associés aux roues et estiment le déplacement en calculant le nombre de rotations effectuées par ces dernières. Sur un terrain lisse, le glissement des roues cause beaucoup d'erreurs d'estimation du déplacement. Des erreurs qui s'accumulent en parcourant de grandes distances et qui s'aggravent encore plus lorsque le robot effectue des mouvements de rotation en plus de sa translation. Des techniques d'odométrie nouvelles viennent remédier à ces problèmes comme l'odométrie visuelle mais aussi le filtrage bayésien qui est aussi utilisé dans l'estimation de la position du robot lui-même. La référence [7] donne la technique employée et le raisonnement bayésien dans le cas appelé « localisation multi hypothèses ». Cette méthode de filtrage permet d'intégrer de manière similaire les deux types d'informations (odométrie et perceptions), mais ne gère pas explicitement les hypothèses de position. Les différentes hypothèses sont ici remplacées par une distribution de probabilité de présence du robot sur l'ensemble des positions possibles de la carte. Le raisonnement bayésien sera exposé en détail aux prochains chapitres de ce mémoire. On verra par la suite de nouvelles techniques plus précises comme la localisation du robot par les techniques ICP en SLAM qui se basent sur les capteurs extéroceptifs à grand succès que sont les télémètres.

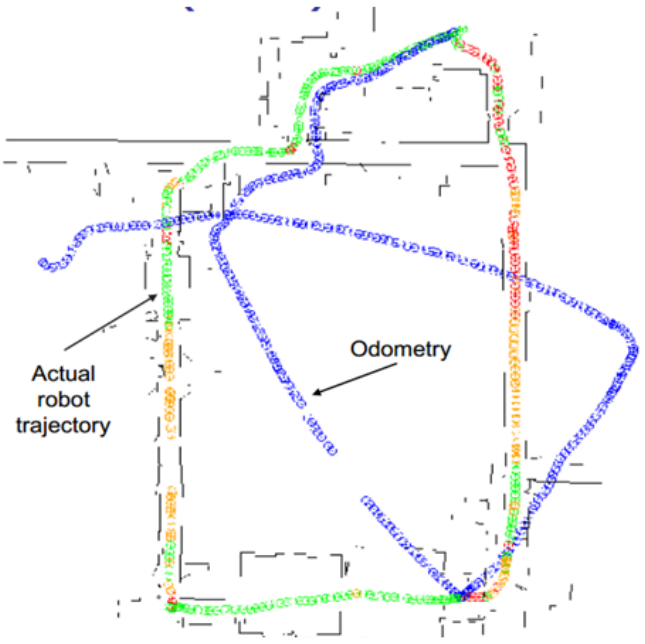


FIGURE 1.4 – Dérivation de l'odométrie classique d'un robot mobile

1.3.2 Les capteurs extéroceptifs

Ils fournissent des données de l'extérieur et de l'environnement du robot. Différents choix de capteurs extéroceptifs s'offrent au concepteur de robots, selon le type de robot mobile et son environnement. Les plus utilisés en DATMO, sont les télémètres laser et les caméras. De nos jours les capteurs ultrason sont devenus complétement obsolètes. Certains systèmes incorporent parfois des radars [22]. Les systèmes GPS sont aussi largement utilisés en localisation outdoor, jamais seuls. Dans ce paragraphe on se limite aux deux capteurs concurrents, aussi performants l'un que l'autre, qui se disputent le monopole des voitures de demain. D'une part nous distinguons les lasers «laser range finders» et d'autre part les caméras (caméra simple, stéréoscopique, voire caméra avec capteur de profondeur de type Microsoft Kinect).

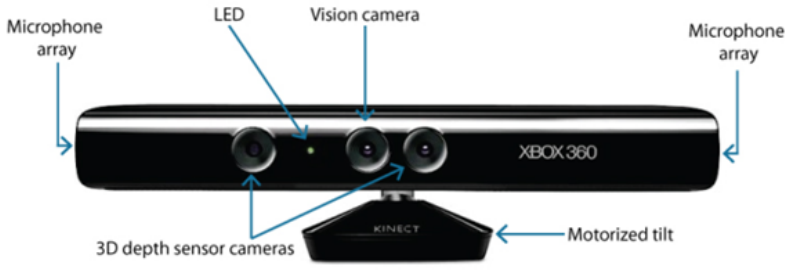


FIGURE 1.5 – La Kinect de Microsoft- utilisée en XBOX et très répandue en robotique

Les caméras optiques

Largement utilisées, ces capteurs coûtent relativement peu cher et se caractérisent par la grande quantité d'informations qu'ils fournissent. Dans le domaine du SLAM et du DATMO on retrouve deux technologies de caméras très réussies : les «charge-coupled devices (CCD)», et la technologie «complementary metal oxide semiconductors» (CMOS). La caméra Kinect de Microsoft (figure 1.5) est aussi très employée en robotique mobile pour les multiples avantages qu'elle offre et sa facilité d'intégration dans les systèmes les plus complexes. Elle s'est écoulée en plusieurs milliers d'exemplaires dès sa sortie au grand public et elle équipe aujourd'hui 90% des plates-formes d'essais des équipes de recherche.

FIGURE 1.6 – (a) i-Nova CCD PLA-C+ 310Kp (b) CMOS CAMCOLMBLAH2

Laser Range Finders

Les télémètres laser utilisent le principe du temps de voyage du faisceau laser qu'ils envoient pour mesurer les distances. Ce faisceau laser fournit un balayage plan de 100 °, 180 ° ou 360 ° selon le modèle du capteur dont on dispose. Le « Robucar » du CDTA (figure 1.3) dispose de deux détecteurs laser (LMS511 installé en avant et LMS200 installé à l'arrière du véhicule). Leur balayage est en réalité de 190 ° chacun, limité à 180 ° volontairement, avec une précision de 1 °.

Ces capteurs sont les plus répandus et on les retrouve sur tous les robots de type voiture grâce à la précision qu'ils offrent même s'ils restent relativement chers par rapport aux caméras. Les capteurs laser fonctionnent en l'absence de lumière, néanmoins, les rayons solaires peuvent causer quelques fausses mesures si le robot y est fortement exposé. Il existe aussi quelques objets non détectables qui sont soit trop réfléchissants (les éléments en acier chromé par exemple) soit pas réfléchissants du tout (des vitres). Des cas pratiques seront exposés au chapitre Implémentation et Expérimentation.

Récemment ces laser ont évolué de l'espace 2D à ce qu'on appelle les «Multilayer 2D-devices». Les multilayer fournissent plusieurs faisceaux, souvent quatre, parallèles et horizontaux et servent d'une part à détecter des obstacles de hauteurs différentes, et d'autre part à éliminer les fausses détections notamment ce qu'on appelle «le problème de la route inclinée».

Une innovation très récente dans l'industrie du capteur laser est le nouveau 3D range finder Velodyne. Ce dernier est équivalent à 64 capteurs laser 2D et il fournit plus de deux

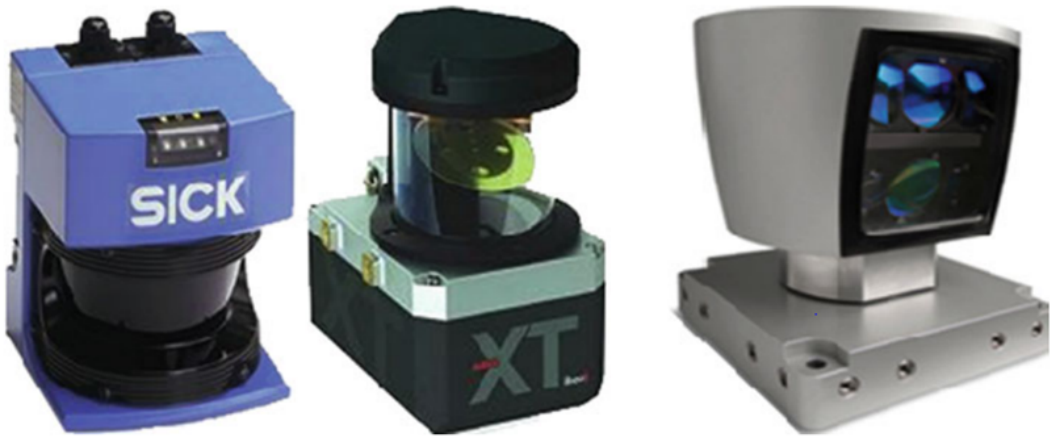


FIGURE 1.7 – 2D Sick LMS200 – Multilayer 2D laser IBEO – 3D Velodyne HDL

millions de points de mesures par seconde. Le premier succès du Velodyne était dans le DARPA Urban Challenge en 2007 et depuis il se positionne comme le meilleur équipement prévu pour les voitures autonomes du futur. Enfin, la figure 1.8 donne une comparaison non exhaustive entre les lasers et les caméras spécialement dans le domaine du DATMO.

Camera	Laser range finder
- Prix raisonnable	- Prix très élevé
- Complexité des outils de développement.	- Simplicité d'utilisation
- Grande précision et grande résolution	- Grande précision et faible résolution
- Image colorée	- Image non colorée
- Robustesse moyenne	- Robustesse moyenne

FIGURE 1.8 – Comparaison entre capteurs de type télémètres laser et caméra

On note que le Robucar est doté de tous les moyens de perception cités ci-dessus. Néanmoins la problématique à laquelle ce travail répond a fait en sorte qu'on opte pour les capteurs laser plutôt que pour les caméras. La première raison en est la grande précision des capteurs Laser-LMS qui équipent le Robucar ; la deuxième raison est relative au temps de calcul et aux moyens de traitement qu'une application temps-réel comme la nôtre nécessiterait si des caméras étaient utilisées.

1.4 Présentation du SLAM et du DATMO

Le SLAM (Simultaneous Localisation And Mapping) est un domaine récent en robotique mobile qui cherche à rendre les robots complètement autonomes en les dotant à la fois de la capacité de se localiser et de créer une carte de l'environnement qui les entoure de manière instantanée. Cet environnement se caractérise par la présence d'objets figés ou mouvants d'où l'utilité de joindre des techniques de détection et de suivi d'objets mobiles (DATMO).

Ces techniques permettent à la fois une navigation saine du robot en évitant toute collision avec ces objets mobiles qui risquent de croiser sa trajectoire, mais aussi de corriger et d'éliminer les erreurs sur les cartes qu'il retrace par les techniques de SLAM. [5] Le module DATMO s'avère essentiel pour sécuriser la conduite des robots mobiles autonomes. Cependant, sa conception reste assez délicate à cause du nombre important et des divers obstacles qu'un véhicule peut croiser pouvant être des voitures, des camions, des bicyclettes, des animaux ou des piétons. On va voir dans la suite de ce chapitre que les modèles de scènes et d'obstacles qui s'utilisent sont aussi divers que la diversité des obstacles eux-mêmes.

Quant au domaine du SLAM, il bénéficie actuellement d'un intérêt particulier de la recherche spécialisée dans les véhicules intelligents [12]. Jusqu'à présent la plupart des travaux en SLAM étaient réalisés dans des environnements statiques. Lorsqu'on tient compte des objets dynamiques on parle plutôt de SLAMIDE (SLAM In Dynamic Environments) ou encore de SLAMOT. La détection des obstacles et la détermination de leur nature (statiques/dynamiques) devient alors un enjeu majeur et une nécessité pour que le robot accomplisse sa tâche.

Ces objets mouvants sont généralement classés en deux types : Les objets en mouvement permanent et ceux qui bougent momentanément. [5] Cette deuxième catégorie cause plusieurs problèmes pratiques car souvent ces objets sont considérés comme statiques et faisant partie du décor et réduisent ainsi la précision et l'exactitude du paysage retracé. Le SLAMIDE est un domaine très récent et plusieurs questions demeurent non tranchées :

- Comment peut-on distinguer entre les objets statiques et dynamiques ?
- Comment peut-on représenter ces objets statiques ou dynamiques ?
- Comment prédire le comportement et les trajectoires futures de ces objets dynamiques ?

Le SLAM et le DATMO jouent alors un rôle clé dans la robotique et l'automatique moderne [48]. Le but de ce mémoire est d'apporter des éléments de réponses aux questions ci-dessus. Le module de DATMO résout ces importants questionnements qui sont pourtant posés en SLAMIDE et montre ainsi l'étroite relation entre les deux concepts.

1.5 Les systèmes d'assistance au conducteur (ADAS)

Depuis son invention, le concept de voiture n'a pas beaucoup changé [14]. En effet, on retrouve toujours la même forme avec quatre roues et un moteur de propulsion. Evidemment les avancées technologies ont fait que les voitures sont de plus en plus performantes, rapides et avec des moteurs de plus en plus puissants. Mais ce n'est que ces dernières années que les systèmes d'assistance au conducteur sont apparus. D'abord avec l'autoradio qui est un moyen de divertissement du conducteur et ensuite le GPS pour la localisation et l'aide à la navigation. Les systèmes ADAS sont en général divisés en deux grandes familles :

Les systèmes d'information et de confort : comme l'autoradio, le GPS, les ordinateurs de bord qui fournissent des informations sur les chemins optimaux et les informations sur le trafic routier et les embouteillages. On cite aussi les systèmes multimédias récents

(DVD players), la climatisation et les sièges automatiques et intelligentes.

Les systèmes de sécurité (safety systems) : destinés à la sécurité des passagers du véhicule et aussi à la préservation de la voiture. On peut citer les systèmes de verrouillage automatique et d'alarmes, les systèmes ABS (Anti Blocking System) et ESP (Electronic Stability Program) mais aussi les systèmes d'aide au stationnement et enfin les très performants systèmes de « cruise control » pour garder les distances de sécurité entre les véhicules, limiter les vitesses de circulation selon les régions et l'avertissement en cas de risque de collision.

Ces deux familles interagissent parfois, ce qui est le cas pour les systèmes d'amortissement récents qui sont d'abord des systèmes de sécurité pour préserver la voiture et les passagers mais aussi pour le confort de ces derniers avec des sièges intelligents qui amortissent et réagissent au mouvement global du véhicule.

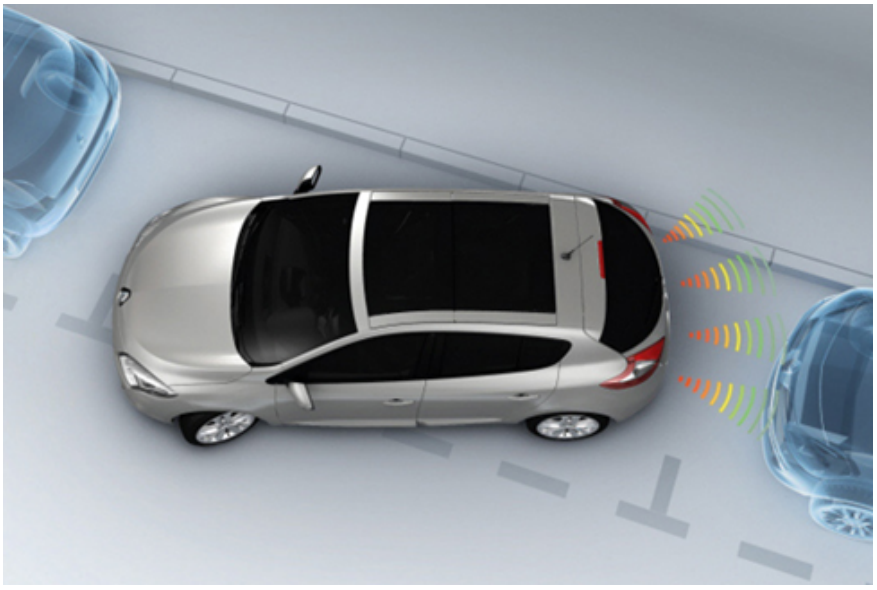


FIGURE 1.9 – Aide au stationnement par capteurs ultrasons pour une voiture Renault

Dans ce premier chapitre on désire aboutir à un positionnement clair de la problématique du DATMO. On note que les systèmes ADAS diffèrent du module DATMO dans la mesure où ce dernier est destiné à la navigation autonome des véhicules sans aucune intervention humaine. En revanche, les systèmes ADAS ne dépassent pas le cadre de l'assistance au conducteur qui reste le maître à bord du véhicule. Cette différence fondamentale qui sépare ces deux domaines de l'automobile moderne induit d'abord une disparité en termes de complexité. Il est évident que plus le degré d'autonomie du véhicule est grand plus les problèmes liés à la conception de tels systèmes deviennent importants. En second lieu, comme il est expliqué par Christian Laugier et son équipe dans [22], le critère même d'évaluation de la qualité et la fiabilité d'un module de DATMO devient complétement opposé au critère d'évaluation d'un système ADAS. En effet, si pour les concepteurs DATMO le fait de ne pas détecter des obstacles existants, *False négatives*, tend à être très dangereux alors que le contraire, *False positives*, est moins grave. On note que pour les systèmes ADAS c'est tout à fait le contraire.

1.6 Séparation entre SLAM et DATMO

Dans ce paragraphe on va illustrer les disparités qui existent entre SLAM et DATMO malgré la présence de plusieurs points communs qui les relient et qui font que la séparation entre ces deux concepts n'est pas aussi évidente que celle entre ADAS et DATMO.

Author-Paper	SLAM Method	DATMO Method		Navigation sensors	Moving objects	Test environment	Experimental platform	2D/3D Map	Contribution
		Data Association	Tracking						
Wang 2002-2004	Grid-based (Bayesian State Estimation:EKF)	MHT	IMM	Laser range finders, odometry (main sensors)	People, cars, bikes, buses	Outdoor (crowded urban areas.)	Navlab11	3D (2.5D)	Pioneer of SLAM & DATMO
Haehnel 2003	Grid-based (Bayes filter)	SBJPDA		Pioneer 2: 2D SICK laser range finder RW1 B21 Rhino: 2D SICK laser range finder	People	Indoor	Pioneer 2 RW1 B21 Rhino:	2D	Obtained better pose estimates and reduced spurious objects with a more robust algorithm
				Tilting 3D laser, 2D laser		Outdoor	Pioneer 2 AT platform	3D	
Montesano 2005	Grid map	NNR	EKF	SICK 2D laser rangefinder, odometry	People, doors	Indoor (office)	Robotic wheelchair	2D	Differentiated static & dynamic objects in estimation problem
Sola 2007 (PhD Thesis)	BiCamSLAM (through Monoslam algorithm)	EKF		2 cameras	People, Movable objects(boxes, table, bin, fence)	Indoor	Robot	3D	BiCamSLAM (through Monoslam algorithm)
Vu 2009 (Thesis)	Occupancy grid framework(Elfes 1989) (ML-SLAM & EKF)	DDMCMC		2D laser range finder	People, cars, bikes, buses	Outdoor	Simulation from Navlab dataset of Wang (2004)	2D	A new fast laser-based incremental localization method and DDMCMC
		GNN	KF	2D laser range finder, odometry 2 short range radars	Objects that may lead to collisions in traffic		Mercedes Benz E class		A new fast laser-based incremental localization method
Burlet, Vu, Aycard 2010	Occupancy grid framework(Elfes 1989) (Bayes) ICP (localization)	MHT	IMM	2D laser range finder, odometry, 2 short range radars, camera (visualization only)	People, cars	Outdoor	Diamler	2D	A new fast laser-based incremental localization method

FIGURE 1.10 – Récapitulatif des techniques employées en SLAM et en DATMO

La figure ci-dessus résume les travaux phares réalisés en SLAM et en DATMO de 2002 à 2011. [5] On remarque d'abord que ces deux domaines (SLAM et DATMO) sont très récents. En effet, les premières tentatives de Wang datent de 2002. La deuxième remarque est que ces techniques se trouvent simultanément employées sur les mêmes véhicules expérimentaux et en utilisant les mêmes capteurs extéroceptifs et proprioceptifs. Pourtant ce sont deux problématiques qui se distinguent par leurs finalités. D'abord la fonction DATMO nécessite de résoudre deux problèmes de nature très différente : la détection Moving Objects Detection (MOD) et le suivi, ou Moving Objects Tracking (MOT). Le problème MOT est résolu par des approches d'estimation [65].

La figure 1.11 illustre le principe de détection d'obstacles à partir de données laser brutes en entrée (ce qui est le cas pour le Robucar). Cette étape est commune au SLAM et au DATMO et utilise les mêmes techniques d'acquisition et de traitements de données primaires.

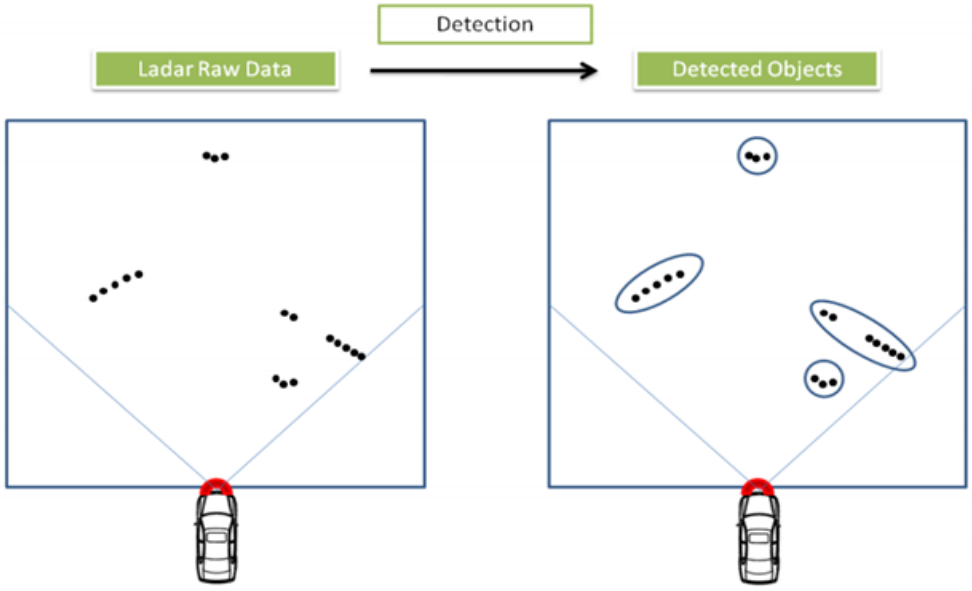


FIGURE 1.11 – Illustration du principe de détection d'obstacles

Pour le DATMO uniquement, l'étape clé consiste au « suivi » (*Tracking*) des entités détectées comme illustré sur la figure ci-dessous :

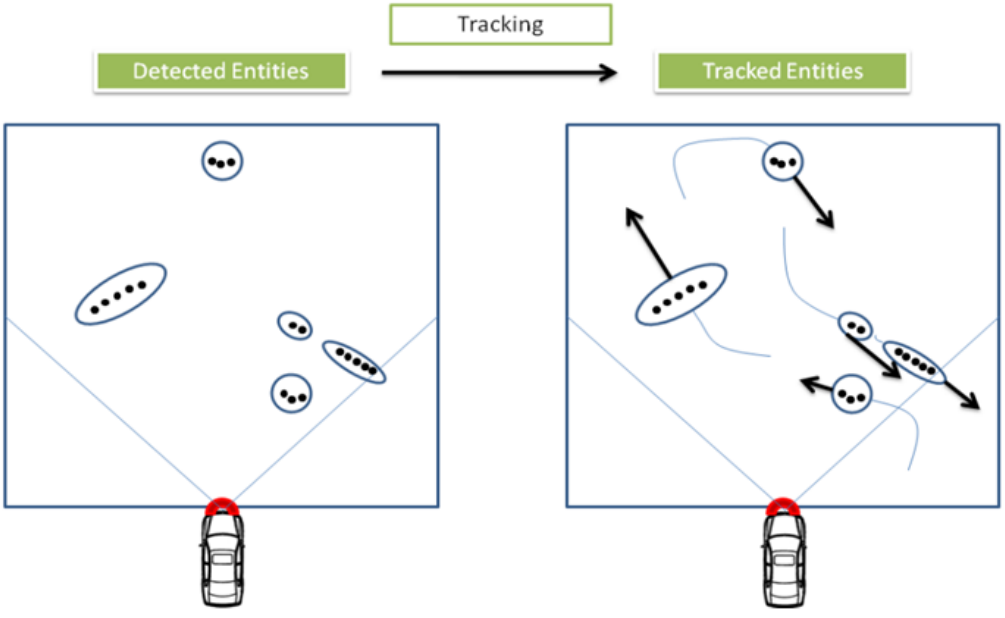


FIGURE 1.12 – Illustration du principe de suivi d'obstacles

La classification est une autre étape commune mais non-essentielle et peut s'insérer dans plusieurs contextes. On verra par exemple qu'en DATMO, il est parfois préférable de classer les objets entre statiques et dynamiques avant d'appliquer des filtres bayésiens pour l'estimation de la position et de la vélocité des obstacles détectés ou pour déterminer le

«dynamic model» qui peut être le «Brownian Motion Model», le «Linear Motion Model» ou le «Bycicle Motion Model» que nous allons illustrer au prochain chapitre de ce mémoire.

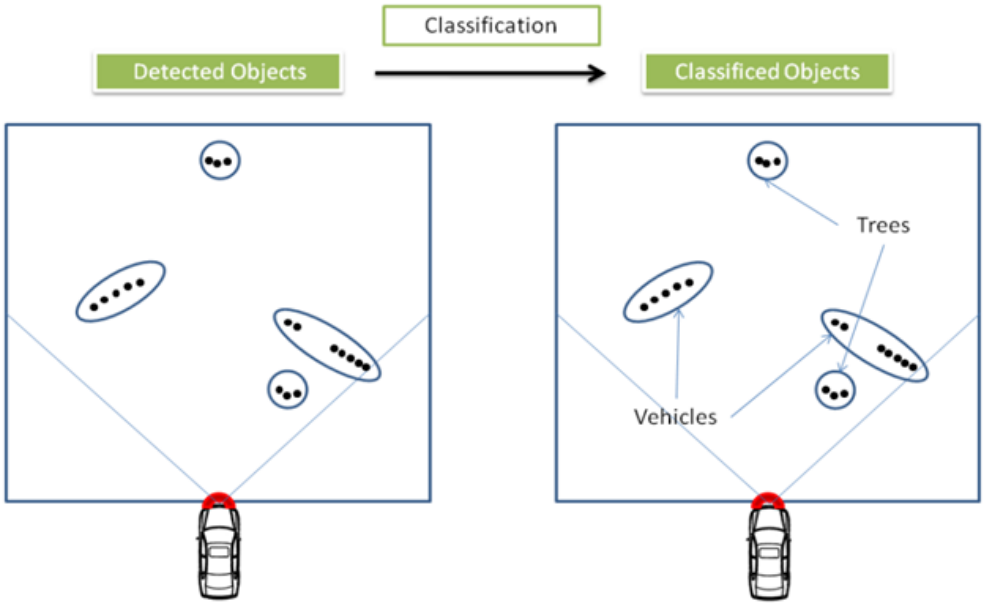


FIGURE 1.13 – Illustration du principe de classification d'obstacles

Par contre, en SLAM, les techniques de «detection» et de «classification» sont automatiquement suivies par des techniques de «Mapping» où des cartes de l'environnement sont retracées et mémorisée. Cette étape n'existe pas en DATMO.

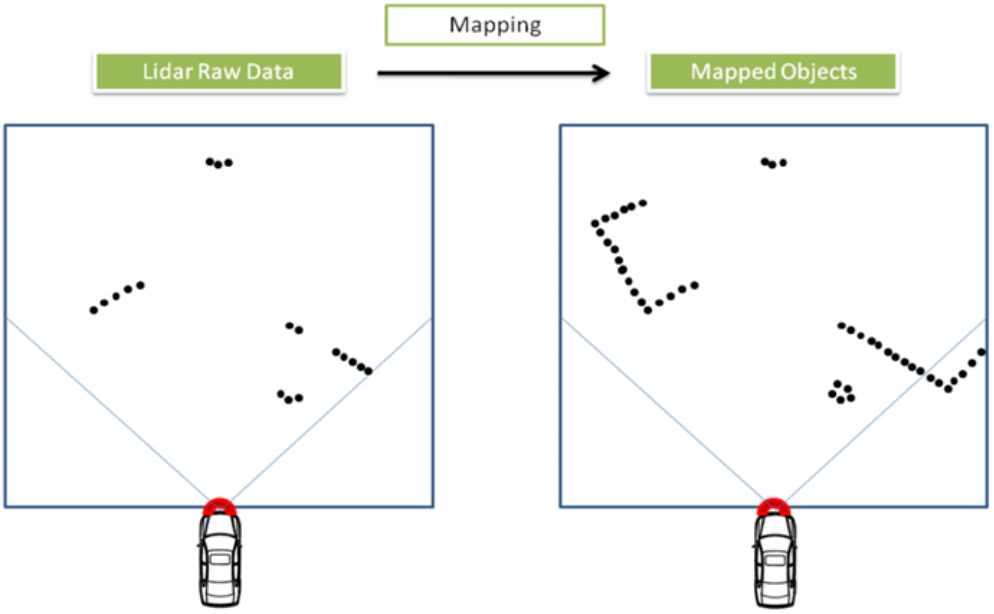


FIGURE 1.14 – Illustration du principe de classification d'obstacles

On récapitule par ces figures illustratives tirées de Wang et Thrun en 2007 [15]. Ces travaux sont les premiers essais réalisés en SLAMIDE.

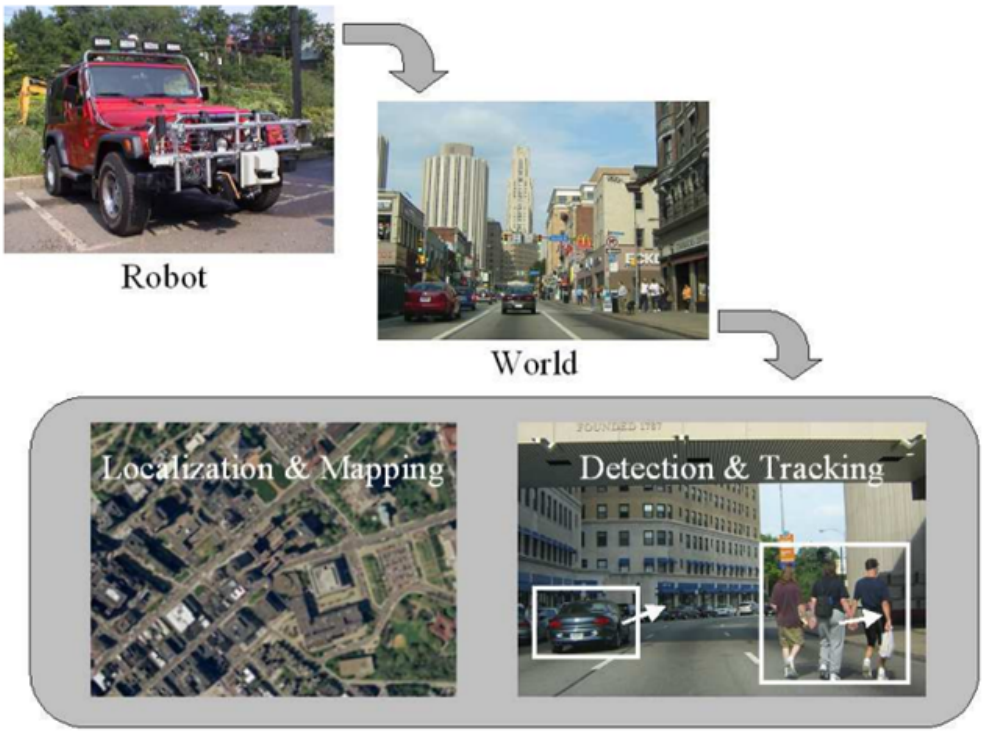


FIGURE 1.15 – Localization, mapping and tracking are critical to autonomous vehicules

1.7 Conclusion

Un grand enjeu économique et sociétal a généré cet intérêt pour l'automobile autonome et les systèmes d'assistance aux conducteurs (ADAS). Premièrement on a vu que les deux critères de complexité et fiabilité des systèmes font la séparation entre les concepts d'ADAS et de DATMO.

Dans un second lieu, la comparaison entre le DATMO et le SLAM montre que plus ces deux modules travaillent indépendamment mieux ils interagissent et se complémentent lorsqu'ils sont implémentés sur une même plate-forme mobile. La suite de ce mémoire sera uniquement consacrée au DATMO : dans le contexte des véhicules intelligents destinés au transport urbain, il est difficile de concevoir un système de détection et suivi d'obstacles dynamiques fonctionnant parfaitement sans fausse alarme [13]. On a vu que pour concevoir un tel module il faut détecter et localiser premièrement les obstacles dans l'espace. Des informations supplémentaires comme la profondeur et la taille des obstacles peuvent servir à classer ces derniers en différentes groupes selon leurs natures (piétons, voitures, bicyclettes...), pour ensuite déterminer leur évolution dynamique. Le chapitre suivant présentera les concepts bayésiens généraux du DATMO.

Chapitre 2

Formulation bayésienne du DATMO

2.1 Introduction

Le grand défi des voitures autonomes est de devoir faire parfaitement la séparation entre l'environnement statique et l'environnement dynamique. C'est une tâche délicate à cause de la complexité des milieux dans lesquels une voiture autonome évolue, du nombre important d'obstacles qu'elle peut rencontrer et de la diversité des ces obstacles. Ces derniers se classent d'après leur nature (arbre, camion, piéton, vélo,...) ou leur dynamique (objet fixe, objet en mouvement permanent ou objet momentanément à l'arrêt). Ajouter à tout cela les imprécisions des technologies actuelles, dont on a déjà parlé au premier chapitre, comme la grande dérive des odomètres en ce qui concerne les capteurs proprioceptifs du véhicule autonome ou encore les imprécisions des télémètres extéroceptifs et enfin les bruits qui entachent toute mesure. En DATMO on considère que le robot évolue dans l'environnement complexe et non-structuré décrit ci-dessus. L'objectif est non seulement de séparer les univers statique et dynamique mais aussi de suivre le mouvement de tout obstacle détecté dynamique en estimant sa position et sa vélocité pour les prochains instants à venir. Le but est d'éviter toute collision tout en ayant une navigation sécurisée du robot.

FIGURE 2.1 – Thomas Bayes – Mathématicien Anglais 1702-1761

Dans des environnements complexes où plusieurs entités interagissent de façon forcément non-prédictive, on ne peut décrire les situations ou «capturer» les informations que d'une façon probabiliste/statistique. C'est cette formulation, qu'on appelle bayésienne que nous allons exprimer au début de ce chapitre. Ensuite, on expose les différentes approches du DATMO existantes en mettant en relief les points forts et les points faibles de chacune.

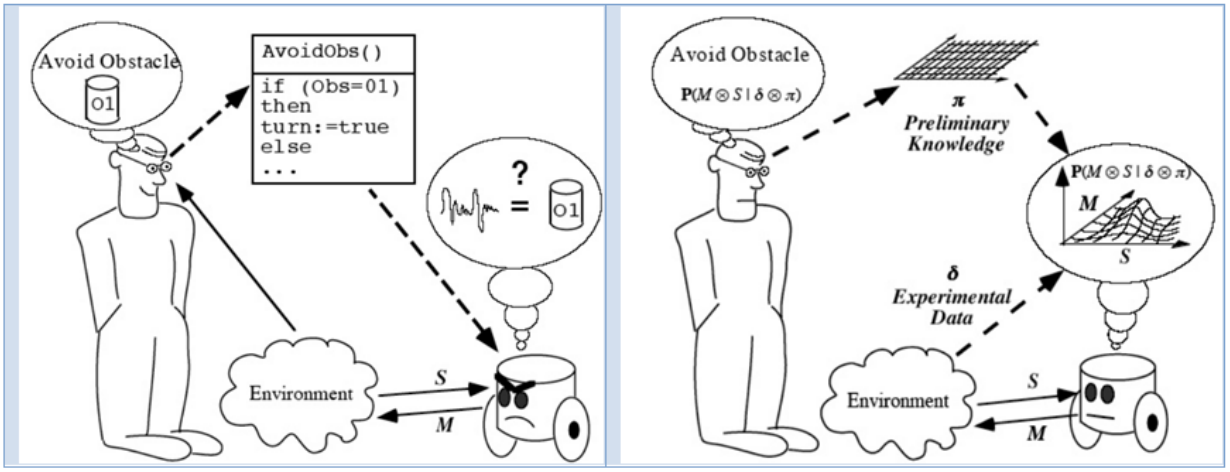


FIGURE 2.2 – L'interprétation probabiliste pour une bonne compréhension en robotique

2.2 La Probabilité comme une alternative à la logique en raisonnement et prise de décision

On estime que les êtres vivants, tout comme les robots, font face à la même difficulté fondamentale qui est l' «incomplétude» des informations qu'ils acquièrent de l'environnement qui les entoure [47]. Cette incomplétude cause en premier lieu un problème d'incertitude vis-à-vis du monde et la question centrale, à laquelle on veut répondre, que ce soit pour les humains ou pour les robots, est de savoir comment percevoir, analyser et décider efficacement en se basant sur un modèle incomplet de l'univers incertain.

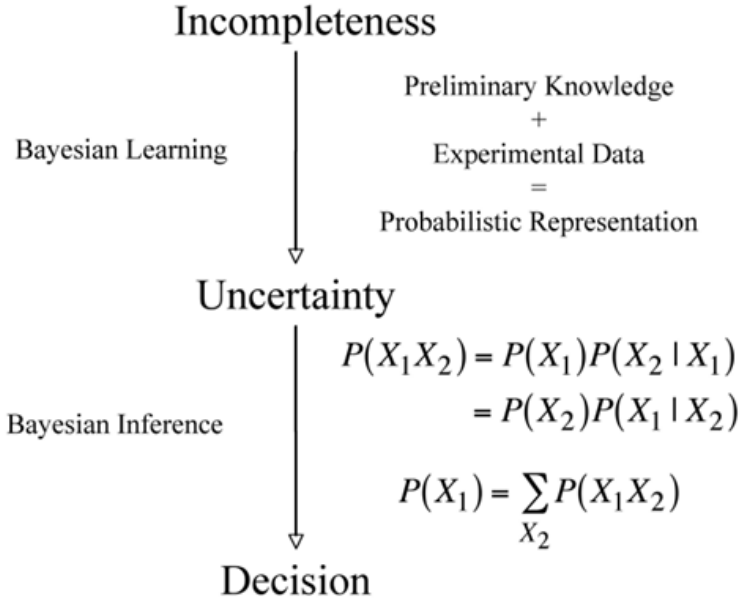


FIGURE 2.3 – La probabilité comme une alternative à la logique

En premier lieu, on s'intéresse aux bruits de mesures et aux incertitudes liées aux capteurs de façon générale. Dans d'autres contextes et dans d'autres circonstances, des hypothèses simplificatrices et des connaissances a priori de l'environnement et du capteur, permettent de « contrôler » ces bruits au point de ne plus les considérer. Ce n'est pas le cas en robotique mobile autonome où la seule solution est de prendre en considération, ou d'admettre, ce manque d'information. Comme nous allons voir par la suite, le modèle du capteur lui-même va être pris comme une distribution probabiliste avec certaines caractéristiques. Le challenge que nous allons relever par la suite est d'aboutir au final à un raisonnement et à une prise de décision tout deux rationnels vis-à-vis du monde externe.

La figure 2.3 représente les deux étapes nécessaires à la prise de décision :

- Etape 1 : La transformation de l'incomplétude en une incertitude en partant des connaissances préalables et des données expérimentales acquises. Ce processus appelé « learning » ou apprentissage construit les distributions probabilistes qui décrivent l'environnement. On note que plus les connaissances passées sont précises plus l'information portée par ces distributions est fiable.
- Etape 2 : C'est un raisonnement sur les distributions de probabilités obtenues à l'étape 1. Pour ce faire, nous avons besoin de deux règles qu'on appelle les règles du « bayesian inference ».

2.3 Les concepts de la programmation Bayésienne

D'aucun ayant eu à programmer un robot réel dans un environnement physique s'est inévitablement retrouvé confronté aux problèmes de l'incertitude [32]. Comme on l'a expliqué auparavant, pour faire face à ces problèmes d'incertitude on attache des probabilités à toutes les propositions possibles. Mais d'abord, on va définir les concepts de base de la programmation Bayésienne.

2.3.1 Une proposition

Une proposition logique peut être vraie ou fausse. On peut obtenir de nouvelles propositions par des combinaisons à travers des opérateurs logiques comme la conjonction des propositions a et b qu'on dénote $a \wedge b$, ou leur disjonction $a \vee b$. On dénote aussi la négation d'une proposition a par $\neg a$.

2.3.2 Une variable

Par définition, une variable discrète X est un ensemble de propositions logiques x_i qui s'excluent mutuellement (en d'autres termes : $\forall i, j$ tels que $i \neq j$, $x_i \wedge x_j$ est fausse) et qui sont exhaustives (c.à.d. au moins une des propositions x_i est vraie).

On note par $< X >$ le cardinal de X qui est le nombre de propositions x_i et qui équivaut au nombre de valeurs que la variable X peut prendre.

La conjonction $X \wedge Y$ des deux variables X et Y, notée $< X > . < Y >$, est définie par l'ensemble des propositions $x_i \wedge y_j$. On démontre que $X \wedge Y$ est un ensemble de propositions qui sont-elles mêmes exhaustives et s'excluent mutuellement. Par conséquent, $X \wedge Y$ est une nouvelle variable. Ce résultat se généralise à la conjonction de n variables [47].

2.3.3 Une probabilité

On considère que pour assigner une certaine probabilité à une proposition donnée il faut qu'on dispose d'un minimum de connaissances préalables. Par exemple, en DATMO, on peut exploiter les données issues du capteur laser à l'instant t mais aussi aux instants précédents. Les connaissances préalables et antérieures en ce qui concerne les positions et les vitesses des objets détectés sont toutes regroupées sous forme d'une nouvelle proposition logique qu'on note π. En conséquence, la probabilité d'une proposition a s'exprime de manière conditionnée par π : $P(a|\pi)$.

Toujours en DATMO, on s'intéresse beaucoup plus aux probabilités conditionnelles de conjonctions $P(a \wedge b|\pi)$, de disjonctions $P(a \vee b|\pi)$, de négation $P(\neg a|\pi)$ et enfin à la probabilité de a conditionnée à la fois par π et par une autre proposition b : $P(a|\pi \wedge b)$.

2.3.4 Règles et postulats de l'inférence

Larousse définit l'inférence comme suit : «Opération par laquelle on passe d'une assertion considérée comme vraie à une autre assertion au moyen d'un système de règles qui rend cette deuxième assertion également vraie». On peut remplacer le mot assertion par proposition logique et donner une définition plus mathématique de l'inférence : « l'inférence désigne les actions de mise en relation d'un ensemble de propositions, aboutissant à une démonstration de vérité, de fausseté ou de probabilité, sous la forme d'une nouvelle proposition».

L'observation incomplète de l'environnement du robot et les multiples incertitudes nous imposent un raisonnement bayésien en DATMO. On parle alors d'inférence bayésienne qui est une des méthodes d'inférence permettant de déduire la probabilité d'un événement à partir de celles d'autres événements déjà évaluées. Elle s'appuie principalement sur le théorème de Bayes. Ce raisonnement probabiliste se base sur les deux règles suivantes :

- Règle de la conjonction :

$$\begin{aligned} P(a \wedge b|\pi) &= P(a|\pi)P(b|a \wedge \pi) \\ &= P(b|\pi)P(a|b \wedge \pi). \end{aligned} \tag{2.1}$$

- Règle de la normalisation :

$$P(a|\pi) + P(\neg a|\pi) = 1. \tag{2.2}$$

Ces deux règles sont les éléments de bases de toutes les règles d'inférence probabiliste. Pour rester dans la problématique du DATMO où ces postulats sont utilisés et implémentés dans un cadre plus large en remplaçant les propositions logiques par des variables aléatoires discrètes.

Dans ce cas là, on exprime les régles (2.1) et (2.2) comme suit :

- Règle de la conjonction pour les variables aléatoires :

$$\begin{aligned} P(X \wedge Y|\pi) &= P(X|\pi)P(Y|X \wedge \pi) \\ &= P(Y|\pi)P(X|Y \wedge \pi). \end{aligned} \tag{2.3}$$

- Loi de la normalisation pour les variables :

$$\Sigma_X P(X|\pi) = 1. \tag{2.4}$$

- Loi de marginalisation pour les variables :

$$\Sigma_X P(X \wedge Y|\pi) = P(Y|\pi). \tag{2.5}$$

2.4 Programme bayésien

En anglais *« Bayesian program »*, qui sera noté BP dans la suite de ce mémoire, est défini comme étant un formalisme générique pour construire des modèles probabilistes et pour résoudre les problèmes d'inférence et de décision qui y sont liés [17]. Ces formalismes sont très utilisés en robotique mobile pour diverses applications dont le DATMO. De manière générale, un BP est composé de deux parties qui sont «la description» et «la question».

2.4.1 La description

La description est la notion centrale d'un système de programmation probabiliste. C'est par son biais que le programmeur décrit les connaissances préalables dont il dispose, soumet ces connaissances à l'expérience et enfin les restitue sous la forme d'une question probabiliste. Ces trois phases forment la brique de base pour toute programmation bayésienne d'un robot. On les résume comme suit :

- Spécification des connaissances préalables C.
- Identification des valeurs des paramètres des distributions de probabilités.
- Utilisation de la description [32].

Une description est notée formellement par la probabilité conjointe d'un ensemble de variables $X_1, \ldots X_n$, déterminée au vu des connaissances préalables spécifiées par le programmeur (notées C) et d'un ensemble de données expérimentales Z :

$$P(X_1 \wedge ... \wedge X_n|C\ Z). \tag{2.6}$$

Malheureusement, l'implémentation de cette distribution conjointe est en général trop complexe [17]. On remarque que l'ensemble des connaissances devant être fourni se trouve circonscrit par les connaissances préalables C (comme la position ou la vélocité aux instants précédents d'un obstacle déjà détecté) et le jeu de données expérimentales Z (comme les nouvelles acquisitions du télémètre laser ou les informations concernant l'odométrie du véhicule).

Cette étape de description est elle-même divisée en deux sous étapes par les programmeurs bayésiens : une étape de spécification et une étape d'identification. Sur la figure ci-dessous on donne un exemple de BP :

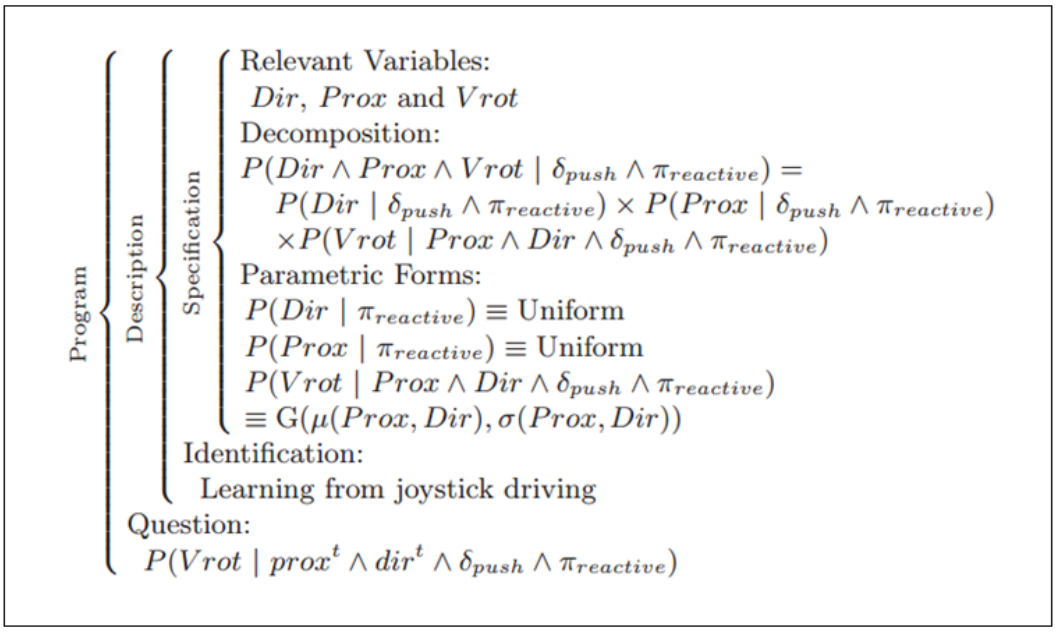


FIGURE 2.4 – Exemple d'un BP générique

La spécification

Le programmeur énonce clairement et explicitement, lors de cette phase, ses connaissances et celles qui résultent d'un jeu particuliers de données expérimentales. Il s'agit de l'étape la plus délicate en programmation et au cours de laquelle le programmeur spécifie les trois points suivants :

• Le choix des variables pertinentes.

• L'expression de la dépendance entre les différentes variables retenues sous la forme d'un produit de distributions de probabilité élémentaires.

• La forme paramétrique associée à chacune de ces distributions élémentaires.

Le programmeur commence donc par spécifier les variables à manipuler. Ensuite, il exprime formellement les dépendances entre celles-ci par des lois de probabilités conjointes. Cette structure de dépendance permet une décomposition en produit de distributions plus simples et par la suite élimine des termes dont on sait par des connaissances préalables ou pratiques qu'ils sont conditionnellement indépendants.

Pour éclaircir ce point on considère le cas d'un programmeur en DATMO qui retient trois variables pour une description : X, Y et V. La règle de conjonction (2.3) exprimée auparavant nous permet d'écrire :

$$\begin{aligned} P(X \wedge Y \wedge V|C) &= P(X|C)P(Y \wedge V|X \wedge C) \\ &= P(X|C)P(Y|X \wedge C)P(V|X \wedge Y \wedge C). \end{aligned} \tag{2.7}$$

En tout pour le choix de trois variables de description X, Y et V et une variable de connaissances préalables C on peut énoncer 13 formulations différentes. Si des considérations pratiques nous permettent d'affirmer que la valeur de la variable V ne dépend pas de Y alors

on peut ainsi simplifier (2.7) pour obtenir :

$$P(X \wedge Y \wedge V|C) \quad = \quad P(X|C)P(Y|X \wedge C)P(V|X \wedge C). \tag{2.8}$$

On vient de voir comment utiliser les connaissances préalables de dépendances avec un choix adéquat de la distribution conjointe afin d'exprimer cette dernière sous forme de produit de distributions plus simples. C'est ce qui sera exploiter dans le module DATMO.

Dans un second lieu, on exploite les connaissances préalables d'observation afin de rendre effective la description. En d'autres termes, on va préciser les valeurs associées à chacune de ces distributions probabilistes élémentaires en se basant sur l'évolution des données d'observation acquises.

L'identification

En partant de l'expression de probabilité conjointe globale exprimée par la relation (2.6), et en exploitant les connaissances préalables de dépendances et d'observations, on parvient à formuler une description basée sur des produits de distributions de probabilités plus simples que l'expression de départ. Par la suite, à chaque distribution de probabilité on associe une forme paramétrique particulière. Les plus utilisées en robotique sont : la loi uniforme, la loi de succession de Laplace et la loi normale (ou loi de Gauss). L'étape d'identification consiste à fixer les valeurs de ces formes paramétriques.

Dans le cas d'une loi Normale (une Gaussienne) on ne retient des données expérimentales que la moyenne des valeurs rencontrées et leur variance. Les valeurs des paramètres sont soit données a priori, soit identifiées sur la base d'un jeu de données expérimentales provenant de l'observation d'un même phénomène. De ce jeu de données, un certain nombre d'observables sont évaluées (par exemple la moyenne empirique des moments d'ordre 1 et 2 d'une variable). Sur la base de ces observations, les paramètres des distributions spécifiées par les connaissances préalables sont identifiés (par exemple valeur moyenne et écart-type) [32]. Ainsi s'achève la partie de description.

2.4.2 Utilisation et formulation de la question

Au terme des phases de spécification et d'identification décrites précédemment, on dispose d'une description bayésienne complètement définie.

En partant toujours de la distribution $P(X_1 \wedge ... \wedge X_n|Z\ C)$, poser une question consiste à chercher la distribution de probabilité d'un certain nombre de variables de la description connaissant les valeurs d'autres variables. En d'autres termes, on partitionnera l'ensemble $[X_1, ..., X_n]$ en trois sous groupes.

- Les variables recherchées : « Search »
- Les variables connues : « Known »
- Les variables libres : « Free »

Pour une certaine valeur de la variable « Known », une question se définit par la distribution probabiliste suivante :

$$P(X_1 \wedge ... \wedge X_k | X_{k+1} \wedge ... \wedge X_n \; Z \; C), \tag{2.9}$$

avec :
$[X_1 \ldots X_k]$ = « Search »
$[X_{k+1} \ldots X_n]$ = « Known ».

L'intérêt de l'inférence bayésienne, expliqué précédemment, est d'implémenter de manière efficace toutes ces distributions probabilistes dans un programme bayésien qui réponde à la question posée. Dans le cas du DATMO, ces questions sont relatives aux positions et aux vélocités des obstacles détectés.

2.5 Formulation Bayésienne du DATMO

On a justifié l'interprétation bayésienne du problème du DATMO par l'existence des incertitudes et des incomplétudes des informations acquises. Si on prend l'exemple du Robucar qui perçoit son environnement à travers le capteur laser LMS511, on constate que les données transmises sont mieux interprétées sous forme de distribution de probabilité qu'on note :

$$bel \;\; := \;\; P(S|Z), \tag{2.10}$$

S : l'état du système ou de l'environnement.
Z : l'ensemble des mesures acquises.
La distribution de probabilité $bel = P(S|Z)$ est appelée *posterior distribution* ou *Bayesian Belief*.

2.5.1 Equation bayésienne récursive du DATMO

L'environnement du robot (S) change constamment en fonction du temps et notre but sera d'estimer cette croyance (Bayesian Belief) à un instant t :

$$bel_t \;\; = \;\; P(S_t|Z_1 ... Z_t), \tag{2.11}$$

Le comportement du système (S) est décrit par deux lois de probabilités :

- Le modèle de mesures (measurement model) : Exprime la manière dont les mesures sont obtenues et qu'on note $P(Z|S)$.
- Le modèle dynamique (dynamics model) : Exprime la manière dont le système S évolue entre deux instants et qu'on note $P(S_t|S_{t-1})$.

Etant données ces deux lois et les nouvelles mesures acquises à l'instant t, la croyance (bayesian beleif) peut être implémentée de manière récursive en utilisant un algorithme de

filtre bayésien en rapport à l'équation récursive suivante (μ est une constante de normalisation) :

$$bel_t \quad = \quad \mu P(Z_t|S_t) \int P(S_t|S_{t-1}) bel_{t-1} dS_{t-1}, \tag{2.12}$$

L'équation (2.12) est fondamentale en DATMO.

2.5.2 Assignement des variables en DATMO

Souvent en DATMO, le système (S) représente l'ensemble des obstacles mobiles détectés par le Robucar et que l'on posera $\{T^1, \ldots, T^{K_t}\}$. Le nombre d'obstacles K_t dépend du temps t du fait que des obstacles entrent et sortent constamment du champ de vision du robot.

Pour tout obstacle T, les paramètres qu'on veut estimer sont :

- Sa position momentanée $X_t = (x_t, y_t, \theta_t)$ formée par sa position cartésienne (x_t, y_t) et son orientation θ_t à l'instant t,
- Sa vélocité v_t ,

d'où l'état bayésien correspondant à chaque cible séparée : $S\prime_t = (X_t, v_t)$. D'autres notations peuvent être rencontrées dans la littérature. Certains auteurs considèrent la position momentanée comme étant l'unique position cartésienne (x_t, y_t) et insère l'orientation θ_t dans l'estimation de la vélocité v_t.

On peut ajouter d'autres variables qui décrivent la forme des obstacles dans un but de «classification» de ces derniers. La séparation entre SLAM, DATMO et classification étant déjà détaillée dans le chapitre 1, on va se contenter de cette formulation dans la suite de ce mémoire.

On remarque aussi que le modèle dynamique (dynamic model) dont on a parlé précédemment, peut se décomposer en deux modèles pour chaque cible détectée : Le modèle de vélocité (velocity model) exprimé par sa distribution $P(v_t|v_{t-1})$ et le modèle de mouvement (motion model) exprimé par : $P(X_t|X_{t-1} \ v_t)$. L'état bayésien global se compose des états bayésiens de toutes les cibles suivies : $S_t = (S_t^1, \ldots, S_t^{K_t})$. Enfin, à chaque cycle, on acquiert un nouveau fichier de mesures du capteur laser que l'on note Z_t.

2.5.3 Modélisation probabiliste des capteurs

L'équation (2.12) qui est fondamentale en DATMO fait intervenir le terme $P(Z|S)$ représentant le modèle de mesures (sensor model ou measurement model). Cette distribution probabiliste exprime le mécanisme de mesure d'une manière statistique, c'est-à-dire la probabilité de la mesure en fonction de la distance réelle à laquelle se trouve l'obstacle. En DATMO, il s'agit de déterminer l'ensemble de mesures Z_t en se donnant une position X_t d'un objet ou d'un obstacle suivi T^{K_t}.

La détermination exacte d'un tel modèle n'est pas possible à cause des multiples effets inconnus qui peuvent influencer le fonctionnement et les performances du capteur utilisé [22]. On peut citer par exemple l'influence du degré de luminosité à l'heure d'utilisation, la

poussière, le brouillard et la pluie, mais aussi certains phénomènes physiques en rapport avec les caractéristiques et le fonctionnement du capteur utilisé.

Au chapitre 1, on a comparé les télémètres laser aux caméras. Ces deux capteurs sont considérés comme les plus performants et pourtant ils auront des caractéristiques probabilistes très différentes en fonction des conditions d'emploi. On prend le cas d'une représentation gaussienne pour modéliser la probabilité $P(Z|S)$: pour une caméra utilisée avec une bonne luminosité ou un télémètre laser dans un milieu non-réfléchissant, la variance de la distribution $P(Z|S)$ sera très faible. Dans les cas contraires cette caractéristique peut se dégrader. Dans la littérature, on trouve des modèles un peu plus élaborés qu'une distribution gaussienne. En général, c'est une combinaison de lois qui modélisent divers phénomènes :

- la mesure effective de l'obstacle visé, modélisé par une gaussienne en général. θ_t à l'instant t,
- la perception d'un obstacle imprévu, modélisé par une loi décroissante,
- la non perception de l'obstacle qui donne une mesure à la distance maximale du télémètre, modélisé par un pic. [57]

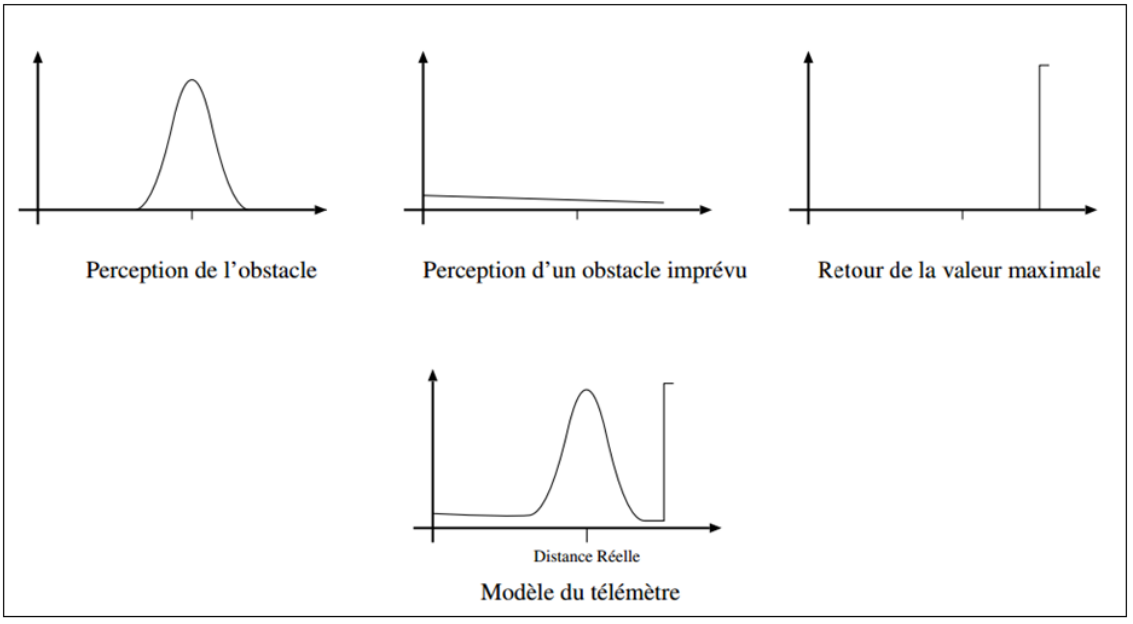


FIGURE 2.5 – Exemple de modèle probabiliste d'un télémètre laser

2.6 La modélisation de la scène

En robotique mobile, on entend par « scène » l'ensemble de l'environnement externe au robot. Ce paragraphe s'intéresse aux différentes modélisations de la scène et aux modèles dynamiques (dynamic models) les plus utilisés pour la modélisation des obstacles présents dans cette scène.

2.6.1 Modélisation par Clustering

Le premier modèle, «Cluster-Based Model», représente la plus ancienne et la plus simple modélisation de scènes. Cette représentation se base sur le regroupement des points de mesures qui sont susceptibles d'appartenir à un même objet ou obstacle. Plusieurs méthodes et algorithmes existent pour traduire les données brutes acquises par le capteur du robot en une représentation par regroupement. Mais cette simple interprétation des données s'avère insuffisante pour reconstituer fidèlement la scène réelle. En effet, on peut imaginer beaucoup de situations courantes où cette logique de regroupement de points assez proches associés à un obstacle unique, peut nous induire en erreur. De même, cette méthode peut séparer les parties d'un unique obstacle en plusieurs objets.

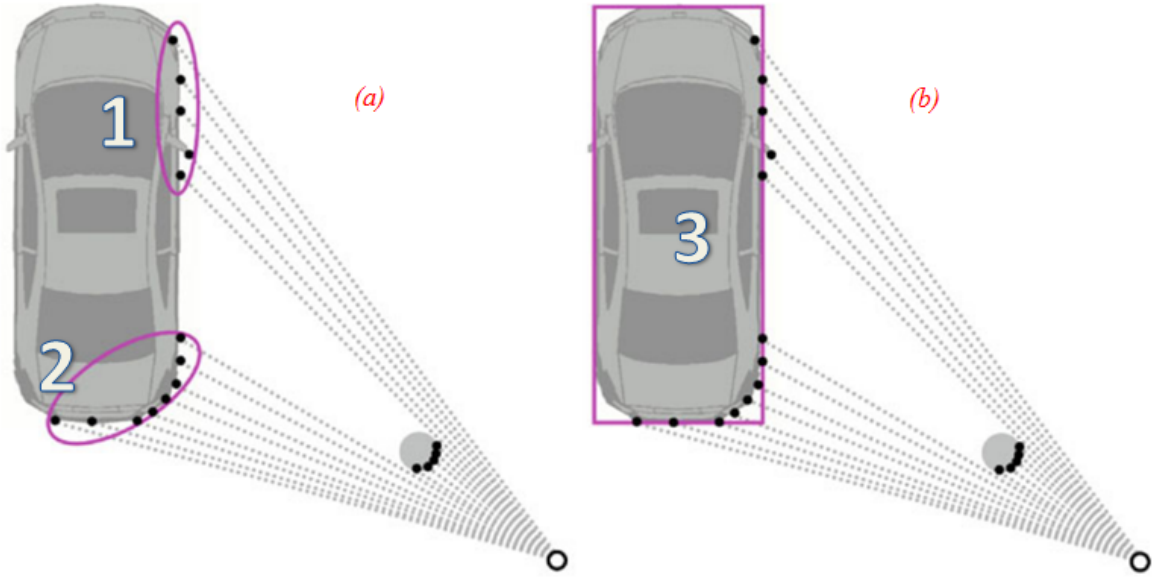


FIGURE 2.6 – (a) : Cluster-Based Model, (b) : Geometric-Based Model

Sur la figure 2.6, les obstacles 1 et 2 sont considérés par le robot comme des objets différents et indépendants alors qu'en réalité ils font partis du même obstacle.

Une nouvelle représentation, la **«Geometric Model»**, est apparue comme une alternative pour remédier à ces problèmes pratiques. Elle tient compte des formes géometriques des obstacles, les plus utilisées étant le rectangle et le cercle. Des variables supplémentaires sont ajoutées au modèle bayésien, décrit précédemment, afin d'estimer la forme géométrique en même temps que l'estimation de la position et de la vélocité.

2.6.2 Modélisation par grille d'occupation

La troisième et dernière représentation est la « **Grid-Based Model** ». Cette modélisation est introduite pour la première fois en 1988, dans un contexte de fusion de données par Moravec [18]. Ensuite Elfes l'a utilisé en détection d'obstacles statiques dans ses travaux en 1989 [10]. L'idée est de diviser l'environnement entourant le robot en une grille formée

par plusieurs cellules. La grille d'occupation offre un moyen robuste pour une variété de problèmes en localisation, perception et planification [10].

On associe à chaque cellule de la grille une probabilité d'occupation $P(O_i)$ entre 0 et 1. Une probabilité $P(O_i) = 0$ signifie que la cellule est forcément vide (free), une probabilité $P(O_i) = 1$ signifie qu'elle est sûrement occupée (occuped) et pour les cellules dont on n'a pas d'information (par exemple l'espace caché derrière un obstacle) on donne généralement une probabilité nulle ou bien médium $P(O_i) = 0.5$ (unkonwn).

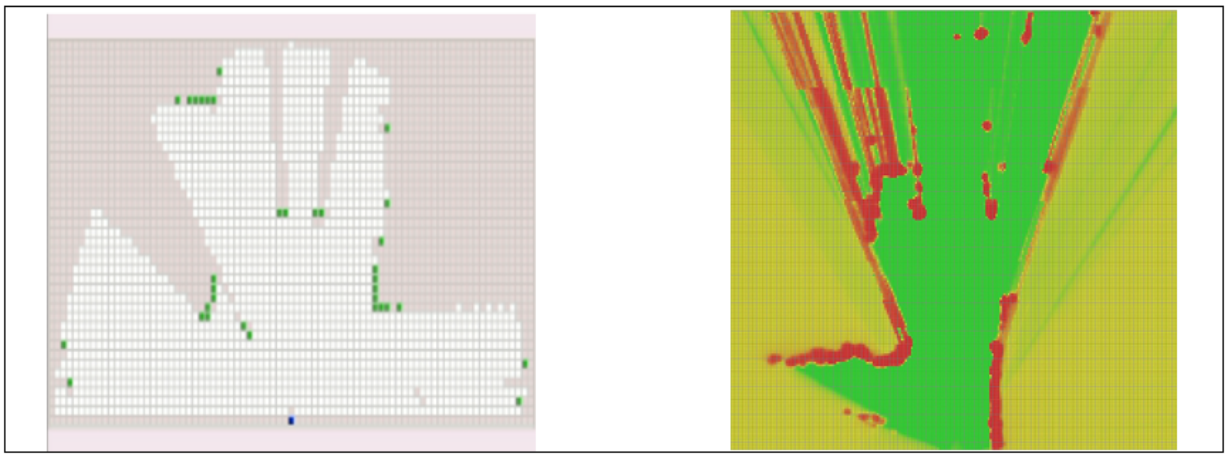


FIGURE 2.7 – Modélisation de scènes par des grilles d'occupation

En DATMO cette grille est localement associée au robot (grille locale) car le but est de repérer les dangers potentiels pour pouvoir les éviter. En revanche, en SLAM on implémente des grilles globales pour représenter l'environnement car le but est de reconstituer des cartes de l'ensemble des lieux [40].

La spécification de la taille de la grille et les dimensions de chaque cellule se font selon des critères et des compromis entre temps de calcul et précision. Le choix d'un nombre important de cellules de petites dimensions permet une représentation et une interprétation plus précise. Néanmoins, ceci demande d'importantes capacités de calcul et un temps de traitement intolérablement long. En effet, le robot doit détecter vite, estimer la position de l'objet détecté pour les prochaines secondes et enfin planifier et régénérer les mouvements adéquats pour éviter la collision. A titre comparatif, le réflexe d'un conducteur humain au volant est estimé à $1s$. C'est-à-dire que si un piéton traverse la route, en une $1s$ le chauffeur réagira par un coup de frein ou de volant. En faisant un bon compromis les véhicules autonomes peuvent être plus rapides.

Une différence fondamentale entre cette modélisation et les deux représentations précédentes est que les notions telles que « objet » ou « track » n'existent pas dans la grille. Elles sont remplacées par la notion d' « occupation » où on estime l'occupation (et la vélocité) de chaque cellule de la grille sans chercher à associer des cases entre-elles pour reformer l'objet ou l'obstacle qui existe en réalité. Les défenseurs de cette méthode, tel que Perrollaz dans [40], expliquent qu'il est préférable d'avoir une représentation riche en données sensorielles

qui est la grille plutôt qu'essayer de reconstituer les objets tels qu'on les voit en réalité. Surtout qu'en DATMO, la reconstitution de scènes n'est pas l'objectif premier du moment que le robot arrive à panifier des mouvements sécurisés [22].

Enfin, un autre avantage pour les grilles d'occupation est que ces dernières sont particulièrement commodes pour le « data-fusion » c'est-à-dire la fusion de données lorsque plusieurs capteurs sont simultanément utilisés en perception par un même robot (data fusion : Moravec 1988 [18]).

2.7 Les modèles dynamiques

Dans les paragraphes précédents, nous avons modélisé les capteurs et la scène. D'autres modèles existent dans la littérature pour représenter les obstacles détectés par le robot. A titre de rappel, le « dynamics model » noté $P(S_t|S_{t-1})$, exprime la manière dont le système S évolue entre deux instants. Ce modèle se compose du velocity model : $P(v_t|v_{t-1})$ et du motion model : $P(X_t|X_{t-1}\ v_t)$.

En DATMO, il est commun d'utiliser un *«constant velocity model»* qui suppose une vélocité constante entre deux instants $t-1$ et t. Dans ce cas, l'évolution instantanée de la vélocité se traduit par l'addition d'un bruit aléatoire maximisé par l'accélération crête a_{max}. Plus précisément, Δv est échantillonné à partir d'une distribution normale $N(0, a_{max}\Delta t)$. Quant au « motion model », on retrouve trois modèles de base dont le plus utilisé en modélisation des piétons est le « Brownian Motion Model ». Ce dernier est commode pour les changements rapides de direction et de vélocité qui caractérisent les êtres humains. Dans ce modèle, la position évolue par ajout d'un bruit gaussien de moyenne nulle. La variance du bruit augmente avec le pas d'échantillonnage Δt.

Le deuxième modèle est le « Linear Motion Model », ce dernier correspond mieux aux véhicules du fait que leur mouvement est plus prédictif. En effet, à cause de son importante inertie et des contraintes non-homonymiques, une voiture ne peut pas changer sa direction ou sa vélocité d'une manière aussi impulsive qu'un être humain. Dans ce cas, le « Linear Motion Model » modélise parfaitement le comportement de cette classe d'obstacles. Ce modèle suppose une variation de la direction $\Delta\theta$ échantillonnée à partir d'une distribution normale $N(0, \Delta\theta_{max}\Delta t)$. En 2005, Sébastien Thrun de la Stanford University a annoncé son « Bicycle Motion Model » qui est une combinaison des caractéristiques des deux modèles précédents.

2.8 Les différentes approches du DATMO

En se basant sur la formulation bayésienne énoncée précédemment, quelques approches de résolution de l'équation fondamentale

$$bel_t \quad = \quad \mu P(Z_t|S_t) \int P(S_t|S_{t-1}) bel_{t-1} dS_{t-1}, \tag{2.13}$$

existent. Mais très vite d'autres difficultés apparaissent dans chacune d'elles et font en sorte que les robots prennent parfois des décisions erronées. A ce jour on reste encore loin du véhicule complétement autonome roulant en toute sécurité dans nos villes et en respectant

le code de la route. Ces approches restent à améliorer alors que le véhicule autonome commercialisé au grand public est annoncé pour 2020. Dans ce paragraphe, nous allons voir les trois approches existantes pour réaliser un module de DATMO. Il se trouve que les deux premières approches reposent sur un même raisonnement contrairement à la troisième qui est fondamentalement différente des autres.

La première approche, appelée approche traditionnelle, va de la segmentation des données à l'application d'un filtre paramétrique de la famille des filtres de Kalman en passant par des méthodes d'association de données qui représentent la difficulté de cette approche.

La deuxième approche ressemble à la première mais elle se base sur les modèles géométriques et ne nécessitent pas de traitement préalable des données acquises par les capteurs. Cette méthode n'utilise donc pas de segmentation ou d'association de données mais par contre elle nécessite l'application d'un filtre plus puissant et non-paramétrique qui est le filtre à particules.

Enfin, la troisième approche est complétement différente des deux précédentes. C'est l'approche par grille d'occupation où l'environnement externe entourant le robot est divisé en un certain nombre de cellules. Contrairement aux deux méthodes précédentes, la notion d' «objet» n'a pas d'intérêt dans les grilles d'occupation. La figure suivant, tirée de *Petrovskaya* en 2012 [22], résume les trois approches précédentes.

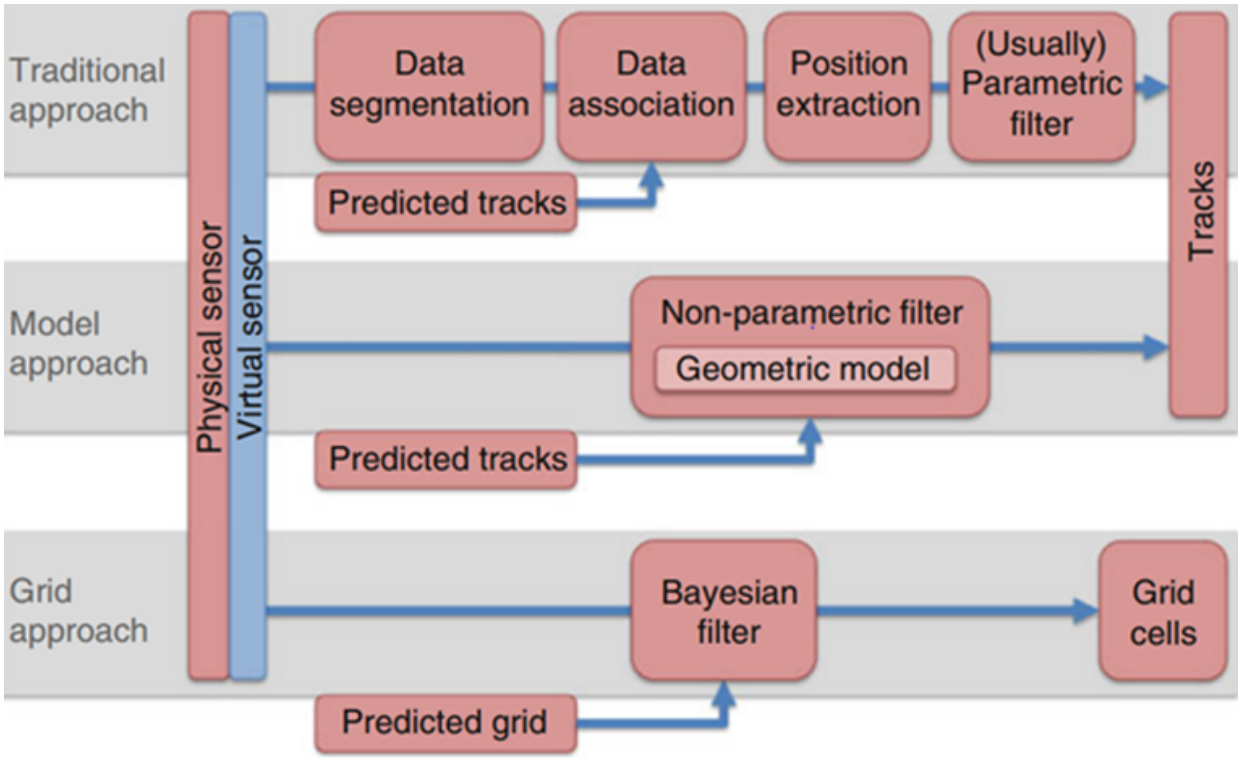


FIGURE 2.8 – Les trois approches bayésiennes du DATMO, *Anna Petrovskaya – Artificial Intelligence Laboratory, Stanford University, USA – 2012*

2.8.1 Approche traditionnelle

Historiquement, l'approche traditionnelle est la première à être utilisée. On note que le gros du travail est effectué avant l'application du filtre bayésien qui est, dans ce cas, un filtre paramétrique type Kalman. Les données acquises par les capteurs sont d'abord segmentées, puis associées aux obstacles déjà détectés (suivis). C'est cette deuxième étape, appelée data association, ou association des données qui représente toute la difficulté de cette approche.

- *Data Segmentation :*

  On a déjà survolé ce point de la segmentation lorsqu'on a exposé le cluster model en modélisation de scène. Pour cette première approche, la segmentation est une étape essentielle même si elle ne donne pas de reconstitution exacte de l'environnement réel. L'application classique se fait soit par repérage de points de ruptures (break points) ou par les algorithmes PDBS (point distance based-method) comme le FOF (friends of friends) habituellement utilisé en cosmologie. La référence [59] regroupe l'ensemble des méthodes utilisées en segmentation des données laser.

- *Data association :*

  A l'issue de la première étape, les données laser brutes sont regroupées en plusieurs segments. Ces groupes ont besoin d'être associés à des obstacles déjà détectés par le passé ou bien à de nouveaux obstacles qui viennent d'apparaitre pour le robot. Une des méthodes d'association est la « Nearest Neighbor Method » (NN). Elle consiste à associer les groupes issus du Clustering aux obstacles les plus proches. Pour que cette association se fasse sans ambiguïté il faut que le traitement soit très rapide. Une amélioration de cette méthode est la « Global Nearest Neighbor Method » (GNN) qui s'assure que chaque groupe (cluster) n'est associé qu'à une cible « trackée ».

  Des algorithmes plus avancés s'utilisent dans des environnements ambigüs comme la MHT « Multiple Hypothesis Tracking » ou l'algorithme du JPDAF « Joint Probability Data Association Filter ».

- *Parametric filter* :

  Le « Kalman Filter » (KF) est une technique largement employée dans l'estimation d'états dynamiques observés à travers des mesures bruitées [49]. Ce filtre est récursif, ce qui veut dire qu'à chaque étape il a besoin de la sortie de l'étape précédente pour en faire une prédiction. A l'issue des deux étapes précédentes, un certain nombre d'obstacles est détecté. Pour chacun d'eux, on désire estimer l'état $S_t = (X_t, v_t)$.

  A noter que, dans le cas du « constant velocity model », les matrices de transition et de covariance du bruit sont relativement faciles à déterminer et à implémenter dans le KF.

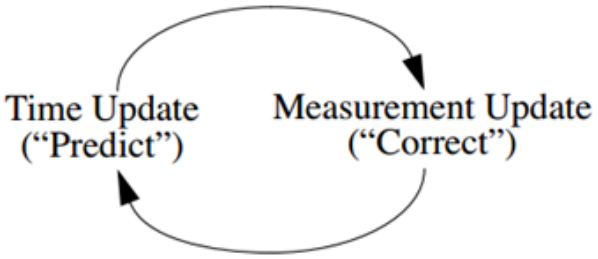


FIGURE 2.9 – Cycles d'estimation et prédiction lors d'un KF appliqué en DATMO classique

La finalité de l'approche classique du DATMO est d'associer un KF à chaque cible détectée. A chaque nouveau scan du laser, les objets « trackés » sont mis à jour par la séquence donnée à la figure 2.10.

```
1:  for each TRACKER do
2:  Calculate Kalman prediction state;
3:  MD_all ⇐ Mahalanobis Distances of all measurement data;
4:  MD ⇐ min(MD_all);
5:  if MD ≤ threshold_1 then
6:  update Kalman Filter with measurement;
7:  hidden = 0;
8:  else
9:  hidden = hidden + 1;
10: end if
11: if TRACKER is hidden for more than threshold_2 then
12: Delete TRACKER
13: end if
14: end for
15: for each unmatched measurement do
16: Create a new TRACKER
17: end for
```

FIGURE 2.10 – Séquence d'exécution en approche classique à chaque nouvelle itération

La deuxième approche repose sur l'application directe d'un filtre plus puissant que le KF qui est le « Particle Filter » (PF). Ce dernier s'applique toujours avec une modélisation géométrique de la scène en DATMO. Cette alternative à l'approche classique présente des caractéristiques intéressantes et un faible taux de fausses détections. Cette amélioration est essentiellement due au remplacement des anciennes méthodes de « data association » par la modélisation géométrique de la scène. Néanmoins, l'application de cette deuxième approche

nécessite une puissance de calcul très importante qui la rend parfois impossible à implémenter sur un véhicule autonome en temps réel.

2.8.2 Approche basée sur les grilles d'occupation

Plusieurs facteurs indiquent que cette récente approche est la plus prometteuse pour l'avenir du DATMO. En effet, la méthodologie de raisonnement se base sur une modélisation de l'environnement par une grille d'occupation et n'utilise pas de « data-association ». Ce qui évite les ambiguïtés souvent rencontrées en approche traditionnelle. Aussi, contrairement aux filtres particulaires, la puissance requise des calculateurs peut être modérée. On fait d'abord un compromis entre précision et temps de calcul puis un choix adéquat de la taille de la grille et du nombre de ses cellules. Enfin, l'application de l'algorithme *«Fast Classification of static and dynamic environment for Bayesian Occupancy Filter»* détaillé dans [52], rend les performances de cette approche très attractives.

L'implémentation pratique de l'algorithme précédent réduit considérablement le temps d'exécution des cycles d'estimation et de prédiction dans le filtre bayésien ainsi que le taux de fausses détections comme l'indique le rapport annuel en 2012 de l'équipe-projet E-motion de l'INRIA [29]. L'inconvénient actuel de cette méthode est que peu de travaux ont été réalisés jusqu'à présent. Dans ce mémoire, on a choisi d'illustrer le filtre bayésien le plus populaire utilisé dans les grilles, intitulé *«The Bayesian Occupancy Filter (BOF)»*.

Bayesian Occupancy Filter (BOF)

C'est un type de filtres qui utilise deux étapes dans le mécanisme d'estimation des filtres bayésiens mais de façon particulière. C'est en fait une adaptation au cas du DATMO spécialement. A chaque cycle, le BOF estime l'état d'occupation de la grille qui modélise l'environnement autour du robot. Cette estimation est en réalité une combinaison entre une étape de prédiction (historique) et une étape d'estimation incluant les nouvelles mesures acquises. L'espace d'état du BOF est une grille 2-D. A chaque cellule de cette grille on associe une probabilité d'occupation et une distribution probabiliste de vélocité.

La formulation utilisé dans ce mémoire est celle énoncée dans [22], [56], [47], [17] et [43] en définissant les variables d'état comme suit :

- C est un indice qui identifie chaque cellule de la grille $2D$.
- A est un indice qui identifie toute cellule antécédente possible de la cellule C de la grille.
- $Z_t \in \hat{Z}$, telle que Z_t est la variable aléatoire modélisant les mesures du capteur relatives à la cellule C.
- $V \in \hat{V} = \{v_1, ..., v_n\}$, où V représente la variable aléatoire de la vélocité de la cellule C dont les valeurs possibles ont été discrétisées en n valeurs.
- $O, O^{-1} \in \hat{O} = \{occ, emp\}$: O représente la variable aléatoire de l'état de la cellule C qui peut être occupée ou libre. Quant à O^{-1}, c'est aussi une variable aléatoire qui représente l'état d'une cellule antécédente de O.

La loi de Bayes et les postulats de dépendance nous permettent de décomposer la distribution de probabilité conjointe associant les variables précédentes comme suit :

$$P(C, A, Z, O, O^{-1}, V) = P(A)P(V|A)P(C|V\ A)P(O^{-1}|A)P(O|O^{-1})P(Z|O\ V\ C). \quad (2.14)$$

On a expliqué que l'intérêt du filtrage par BOF est d'estimer l'état d'occupation et les distributions probabilistes de vélocité pour toute cellule dans la grille. Ceci s'exprime par la distribution de probabilité qu'on note : $P(O\ V|Z\ C)$.

Dans le contexte du BOF, l'étape de prédiction consiste en la propagation de l'occupation des cellules pour toute valeur de vélocité et cellule dans le BOF et qui s'exprime par la distribution $P(O\ V|C)$. Durant l'étape d'estimation, $P(O\ V|C)$ est mise à jour en prenant en considération les nouvelles observations $P(Z|O\ V\ C)$ afin d'obtenir l'estimation du filtre bayésien $P(O\ V|Z\ C)$. Ce résultat est ensuite réinjecté dans l'étape de prédiction au prochain cycle.

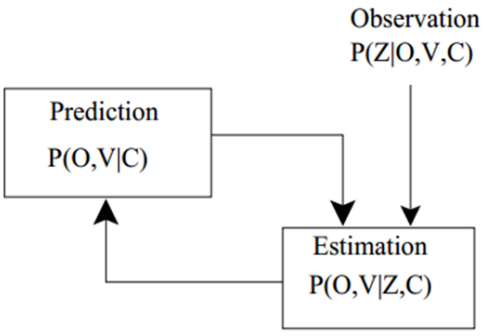


FIGURE 2.11 – Logique du filtrage bayésien appliquée à la grille d'occupation

On a parlé des programmes bayésiens en début de chapitre et de la structure générale de ce type de programmes. Le BOF est un exemple assez complexe de ces modèles de programmations récents en informatique et qui nécessite une forme de décomposition en plusieurs sous programmes bayésiens lors de l'étape d'implémentation. Des programmes et des algorithmes viennent en appui pour rendre possible une telle implémentation. On verra lors du prochain chapitre comment on réduit la question à l'estimation de vitesse uniquement, l'occupation étant déterminée par l'implémentation d'un algorithme de séparation statique/dynamique. Le lecteur est invité à lire les rapports de l'équipe projet e-motion [] qui explique l'intérêt d'une telle simplification. Mais avant, il est important d'analyser l'approche globale donnée par le BP ci-dessous et de remarquer la différence fondamentale qui distingue cette approche de l'algorithme de filtrage de Kalman donné à la figure 2.9.

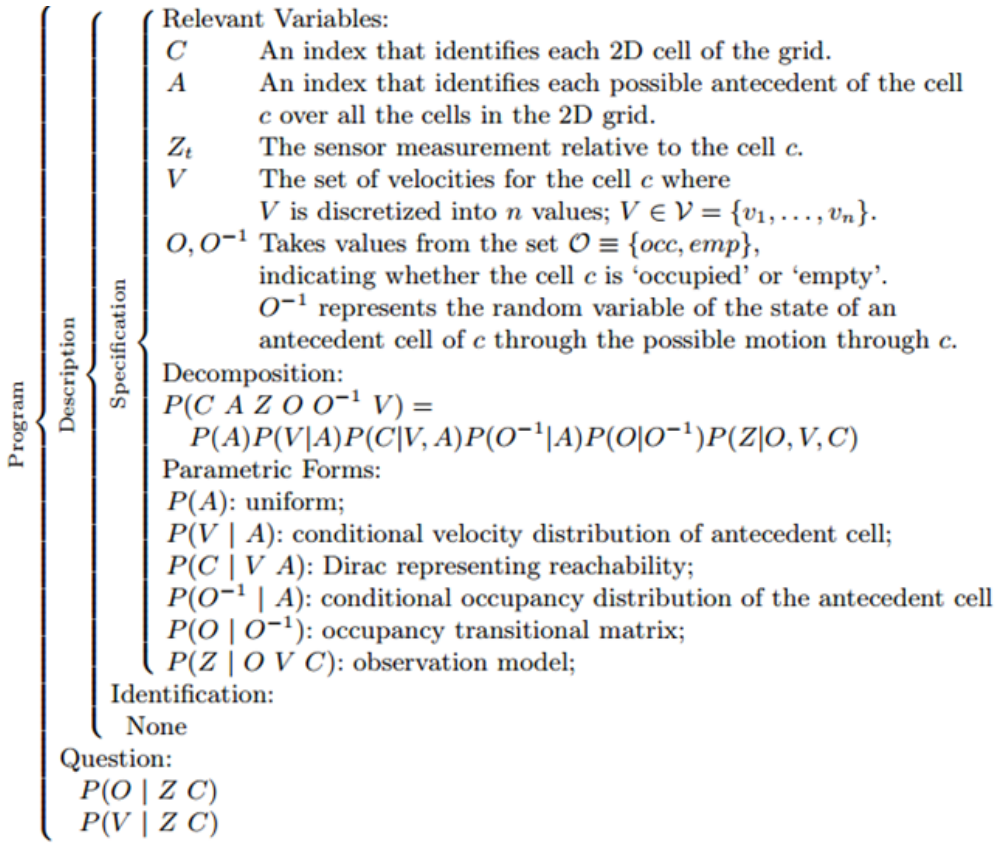


FIGURE 2.12 – Programme bayésien du BOF

2.9 Conclusion

Ce chapitre est un condensé d'informations et de techniques qui couvrent entièrement la formulation bayésienne du DATMO. Très peu de travaux semblables ont été réalisés dans un domaine qui cherche encore ses repères.

En robotique mobile, la nécessité de gérer la dimension incertaine devient incontournable. C'est par ces notions d'incomplétude et d'incertitude qu'on a entamé ce chapitre, le Robucar étant un parfait exemple d'application car les valeurs obtenues par ses capteurs sont bruitées. Les conséquences des commandes motrices ne sont jamais précisément celles escomptées, et les modèles sont entachés d'erreurs. L'approche descreptive est beaucoup plus adaptée dans ces cas là et l'élaboration des BP devient un moyen de communication efficace entre le robot et son concepteur.

La formulation mathématique bayésienne du DATMO a entrainé avec elle une réinterprétation générale de l'environnement interne et externe du Robucar. En effet,on a vu comment modéliser les capteurs du robot, la scène et les obstacles détectés par des représentations probabilistes. La modélisation par grille d'occupation induit à une une approche fondamentalement différente des deux autres approches. Néanmoins elle semble être la plus attractive pour plusieurs raisons. Dans le cas du DATMO, l'approche classique cause plusieurs pro-

blèmes en pratique malgré la grande efficacité du filtre de Kalman et de ses variantes. Ces problèmes proviennent malheureusement des autres étapes préalables comme le 'data association".

Le chapitre prochain est une application directe sur le Robucar. On a opté pour la méthode la plus récente, à savoir celle basée sur les grilles d'occupation. Mais on abordera aussi d'autres questions relatives au choix de la programmation bayésienne pour le Robucar, les techniques de SLAM, la localisation et la plannification... avec des comparaisons et des commentaires sur certaines notions abordées au premier chapitre à chaque fois que nos expériences les induisent.

Chapitre 3

Implémentation et expérimentation : Approche F+D et exigences en milieu dynamique

3.1 Introduction

Afin de valider les développements théoriques illustrés aux chapitres précédents, des simulations et des expériences ont été menées au centre de développement des technologies avancées CDTA.

Dans ce chapitre, on va d'abord justifier notre choix de programmation bayésienne dans le cas du Robucar par une expérience de *scene undestanding*. Puis on réalise une expérience en ICP-SLAM, qui est la première pour le Robucar dans un environnement dynamique, et qui montre la nécessité d'équiper ce dernier d'un module de DATMO non seulement pour les besoins en *mapping* mais aussi en planification de trajectoires. Mais avant cela, on présente le Robucar, l'environnement logiciel permettant la communication avec le robot et enfin les différents modules utilisés et qui interagissent avec le module du DATMO élaboré dans le chapitre 4.

3.2 Présentation du Robucar

Le Robucar est un véhicule automatique et intelligent, commercialisé par la société française ROBOSOFT. C'est un type de véhicule électrique et autonome destiné à évoluer dans des milieux sains ou hostiles pour accomplir des missions spécifiques comme le transport de matériaux dangereux. Ce robot a connu beaucoup de succès et plusieurs laboratoires de recherche l'utilisent comme plate-forme expérimentale : on citera le CDTA mais aussi l'Inria-Grenoble qui utilise la Cycab, plate-forme identique dans sa conception mais avec un look externe différent.

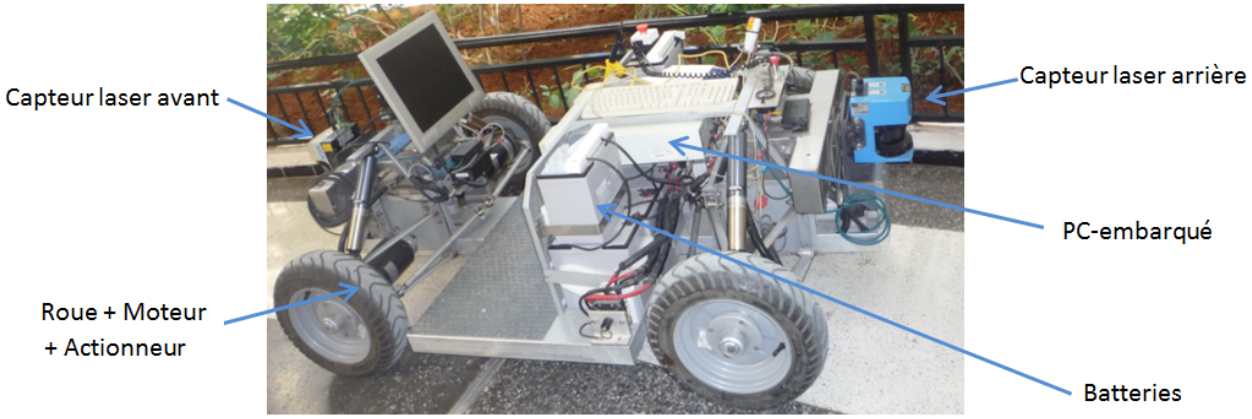


FIGURE 3.1 – Principaux constituants matériels du Robucar

3.2.1 Présentation du matériel

La plate-forme mobile

Sur l'image ci-dessus on voit le véhicule construit sur la base d'un châssis tubulaire. Plusieurs batteries assurent son autonomie énergétique. Deux capteurs laser y sont installés : un LMS511 à l'avant et un LMS200 à l'arrière du véhicule. Un joystick est utilisé pour conduire le robot en mode manuel. Des odomètres ou des codeurs incrémentaux sont associés à chaque roue. Le véhicule présente les caractéristiques générales et techniques suivantes :

Longueur totale	1836 mm.
Largeur totale	1306 mm.
Hauteur	616 mm.
Poids total avec batteries	310 Kg.
Motorisation	4 moteurs électriques de 1200 W.
Roues motrices	4 roues motrices et directrices.
Vitesse maximale	5m/s (18 Km/h).
Autonomie	2h d'utilisation continue.
Capacité d'accueil	2 personnes.
Conduite	Automatique et manuelle.

FIGURE 3.2 – Caractéristiques matériels du Robucar

Ce robot mobile présente des caractéristiques techniques qui le rendent plus performant et plus habile comparé aux voitures classiques. Il a trois modes de braquage illustrés sur la figure suivante :

- Mode simple braquage : la translation est faite par les quatre roues et la rotation par seulement les deux roues du train avant, les deux autres roues (train arrière) seront fixes.
- Mode double braquage : la translation est faite par les quatre roues, et la rotation par les quatre roues également (celles du train avant et arrière) mais dans un sens opposé.

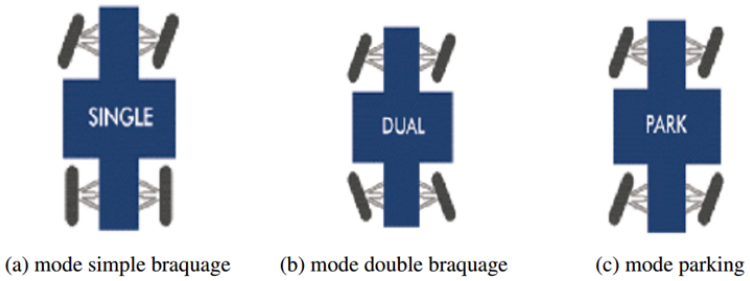


FIGURE 3.3 – Les modes de braquage du Robucar

- Mode parking : la translation et la rotation sont assurées par les quatre roues (celles du train avant et arrière) et les deux trains tournent dans le même sens.

Le capteur laser LMS511

On voit sur la figure ci-dessous le capteur LMS511 qui est placé à une hauteur de 30 cm du sol. Il est spécifié sur son datasheet qu'il peut couvrir une zone allant de 0 ° à 190 ° au maximum. Pour le cas du Robucar on limite cet angle à 180 ° .

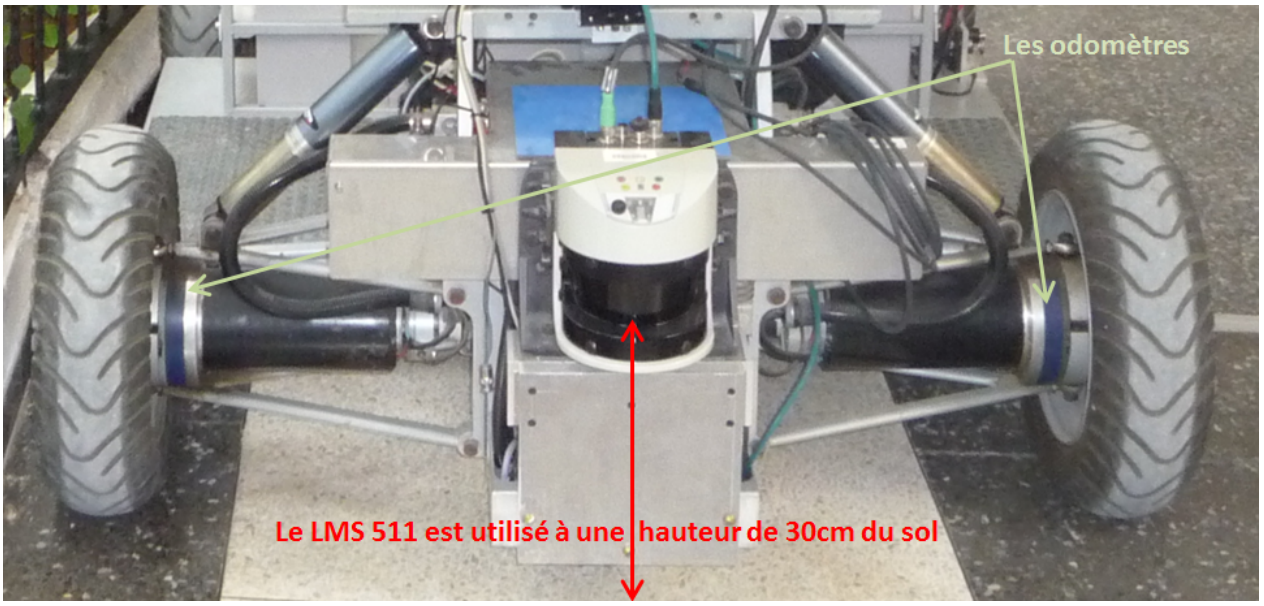


FIGURE 3.4 – Emplacement du LMS-511 sur le robot mobile

Dans le cadrede ce travail, le capteur laser est l'outil matériel le plus important. En effet, il est l'unique capteur extéroceptif utilisé en perception de l'environnement (Cf. chapitre 1), et il joue le rôle du capteur proprioceptif pour localiser le robot. Au premier chapitre, on a exposé les odomètres des véhicules autonomes et leur grande dérive en pratique. Dans les expériences qui vont suivre, le LMS est utilisé en ICP pour une meilleure localisation du Robucar.

Ce capteur fait partie de la famille des LMS*** de l'entreprise allemande SICK, l'un des leaders mondiaux de la fabrication des capteurs.

Quelques caractéristiques du Data-sheet du LMS 511 :

Type	LMS 511 SICK
Field of view	190°
Interface	RS-232/422 - Ethernet
Max range	80 m (souvent utilisé à 65m)
Angular resolution	0.25°, 0.5° or 1°

FIGURE 3.5 – Caractéristiques données par le fabricant du LMS 511, [70]

Principe de fonctionnement :

La fonction première du capteur LMS est de balayer l'environnement et d'extraire les obstacles détectés, en renvoyant leur position polaire (distance entre l'objet et le laser, angle correspondant) par la méthode suivante :

Une diode laser émet un faisceau qui est réfléchi par un miroir interne. La rotation de la tête du senseur tourne de l'axe vertical modifie l'orientation du faisceau et crée les angles de balayage.

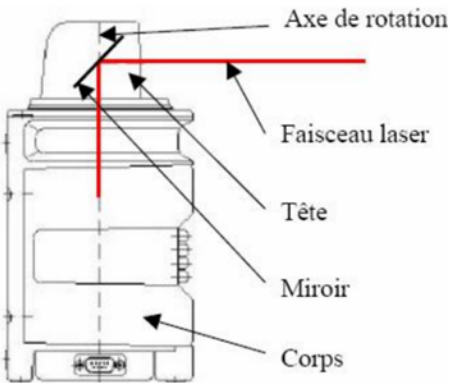


FIGURE 3.6 – Schéma descriptif du fonctionnement du LMS

Au lancement d'un faisceau, un *timer* est déclenché automatiquement et ne s'arrête de compter qu'une fois le capteur reçoit le faisceau de retour. Le temps affiché représente le temps de vol ou de voyage du faisceau et il est lié à la distance à laquelle se trouve l'obstacle qui a causé la réflexion du faisceau émis. Une simple relation permet de déterminer la distance à laquelle se trouve l'obstacle. Dans le cas d'un non-retour de l'impulsion, c'est à dire que le compteur est arrivé au 'temps mort' calculé en fonction de la portée maximale du laser et de la vitesse de propagation, aucun obstacle n'est à signaler dans la direction où le faisseau a été émis.

Une question se pose relativement à la nocivité du faisceau pour un piéton se trouvant exposé dans le champ du capteur laser. Le datasheet du capteur précise que ce denier ne provoque aucune irritation ni pour la peau ni pour les yeux : *«The laser operates at a wavelength = 905 nm (invisible infrared light). The radiation emitted in normal operation is not harmful to the eyes and human skin.»* [70].

Enfin, deux ordinateurs sont utilisés : un ordinateur embarqué qui gère la communication avec le robot bas niveau (roues, actionneurs, odomètrie) et un ordinateur portable sur lequel l'outil GenoM est embarqué.

3.2.2 Présentation de l'environnement logiciel

La plate-forme essentielle de développement et d'implémentation des programmes qui gèrent le fonctionnement du Robucar sont élaborés en langage « C ». Cet environnement destiné spécialement à la robotique mobile est connu sous le nom de « GenoM ». D'autres logiciels viennent en appui et la communication bas niveau avec le robot physique s'effectue en se servant du logiciel Syndex installé sur le PC embarqué.

Generators of Modules «GenoM»

Le générateur de modules Genom est un outil d'aide à la conception d'architectures logicielles temps-réel. Il a été conçu par le laboratoire de robotique LAAS à Toulouse et adopté par l'équipe de robotique du CDTA comme environnement de contrôle et de gestion pour le Robucar grâce aux multiples avantages qu'il offre. En effet, GenoM permet d'intégrer aisément et rapidement des fonctions opératoires dans des modules indépendants communicants chargés de leur exécution. Dans notre cas, ces fonctions (modules) peuvent être des modules de {communication, détection, évitement, localisation, navigation, planification,...}. Ces fonctions s'inscrivent dans le projet global de l'équipe de Robotique qui est «La Navigation et Contrôle d'un Robot Mobile Autonome».

GenoM s'adresse en particulier aux systèmes embarqués complexes tels que les robots mobiles autonomes ou les satellites, qui sont particulièrement exigeants sur le plan de l'informatique temps-réel distribuée car ils impliquent à la fois la cohabitation de plusieurs fonctions opératoires hétérogènes (commande de capteurs et d'actionneurs, asservissement, surveillance, traitement d'images, calcul de trajectoires...) ainsi que la parallélisation et l'embarquabilité des traitements.

Un des principaux avantages de GenoM est que ces modules peuvent être dynamiquement démarrés, interrompus ou (re)paramétrés en adressant des requêtes asynchrones paramétrées et non-bloquantes. On peut par exemple concevoir un module pour gérer une caméra (production d'images, contrôle de paramètres), un actionneur (avec différent modes d'asservissement), un GPS, un détecteur laser, ... ainsi que d'autres modules qui vont adresser des requêtes aux modules précédents afin qu'ils maintiennent des données telles que : une carte, une trajectoire ou un suivi de cible dynamique. Dans ce travail, il est question de générer un module de DATMO qui communique avec d'autres modules déjà existants comme les modules de localisation, communication et enfin le module LMS.

3.3 Approche F+D et principe du «scene undestanding» appliqués au Robucar

On entame la partie expérimentale sur le Robucar par une première expérience fondamentale. En effet, au chapitre précédent on a noté qu'en robotique mobile les informations sont mieux captées sous forme de distributions de probabilités et on a parlé aussi des notions d'incomplétude et d'incertitude. Lors de cette expérience, on montre comment à partir d'un ensemble de mesures incertaines et d'un modèle de capteur probabiliste le robot parvient finalement à une décision rationnelle.

Cette récente approche en robotique mobile est connue sous le nom d'approche F+D (système Formel + Description). Elle vient remplacer l'ancienne approche F+I (système Formel + Interprétation) qui suppose implicitement les deux postulats suivants :

- Le robot est capable, tout comme son concepteur, chacun à travers son appareil sensoriel, de percevoir une **structure ensembliste** (objets et ensemble d'objets) de l'environnement.
- Le robot et son concepteur peuvent percevoir **la même** structure ensembliste.

Beaucoup de difficultés apparaissent en F+I lorsqu'on cherche des réponses précises à des questions précises par le biais de calculs formels. En effet, les questions sont rendues précises par la modélisation mais perdent ainsi, trop souvent, toute pertinence pratique.

Il fallait donc reconsidérer la notion d'interprétation en imaginant un autre moyen de rendre effectifs les résultats des calculs. L'approche F+D est une théorie alternative des systèmes cognitifs sensoriels qui repose sur un fondement mathématique parfaitement clair : les probabilités. Néanmoins, le prix à payer est de ne fournir que des réponses approximatives aux problèmes sensori-moteurs. Cependant, même dans ce cas, ces réponses seront suffisantes pour permettre à un robot d'interagir avec son environnement.

Dans l'article *"interprétation ou description"* [78], une comparaison détaillée entre les deux approches est présentée et accompagnée par plusieurs expériences sur le robot *Khepera*. On commence nos implémentations sur le Robucar par une première expérience, "estimation des positions réelles à travers le LMS bruité", qui est à la fois une illustration du principe F+D et une justification pour notre choix de résolution du problème du DATMO.

Pour se mettre dans une situation pratique, on teste non seulement la robustesse des filtres bayésiens dans l'estimation de positions mais on s'intéresse aussi au temps de convergence de ces filtres. En effet, les auteurs dans le *«Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety»* [40], expliquent que les manœuvres automatiques présentent un réel avantage dans l'évitement d'obstacles et de collisions. L'objectif est que la voiture autonome puisse réagir au moins aussi vite que l'être humain. La capacité d'interpréter la scène réelle (en anglais *« scene undestanding »*), rapidement pour pouvoir dégager la bonne décision s'évalue selon plusieurs critères comme le *«time to collision, TTC»* aussi bien en navigation autonome qu'en navigation assistée.

De façon générale, on compare la voiture autonome à un conducteur humain : on estime que l'être humain interprète la scène qui l'entoure en 0.1s et il peut engager une action (réflexe) dans un temps compris entre 0.1s à 1s. Le but de cette expérience est de voir si, à partir des mesures bruitées du LMS-511, le Robucar peut positionner exactement tous les points qu'il détecte à leurs véritables positions en un temps qui ne dépasse pas 1s.

On rappelle que le LMS-511 est réglé à 75Hz c'est-à-dire que chaque 13.33 ms il renvoie un scan (75 scans en 1 s). On se donne une position exacte d'un point situé à 3m et dans une direction de 30 $^\circ$.

Scénario 1 : *" Le robot scrute la scène de la figure et défile les données numériques correspondant aux mesures acquises par le module LMS. On considère un point de l'obstacle (carton) qui se situe* ***exactement*** *à une distance de 3m selon l'angle 30* $^\circ$ *."*

FIGURE 3.7 – Estimation d'une position réelle par filtrage bayésien à travers le LMS-511

La position (x, y) qu'on désire estimer est donnée par :

$$S_{reel} = (x_{reel} = 300cos(30) = 259.807\,cm \;\; ; \;\; y_{reel} = 300sin(30) = 150.00\,cm). \tag{3.1}$$

Les mesures successives $Z_1, ..., Z_n$ fournies par le LMS avoisinent le point S_{reel}. A un instant t donné, on désigne par

$$P(S_t|Z_t), \tag{3.2}$$

la distribution probabiliste qu'on appelle *posterior*, où S_t est la position (x_t, y_t). On exprime la distribution précédente sous une forme compacte qui est donnée par le théorème de Bayes :

$$P(S|Z) = \frac{P(S)P(Z|S)}{P(Z)} \tag{3.3}$$

Le principe de filtrage bayésien, illustré par la figure 3.8, est de rendre l'expression (3.3) récursive. En d'autres termes, pour chaque nouvelle mesure Z_i, l'ancienne estimation $P(S_{i-1}|Z_{i-1})$ devient le nouveau prior $P(S_{i-1})$ et le robot estime le nouveau *posterior* $P(S_i|Z_i)$ en tenant compte des expériences passées. C'est par ce cumul qu'il aboutit très vite à la vraie information. Le $P(Z_t)$ est difficile à déterminer. On utilise les régles de distributions marginales pour implémenter ce terme. Quant à la vraissemblance $P(Z_t|S_t)$ elle représente la manière dont les mesures sont acquises, dans notre cas par LMS. Pour pouvoir implémenter cette dernière il faut d'abord déterminer le modèle probabiliste du LMS-511.

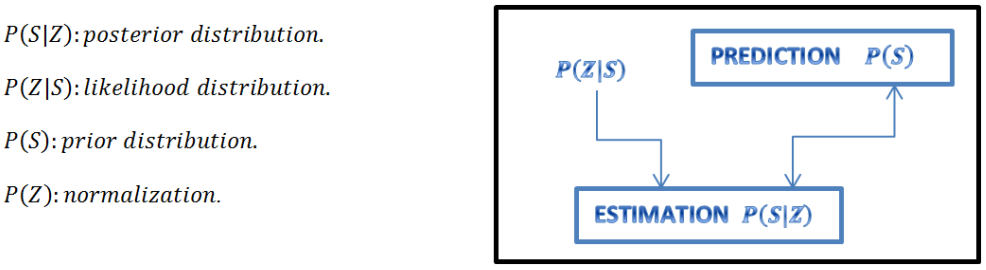


FIGURE 3.8 – Cycle d'estimation prédiction dans le filtrage bayésien.

3.3.1 Modèle probabiliste du capteur laser LMS

On rappelle tout d'abord l'équation fondamentale du DATMO :

$$bel_t \quad = \quad \mu P(Z_t|S_t) \int P(S_t|S_{t-1}) bel_{t-1} dS_{t-1}, \tag{3.4}$$

La formule ci-dessus représente l'équation récursive bayésienne qui se traduit par la suite en un filtre bayésien. Cette expression fait intervenir le modèle de mesures (*measurement model*) noté $P(Z_t|S_t)$.

On a défini le modèle de mesures comme étant une distribution probabiliste bayésienne qui exprime le mécanisme de mesure d'une manière statistique, c'est-à-dire la probabilité d'avoir un ensemble de mesures données sachant qu'on a la vraie valeur.

Dans notre cas, il s'agit de déterminer l'ensemble de mesures Z_t en sachant qu'elle est la position exacte X_t d'un objet présent en scène. Au chapitre précédent, on a donné un exemple du modèle probabiliste d'un télémètre laser et on a vu que pour les capteurs généralement utilisés ce sont les facteurs externes qui influent le plus sur le fonctionnement de ces derniers. Dans ce paragraphe on essaie de décrire le fonctionnement du capteur LMS dans un environnement quelconque. En toute rigueur, on va suivre les étapes de l'élaboration d'un programme bayésien comme il a été illustré au chapitre précédent.

❑ Phase de spécification :

Pour modéliser le fonctionnement du LMS 511 du Robucar (figure 3.4), on choisit les trois variables pertinentes suivantes :

- V_q : « Variable qualité », elle donne les différentes conditions externes au capteur. On peut discrétiser cette variable de 1 à 5 avec un pas de 1. On suppose que 1 est le cas le plus favorable au fonctionnement du laser et 5 le cas le plus défavorable.
- θ_0 : L'angle auquel se trouve un obstacle. A partir des caractéristiques du LMS511 données précédemment, on déduit que :
  $\theta_0 \in [1^\circ, 180^\circ]$, avec un pas de 1°. Les caractéristiques du LMS fournies en début de chapitre indiquent qu'il est possible de choisir d'autres résolutions, inférieures à 1°, de l'ordre de 0.25°. On note n_θ le nombre fini de valeurs que peut prendre θ_0.
- ρ_0 : La distance à laquelle se trouve l'obstacle suivant une direction donnée. On se limite à un rayon de 600 cm dans la suite de ce mémoire. De la même manière, on note n_ρ le nombre de valeurs que peut prendre ρ_0 dans cet intervalle.

❑ Connaissances préalables de dépendance :

Suite au choix des variables pertinentes, la description du modèle porte sur ces variables et s'exprime par leur probabilité conjointe. Cette étape permet d'exprimer nos connaissances sur les phénomènes mis en jeu et ainsi faciliter la réponse à la question posée. On s'intéresse à l'expression suivante :

$$P(V_q\ \theta_0\ \rho_0 | Z\ C), \tag{3.5}$$

avec C : connaissances préalables et Z : ensemble de nouvelles mesures.

Nous décomposons cette probabilité conjointe sous la forme du produit de trois distributions élémentaires :

$$P(V_q\ \theta_0\ \rho_0 | Z\ C) = P(V_q | Z\ C) P(\rho_0 | V_q\ Z\ C) P(\theta_0 | V_q\ \rho_0\ Z\ C). \tag{3.6}$$

D'une part, nous considérons que la connaissance de la direction n'apporte aucune information sur la distance à laquelle se trouve l'obstacle. Nous traduisons cet a priori en considérant que la probabilité de ρ_0 est indépendante de θ_0. D'autre part, on peut imaginer qu'une petite erreur ϵ sur l'angle va s'amplifier pour des grandes distances ρ_0. Les distributions de probabilités dans ce cas ne sont plus indépendantes. Enfin, le capteur laser est l'unique capteur utilisé par le Robucar et il ne fournit aucune information susceptible de donner les conditions et la qualité du milieu externe (contrairement à un capteur de lumière, humidité...).

On simplifie la décomposition précédente en gardant la dépendance entre l'angle et le rayon,

$$P(V_q\ \theta_0\ \rho_0|Z\ C) = P(V_q|C)P(\rho_0|V_q\ Z\ C)P(\theta_0|V_q\ \rho_0\ Z\ C). \tag{3.7}$$

Cette décomposition est très intéressante dans la mesure où elle permet d'exprimer d'une part nos « a priori » sur la distance et la direction de l'obstacle par rapport au robot et, d'autre part, une distribution représentant la valeur V_q en fonction des connaissances préalables (avis expert).

Le LMS SICK est connu pour sa grande précision. Ses performances ne sont influencées que par des facteurs externes comme la pluie, le brouillard, la fumée ou encore la nature des objets à détecter. De plus, les utilisateurs du LMS au CDTA ont remarqué que les performances dans les environnements *"out-door"* sont moins bonnes que les performances en *"in-door"*. On choisit l'échelle de 1 à 5 pour représenter ces diverses conditions d'emploi.

En absence de capteurs pouvant donner des informations caractérisant l'environnement externe, on modélise la probabilité $P(V_q|C)$ par une distribution uniforme U. En d'autres termes, on suppose qu'à chaque instant le Robucar peut rencontrer une situation favorable au fonctionnent du LMS ($Vq = 1$) avec une probabilité de $\frac{1}{5}$ comme il peut rencontrer une situation moins favorable ($Vq = 2, \ldots, 5$) avec une probabilité de $\frac{1}{5}$ pour chaque valeur de V_q. Enfin, on exprime habituellement les connaissances préalables sur θ_0 et ρ_0 par :

$$P([\theta_0 = \theta_{0i}]|C) = \tfrac{1}{n_\theta}, \tag{3.8}$$

$$P([\rho_0 = \rho_{0i}]|C) = \tfrac{1}{n_\rho}, \tag{3.9}$$

pour indiquer que nous ne privilégions pas certaines positions des obstacles vis-à-vis des autres.

❑ Identification des paramètres :

Dans la littérature, les modèles donnés aux distributions de probabilités recherchées sont le plus souvent des lois normales (voir *programmation bayésienne des robots,* [32]). On souhaite justifier ce choix dans le cas du LMS-511 avant de l'admettre comme modèle. Pour cela on va prendre des données réelles acquises par le LMS durant une de nos expériences. Le point dont on mesure la distance est évidemment statique durant l'enregistrement. En réalité on n'a pas de moyens expérimentaux pour déterminer l'imprécision sur l'angle qui accompagne ces mesures mais, malgré tout, cela reste un moyen fiable pour déterminer les caractéristiques statiques du LMS.
La connaissance préalable de l'environnement expérimental permet d'affirmer que les conditions sont favorables ($V_q = 1$). On traduit les mesures obtenues par l'histogramme suivant :

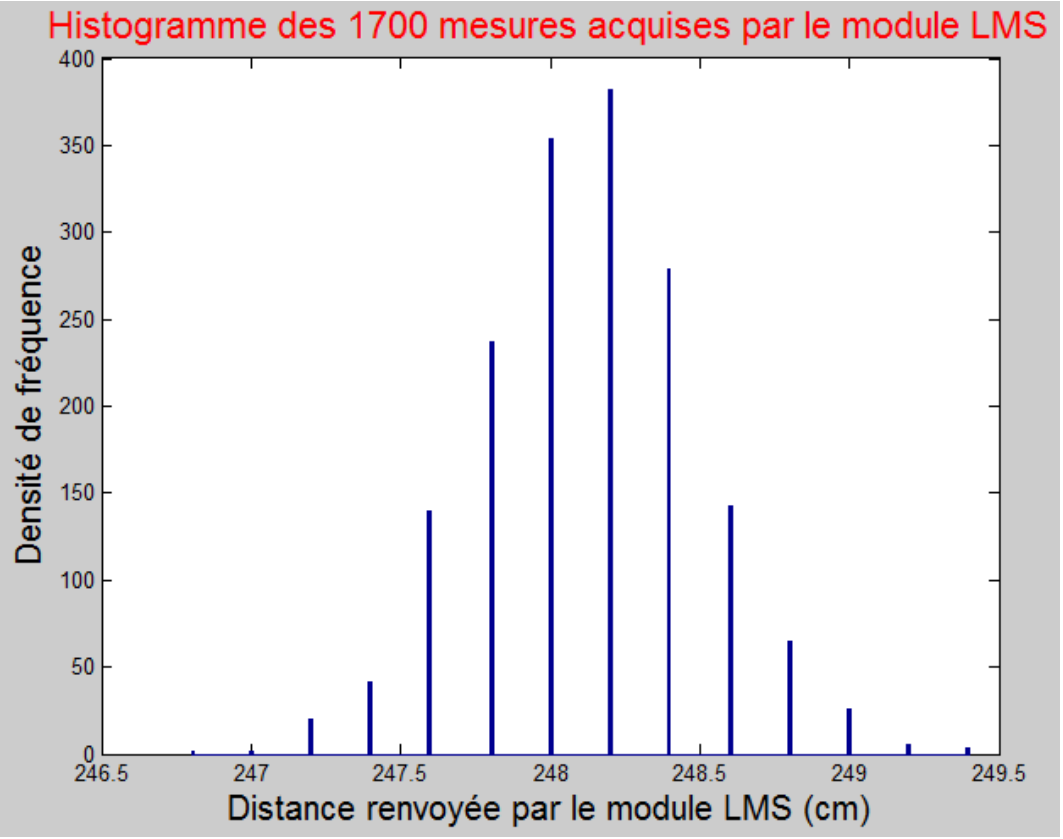


FIGURE 3.9 – Histogramme des mesures acquises pour un point statique

On remarque bien que l'histogramme suit l'allure d'une loi normale. La moyenne est naturellement la distance réelle à laquelle se trouve l'objet et elle est estimée à $\rho^* = 248.13cm$ (pour cet histogramme seulement). Quant à l'écart-type, il est donné par $\sigma = 0.3655$ pour n'importe quelle mesure.

En partant de ces données expérimentales, on suppose que les mesures du LMS selon une direction ρ donnée, suivent de manière sous-jaçante une loi normale $N(\rho^*, \sigma^2)$. Pour valider ce modèle, on génère automatiquement sous Matlab un fichier de valeurs de même taille que le fichier réel. Ces valeurs sont normalement distribuées autour de la moyenne ρ^*. La superposition des deux histogrammes nous permet de valider le modèle. En efft, la correspondance est confirmée sur la figure 3.10. En prenant un nombre de données plus important, les deux histogrammes s'identifient parfaitement.

Contrairement à la mesure de distance, on n'a pas de moyen expérimental pour estimer l'imprécision sur l'angle. On ne connait qu'une borne supérieur (0.25°) qui est la résolution de LMS donnée par le constructeur. On choisit arbitrairement une valeur qui est de l'ordre de la variance déterminée pour ρ_0 et on considère que cette erreur croit par palier avec la distance ρ_0 et linéairement avec les conditions externes V_q. Par exemple, on peut considérer que pour une distance donnée ρ_0, les valeurs des variances de ρ_0 et θ_0 croient selon la relation,

$$Var_{\theta_0,\,\rho_0}(V_{q(i+1)}) \quad = \quad 2Var_{\theta_0,\,\rho_0}(V_{q(i)}) \tag{3.10}$$

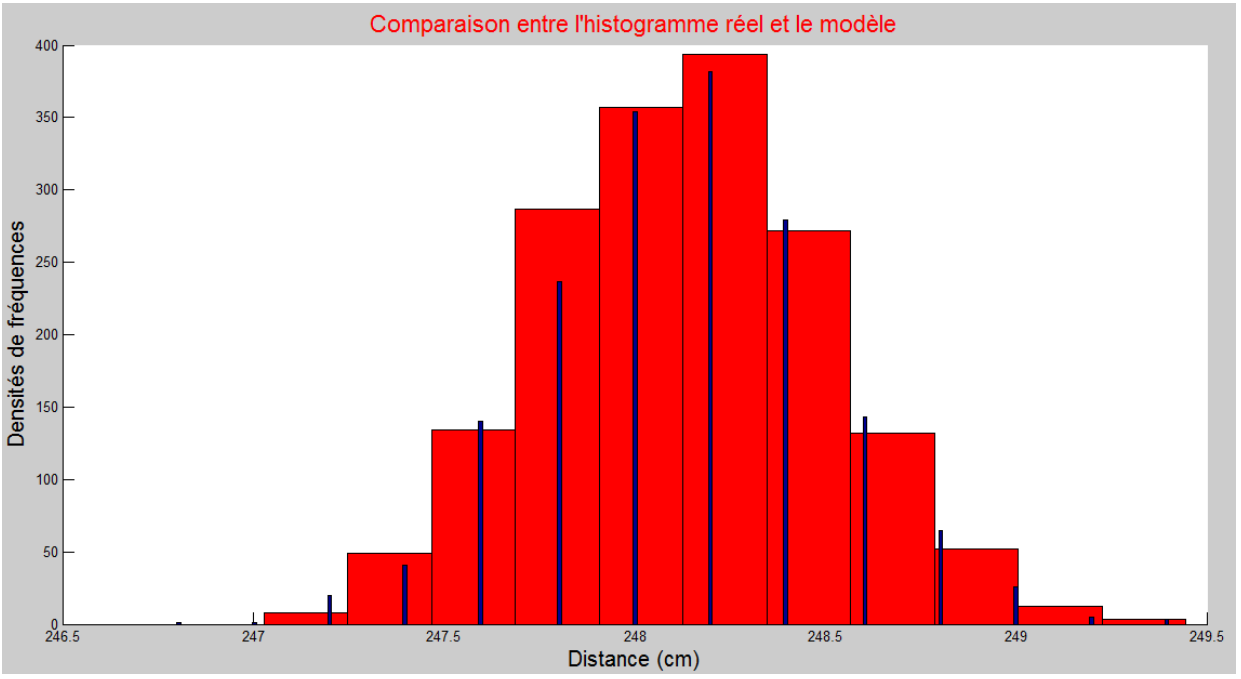


FIGURE 3.10 – Validation du modèle du LMS par comparaison

qui couvre largement toutes les imprécisions possibles du LMS. Suite aux étapes de description et d'identification, survient l'étape de formulation des questions qui sont en fait les distributions qu'on désire déterminer. On rappelle que le *"bayesian program"* décrit au chapitre précédent sert à construire des modèles probabilistes et résoudre les problèmes d'inférence qui y sont liés. Dans le cas du capteur LMS on désire répondre à la question suivante :

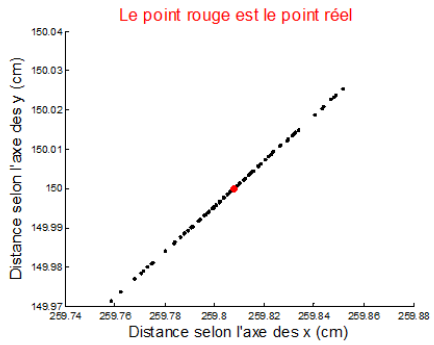
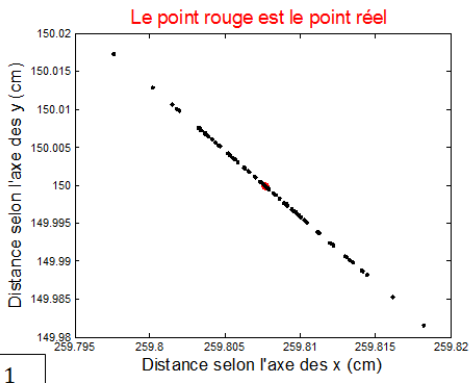
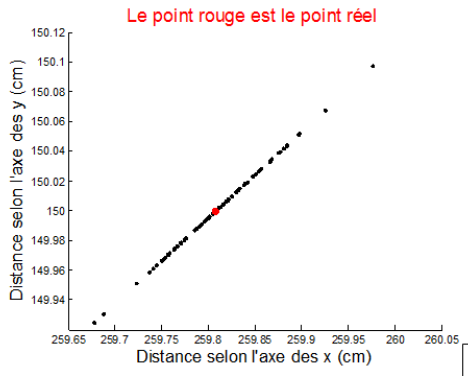
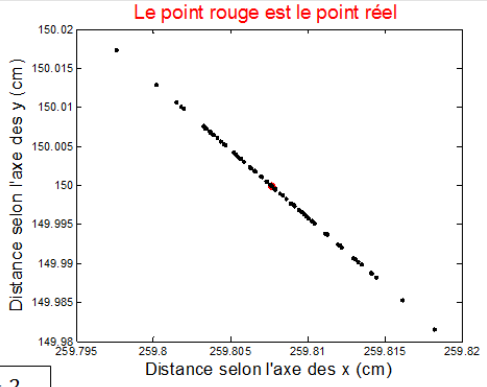
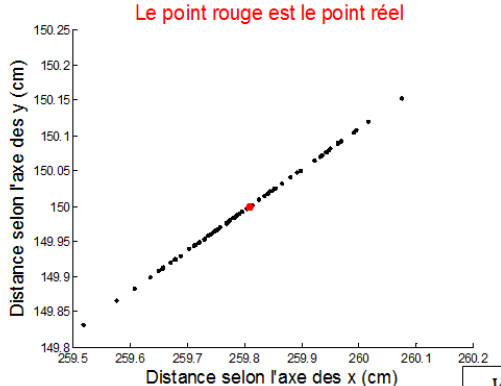
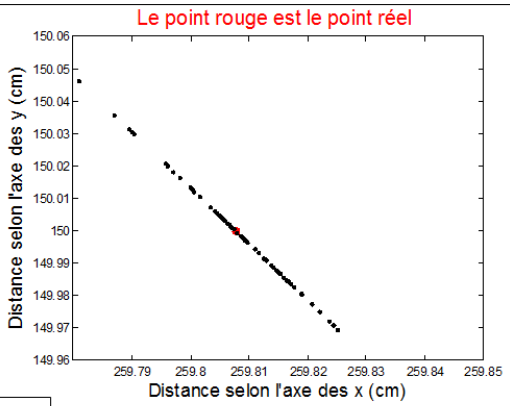
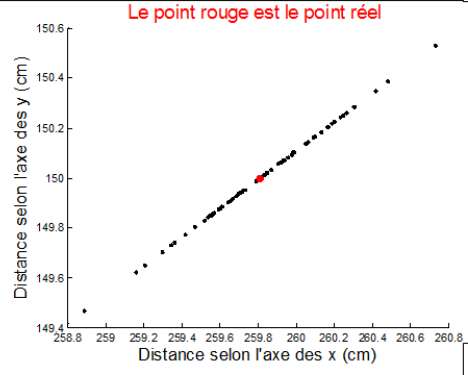
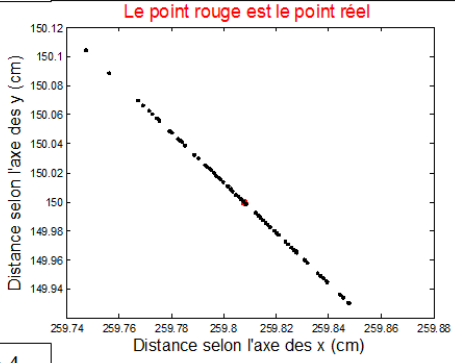
$$P(Z|C\ V_q\ \theta_0\ \rho_0), \tag{3.11}$$

pour qu'en suite on puisse déterminer efficacement les *posterior* :

$$P(\theta_0|Z\ C\ V_q\ \rho_0), \tag{3.12}$$

$$P(\rho_0|Z\ C\ V_q). \tag{3.13}$$

En partant de la description ci-dessus, on arrive à découpler la distribution (3.11) et déduire assez facilement les deux fonctions de vraissemblance qui nous intéressent, à savoir : $P(Z_\rho|Vq\ C\rho_0)$ et $P(Z_\theta|Vq\ C\theta_0\ \rho_0)$.

Les mesures du rayon	*Les mesures de l'angle*
	
$V_q = 1$	
	
$V_q = 2$	
	
$V_q = 3$	
	
$V_q = 4$	

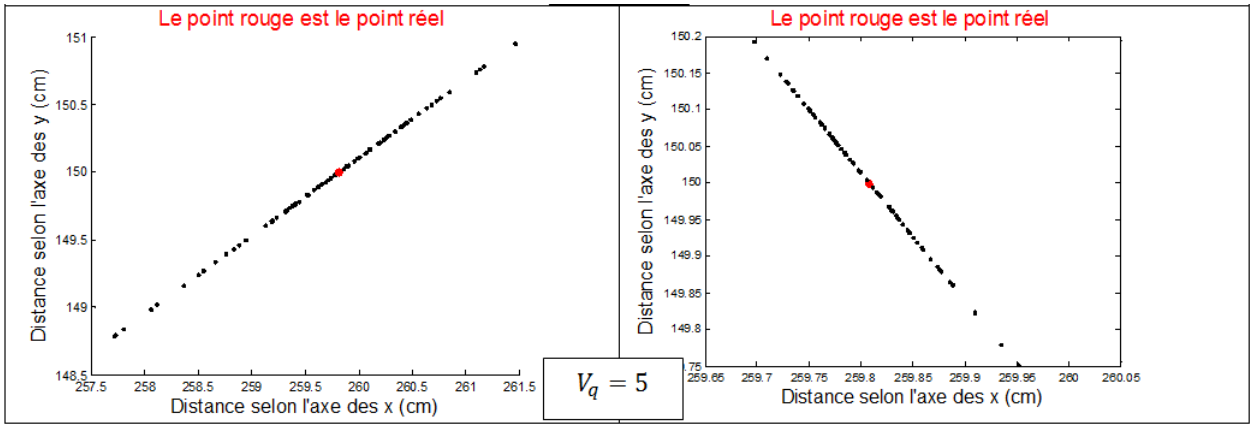


FIGURE 3.11 – Simulations des données renvoyées par le LMS pour différents V_q

La distribution $P(\theta_0|Z\ C\ V_q\ \rho_0)$ est dépendante de $P(\rho_0|Z\ C\ V_q)$ selon une relation de correspondance établie par paliers. On peut déduire le modèle probabiliste du LMS-511 assez facilement pour un V_q donné. La figure 3.12 donne un exemple d'histogrammes pour ces deux distributions pour $V_q = 1$ et 1700 mesures LMS.

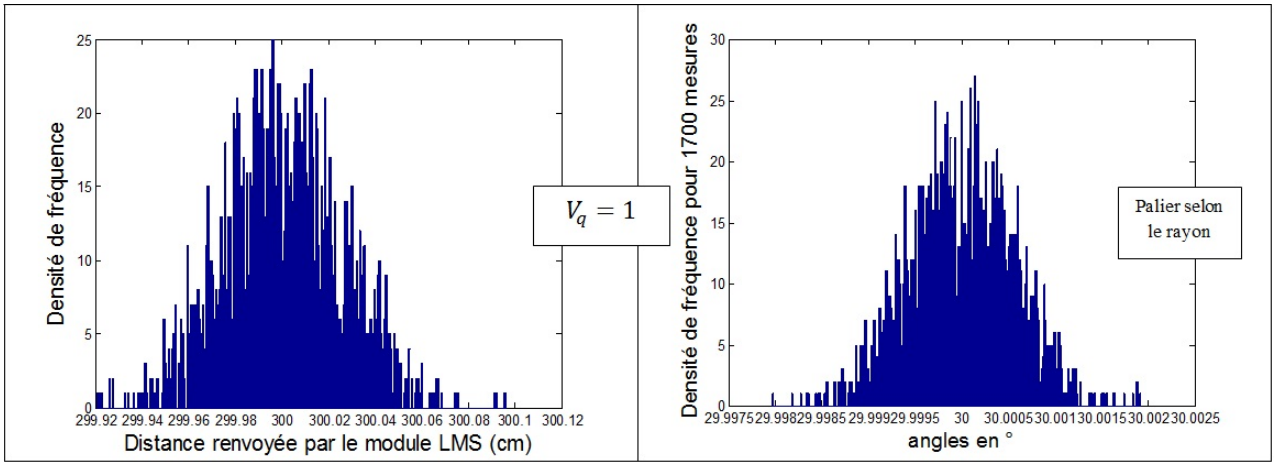


FIGURE 3.12 – Histogrammes des 1700 mesures LMS selon θ_0 et ρ_0 pour $V_q = 1$

3.3.2 Estimation de la position par application du filtre bayésien

Dans cette partie, on voit comment le Robucar va estimer la vraie position du point en exploitant le modèle du capteur précédemment élaboré. La valeur de la position mesurée selon un angle donné lors du premier scan va déterminer le palier de l'erreur selon l'angle. Ensuite, en 0.33 s ou en 1 s, le filtre bayésien va permettre d'estimer la bonne position de tous les points se trouvant devant le robot. On rappelle que les écarts-types déterminés lors la modélisation servent à implémenter les fonctions de vraissemblance dans le filtrage bayésien.

La première mesure de distance détermine le modèle à donner à l'estimation selon θ_0. Ainsi un modèle $P(\theta_0|V_q Z C \rho_0)$ est sélectionné dans le programme MATLAB. Sur la base des modèles selon θ_0 et ρ_0 on déduit l'ensemble des mesures réelles LMS auxquelles on s'attend. L'estimation dans le programme Matlab se fait selon θ et ρ et on en déduit le *posterior* selon x et y à chaque itération.

Sur la figure ci-dessous, à gauche, le point rouge représente le point réel et les points noirs qui l'entourent sont les différentes valeurs retournées par le LMS-511. A droite, les traits en bleu représentent les coordonnées réelles qu'on a estimées. L'implémentation de ce filtre bayésien a donné les bons résultats pour différents tests. Ainsi, le Robucar est capable d'aboutir à des estimations réalistes à travers le modèle probabiliste de son capteur et le filtrage bayésien.

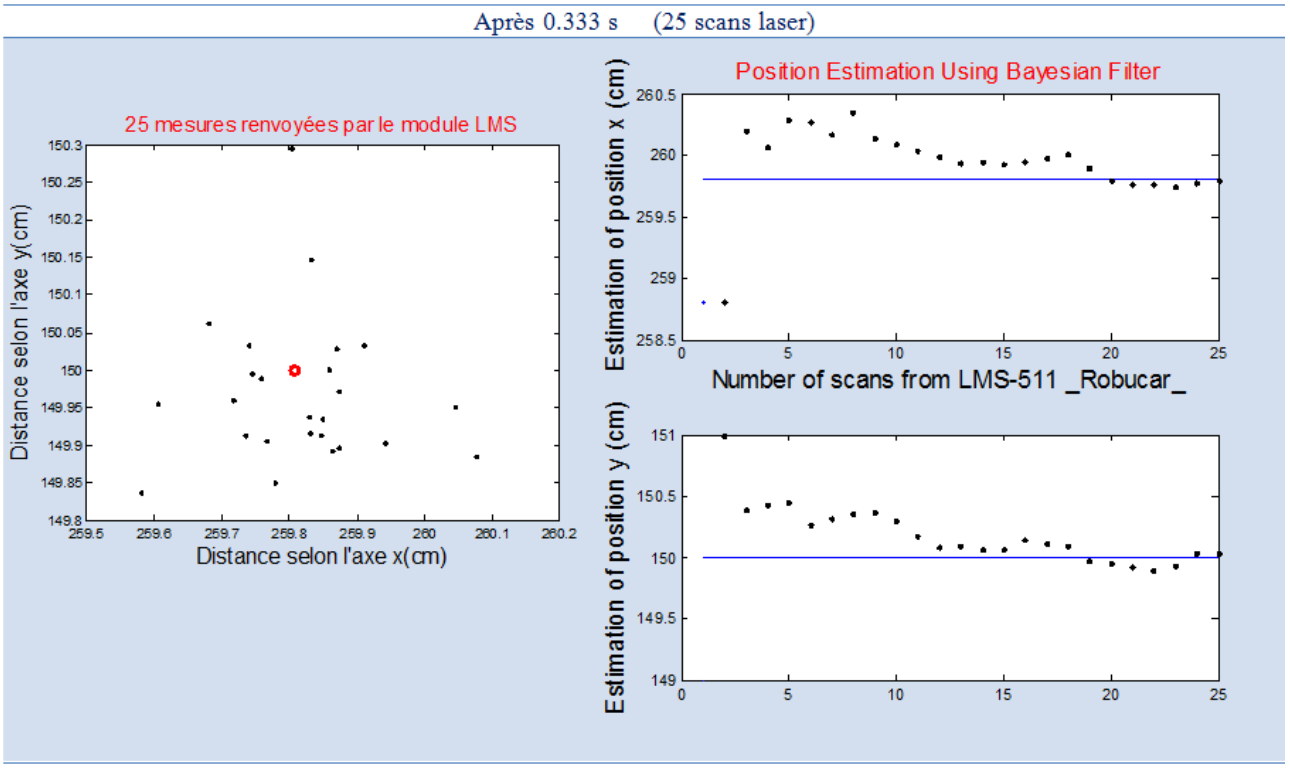


FIGURE 3.13 – Estimation de la position réelle en 0.33 s.

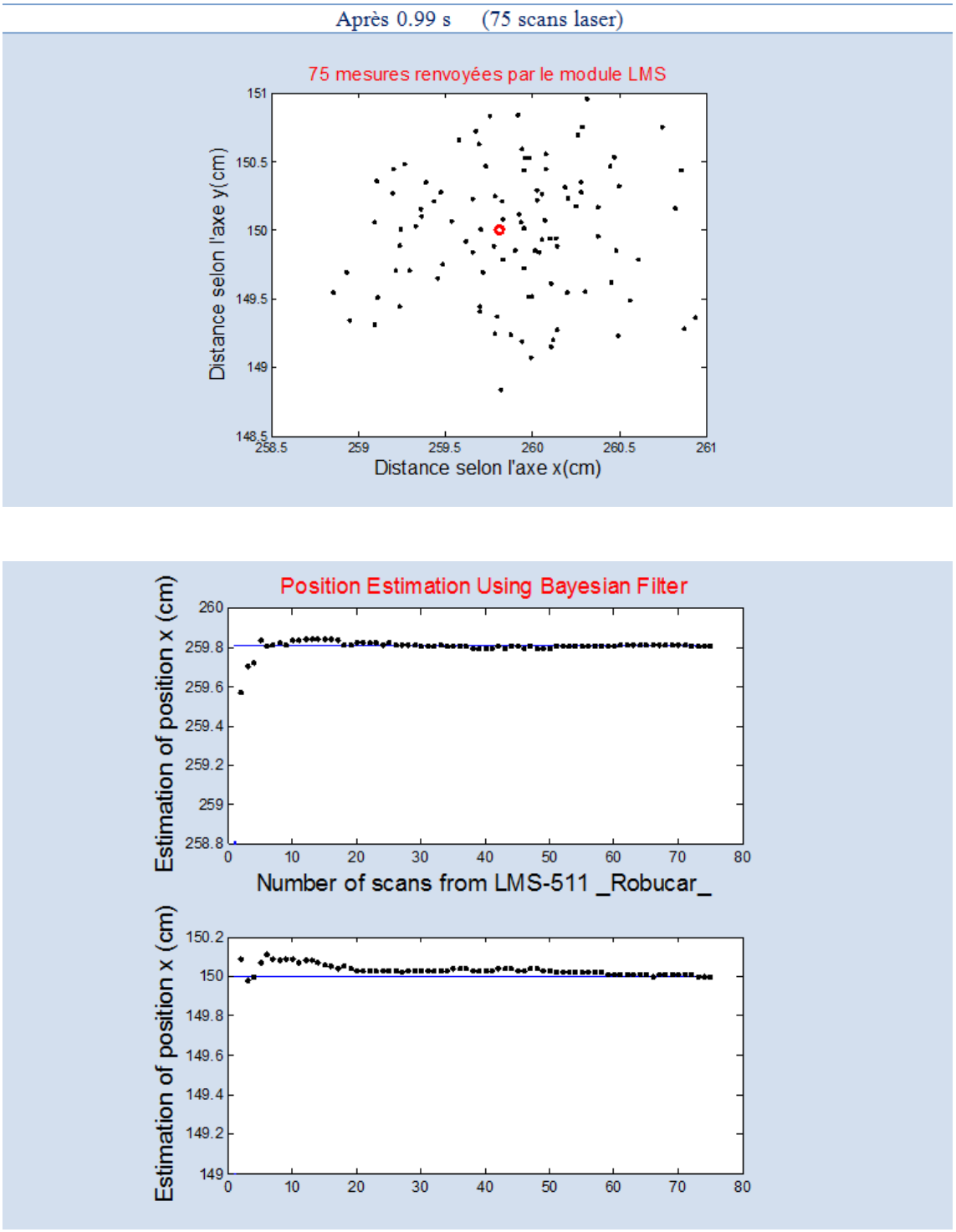


FIGURE 3.14 – Estimation de la position réelle en 1 s.

Commentaire : Le but de cette première expérience est de montrer comment un robot parvient à des décisions rationnelles en partant de mesures bruitées et d'un modèle probabiliste du capteur utilisé. La démarche est la même pour d'autres mesures où les imprécisions sont plus marquées comme les mesures GPS par exemple. Un principe similaire est utilisé

en localisation du robot [7] et dans d'autres domaines relatifs à la robotique mobile. Le programme bayésien permet, comme on l'a illustré pour le cas du LMS, de simplifier la démarche de construction d'un modèle probabiliste et d'organiser la description du phénomène à modéliser. Le choix de la décomposition de la distribution conjointe est l'étape la plus importante. Mis à part son utilisation pour estimer les distances réelles vis-à-vis des obtacles statiques, le modèle probabiliste du LMS-511 se trouve très utile dans la fusion de données entre plusieurs capteurs.

D'autre part, ce premier exemple justifie le choix de la programmation bayésienne pour le Robucar. Par la suite, on va voir l'implémentation du BOF qui est une adaptation particulière de la logique des filtres bayésiens au contexte du DATMO. Mais avant on enchaine sur le SLAM et plus particulièrement sur la technique d'ICP-SLAM qui est utilisée par le Robucar pour se localiser et pour cartographier des lieux statiques. L'expérience qu'on va réaliser en SLAM a été la première pour le Robucar dans un environnement contenant des objets dynamiques. On rappelle que les modules de SLAM et DATMO sont indissociables pour les véhicules autonomes. A la fin de cette expérience on tire un certain nombre de conclusions qui vont guider notre approche en DATMO.

3.4 Application de la technique ICP-SLAM

Dans cette partie nous allons d'abord mettre en oeuvre la technique ICP comme un moyen de localisation en robotique avant de passer à la technique d'ICP-SLAM utilisée pour cartographier un environnement en mettant en évidence ses limites dans des environnements dynamiques.

3.4.1 Description de la technique ICP

L'ICP «*Iterative Closest Points*» est une approche du *scan matching* permettant de réaliser le recalage automatique entre deux modèles de données géométriques quelles que soient leurs natures (points, droite, surfaces). Elle est basée sur la recherche du point le plus proche pour calculer, de façon itérative, la matrice de transformation permettant d'effectuer l'alignement.

Le principe de l'algorithme ICP est d'itérer les étapes d'alignement qui sont : la mise en correspondance des données et l'estimation de la transformation de repères entre les données à recaler. Au bout de chaque itération, l'algorithme fournit une liste de points associés et une estimation de la transformation de repère entre les données. Cette transformation est utilisée à l'itération suivante pour la mise à jour de la liste de points associés. Ces derniers serviront, à leur tour, pour calculer une nouvelle estimation de la transformation. Ces étapes sont répétées jusqu'à la convergence de l'algorithme. La technique ICP n'impose pas de conditions préalables et sa convergence est démontrée pour la majorité des cas.

La figure ci-dessous est un exemple permettant de décrire l'algorithme ICP : Supposons que le robot mobile est dans une position de référence notée P_{ref}, il prend un premier scan noté S_{ref}. Par la suite, le robot se déplace vers une nouvelle position P_{new} par rapport au repère de référence, il prend un nouveau scan S_{new}. La différence approximative de la position

P_{new} à partir de la position P_{ref} (translation et rotation relatives) est connue, dans notre cas, par odométrie. L'algorithme ICP permet de déterminer la vraie position P_{new} en alignant les deux scans (S_{ref} et S_{new}).

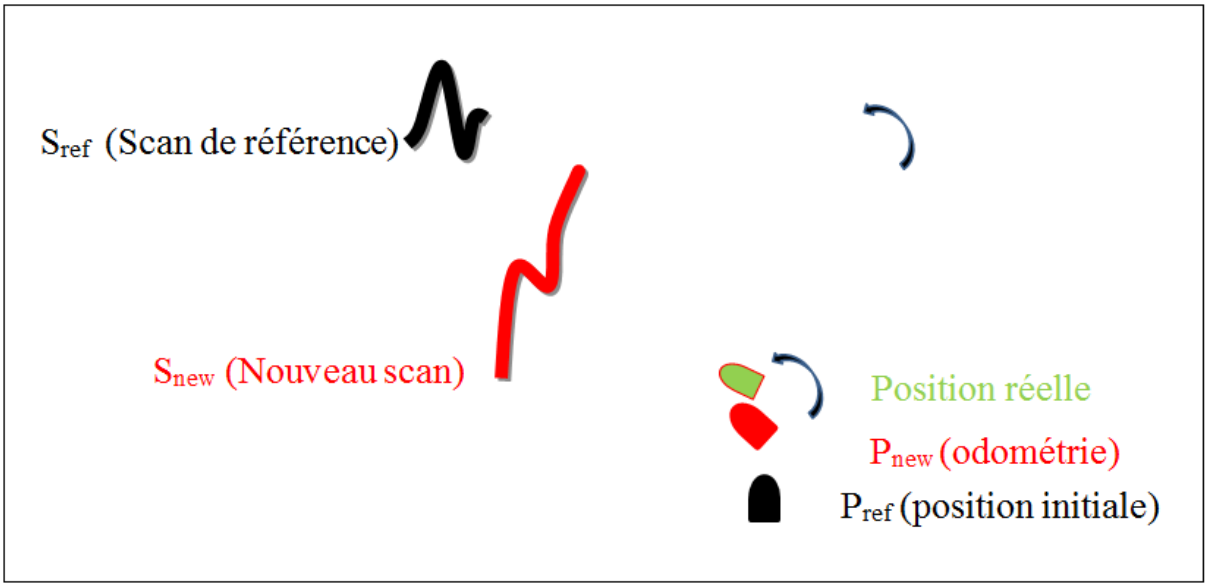


FIGURE 3.15 – Description de la technique ICP

L'algorithme ICP se compose de trois parties principales :

- Projection des données du laser :

  Un scan laser noté S est défini dans le repère local du robot. En revanche, l'odométrie du robot donne la position P de ce dernier dans le repère global. On déduit que chaque nouveau scan S_{new} doit être projeté dans le repère global et on le note $\hat{S}_{new}$. On peut effectuer cette projection par la matrice de transformation homogène entre les deux repères :

$$T = \begin{pmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

- Association de données ou *matching* :

  Dans cette étape, on fait correspondre à chaque point du scan $\hat{S}_{new}$ un point du scan de référence S_{ref}. Le principe est de calculer toutes les distances euclidiennes entre un point de $\hat{S}_{new}$ et tous les points de S_{ref} puis choisir la plus petite distance. La figure 3.16 illustre cette démarche.

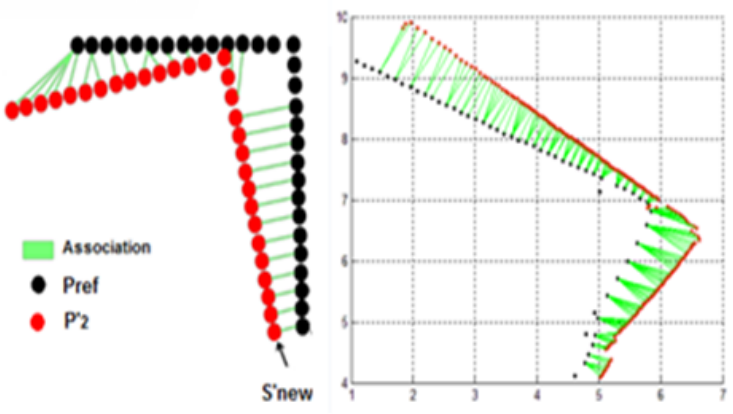


FIGURE 3.16 – Illustration du principe de *scan matching*

A ce stade, on peut voir pourquoi les techniques de *data-association* posent beaucoup de problèmes dans un environnement dynamique complexe. En efft, lorsque les objets ne sont pas statiques, faire correspondre les points entre deux scans rien qu'en se basant sur la distance euclidienne induit le plus souvent à de mauvaises interprétations.

```
for  i=1 à N
  for  k=1 à N0
    dx = X'new(i) - Xref(k)
    dy = Y'new(i) - Yref(k)
    d(k) = sqrt(dx^2 + dy^2)
    if (dmin(i) >d(k))   dmin(i)=d(k)
  end for
    c(i) = k
end for
```

FIGURE 3.17 – pseudo-programme pour le *data association*

avec

- → N : taille de $\hat{S}_{new}$,
- → N_0 : taille de S_{ref},
- → $\hat{X}_{new}$ abscisse du point de $\hat{S}_{new}$,
- → X_{ref} abscisse du point de S_{ref},
- → $\hat{Y}_{new}$ ordonnée du point de $\hat{S}_{new}$,
- → Y_{ref} ordonnée du point de S_{new},
- → d_{min} distance minimale entre un point de $\hat{S}_{new}$ et S_{ref},
- → $c(i)$ rang du point de S_{ref} correspondant au i^{eme} point de $\hat{S}_{new}$.

❍ Estimation de la position :

La troisième étape dans l'algorithme de l'ICP est l'étape d'optimisation d'un critère d'alignement. En effet, on désire minimiser au sens des moindres carrés l'erreur sur la distance entre les points associés :

$$J = \frac{1}{N}\Sigma_{i=1}^{N}\left\|S_{ref}(c(i)) - \hat{S}_{new}(i)\right\|^2 \tag{3.15}$$

On rappelle que $\hat{S}_{new} = TS_{new}$. On déduit que ce critère fait intervenir la matrice T donnée par (3.14). L'optimalité est donnée en dérivant l'équation (3.15) et les résultats sont les paramètres θ, t_x et t_y de la matrice T. Le détail du calcul est donné dans l'article [62]. Ainsi, l'algorithme de l'ICP sert à localiser le robot en estimant à chaque itération la matrice de transformation homogène qui traduit le mouvement effectué. Dans ce cas, l'odométrie ne sert qu'à initialiser l'algorithme de l'ICP. Ce dernier s'applique directement en SLAM, ou ce qu'on appelle la technique **ICP-SLAM**.

Le principe du ICP-SLAM est d'exploiter l'algorithme de l'ICP non seulement en localisation, mais aussi pour construire une carte de l'environnement perçu par le robot.
Les étapes sont décrites dans [62] comme suit :

→ La configuration du robot est initialisée à la position $P_0 = (x = 0, y = 0, \theta = 0)$,

→ Un premier scan S_0 est effectué et projeté sur la carte,

→ Le robot se déplace à la nouvelle position P_{new} et effectue un nouveau scan laser S_{new},

→ Le déplacement est implémenté par l'algorithme ICP,

→ A l'aide de la matrice de transformation homogène T ainsi obtenue, on ramène S_{new} à la position initiale,

→ Les points de S_{new} associés à des points précédemment détéctés sont fusionnés comme indiqué sur la figure 3.18. Les autres points sont projetés sur la carte globale qui grandit à chaque déplacement du robot,

→ On retourne à la deuxième étape et on itère jusqu'à ce que le robot s'arrête.

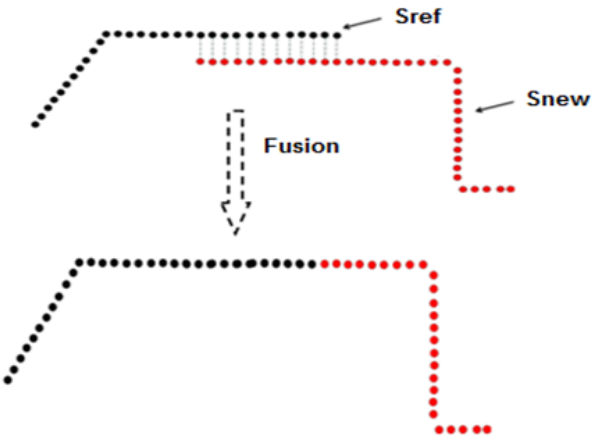


FIGURE 3.18 – Fusion de données dans la carte

Remarque :

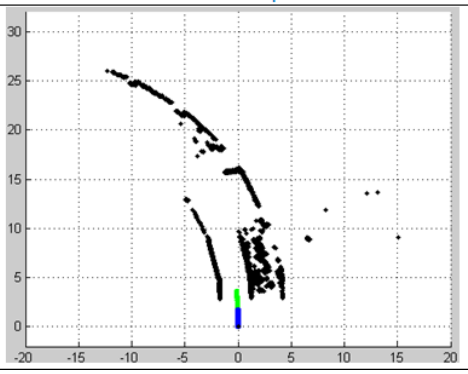
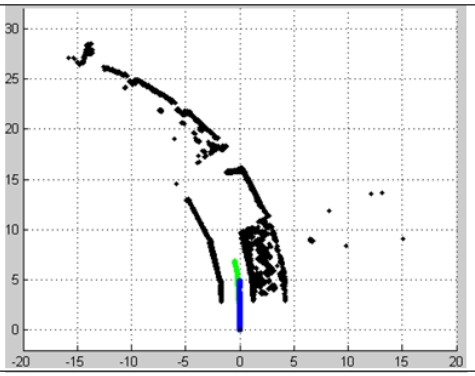
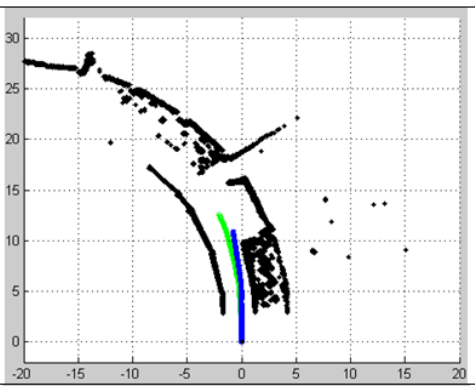
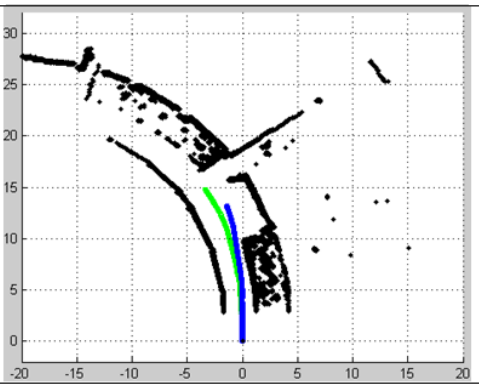
En examinant les étapes de traitement à la fois de l'algorithme d'ICP et d'ICP-SLAM on remarque qu'il n'est apparemment pas destiné aux environnements dynamiques. En effet, comme il a déjà été mentionné, le ICP se base sur une technique de scan maching pour extraire la transformation homogène T en supposant que l'environnement autre que le robot est statique. Quant à l'ICP-SLAM il projette les points sur une carte globale à l'itération i mais il ne prévoit aucune réctification de ces points à l'instant $t+1$. Ainsi, si un point correspond à un objet dynamique à l'instant t, il va réapparaitre à la carte à l'instant $t+1$.

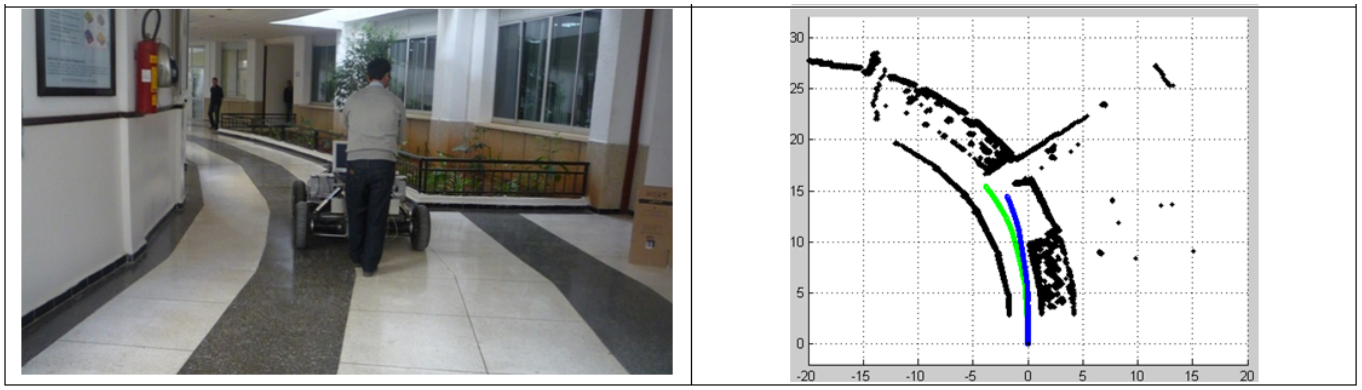
3.4.2 Exploitation du module ICP-SLAM et mise en évidence de ses limites

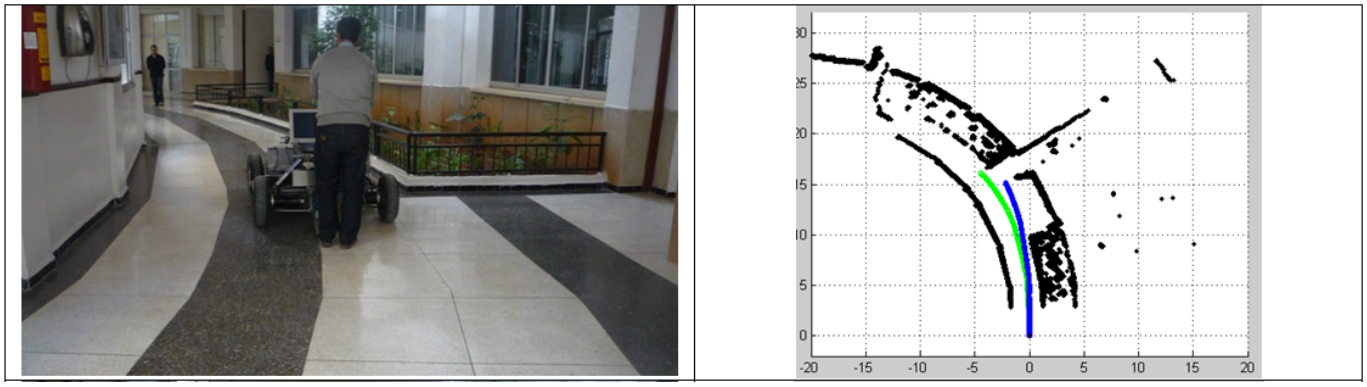
Le module ICP-SLAM est déjà implémenté sur l'environnement logiciel du Robucar. Nous allons l'exploiter pour la première fois dans un environnement dynamique et discuter de ses résultats. L'expérience effectuée aux locaux du CDTA est décrite ci-dessous :

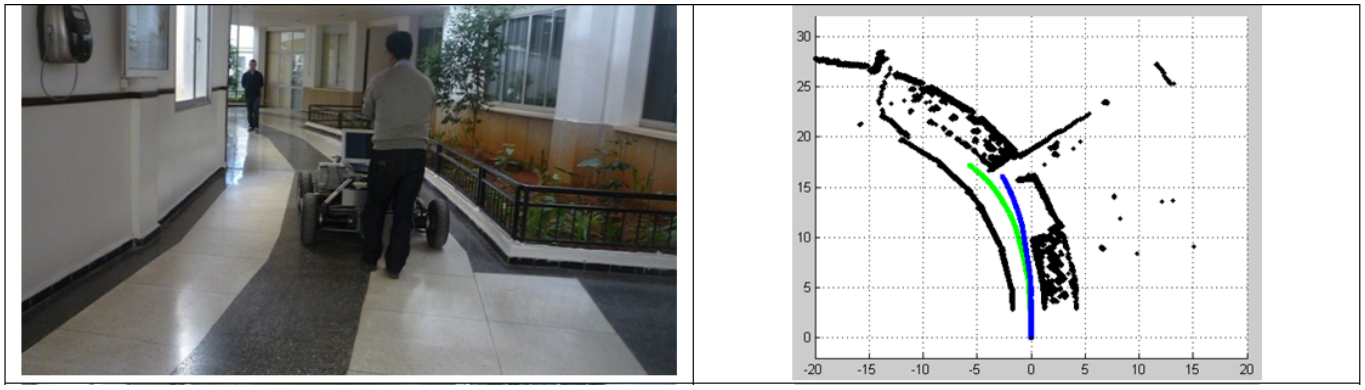
Scénario 2

«Le Robucar va suivre un certain parcours dans les couloirs du CDTA. En se déplaçant, il retrace par la technique ICP-SLAM l'environnement qui l'entoure. Un piéton va se déplacer dans le champ de vision du LMS et se diriger vers le Robucar. Ce scénario est illustré par la succession des figures ci-dessous. Elles donnent d'une part les images réelles illustrant l'évolution de l'expérience, et d'autre part la carte retracée au cours du temps»

Images réelles	Reconstituion ICP-SLAM En vert : localisation par ICP En Bleu : Localisation par odomètrie
	
	
	
	


30
25
20
15
10
5
0
-20
-15
-10
-5
0
5
10
15
20


30
25
20
15
10
5
0
-20
-15
-10
-5
0
5
10
15
20


30
25
20
15
10
5
0
-20
-15
-10
-5
0
5
10
15
20

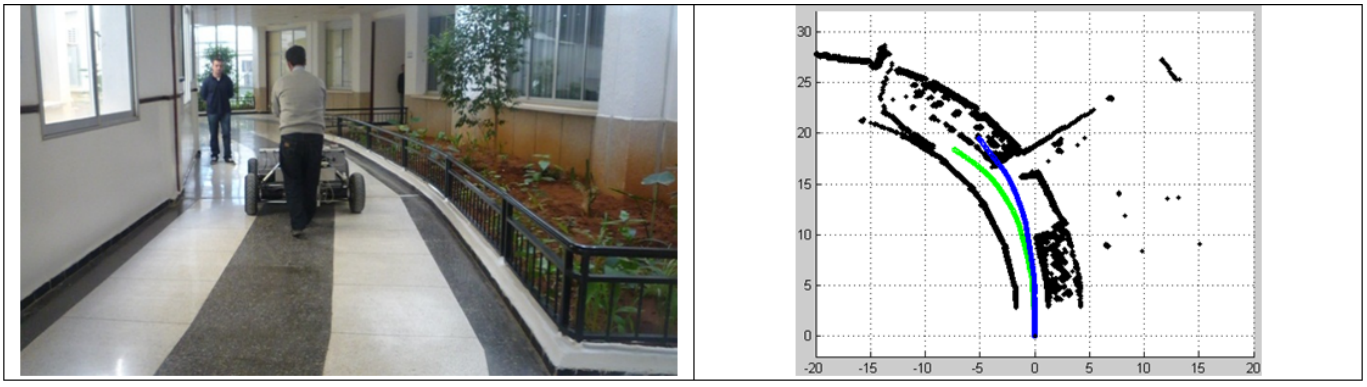

30
25
20
15
10
5
0
-20
-15
-10
-5
0
5
10
15
20

FIGURE 3.19 – Déroulement de l'expérience 2

3.5 Conclusion :

Dans ce troisième chapitre on a élaboré un modèle probabiliste du capteur LMS. Ce dernier a permis au Robucar d'accomplir avec succès l'expérience du *"scene undestanding"* par filtrage bayésien. Il s'agit là d'un résultat fondamental pour la suite du travail. Le modèle probabiliste du LMS est surtout utile en *"data-fusion"* lorsque le véhicule utilise plusieurs moyens sensoriels à la fois. La littérature sur ce thème est abondante. L'intérêt de ce travail en DATMO est justifié par le besoin du Robucar de se doter d'un tel module. L'expérience en ICP-SLAM le confirme et on en tire les résultats suivants :

→ On note en premier lieu l'efficacité de la technique ICP-SLAM pour cartographier des environnements statiques.

→ Sur la dernière figure de 3.19, on remarque que les points représentant l'intrusion du piéton dans la scène ont été sauvegardés. Cette expérience vient confirmer les remarques faites précédemment et indique clairement les limites de la technique ICP-SLAM dans les environnements dynamiques.

→ Enfin, on confirme ce qui a été dit au chapitre 1 quant à la grande dérive des odomètres des véhicules mobiles. On voit que les données odométriques en bleu deviennent rapidement erronées. En revanche, la localisation par ICP qu'on voit en vert est très précise. Cette constatation nous permet de considérer qu'on dispose d'un moyen fiable de localisation du robot, même dans des environnements dynamiques, du moment que les obstacles mouvants ne sont pas nombreux.

Chapitre 4

Implémentation du *Grid-based* DATMO

Afin de remédier aux problèmes signalés au chapitre 3, on propose un module de DATMO basé sur les grilles d'occupation. Ce dernier implémente la méthode du BOF pour l'estimation de vélocité des cellules dynamiques. Il sera donc complémentaire au module de ICP-SLAM, mais pas seulement. En effet, dans la dernière partie on va voir comment le module de DATMO va être relié au module de planification de trajectoires pour que le Robucar puisse planifier son chemin en tenant compte des vélocités des obstacles dynamiques.

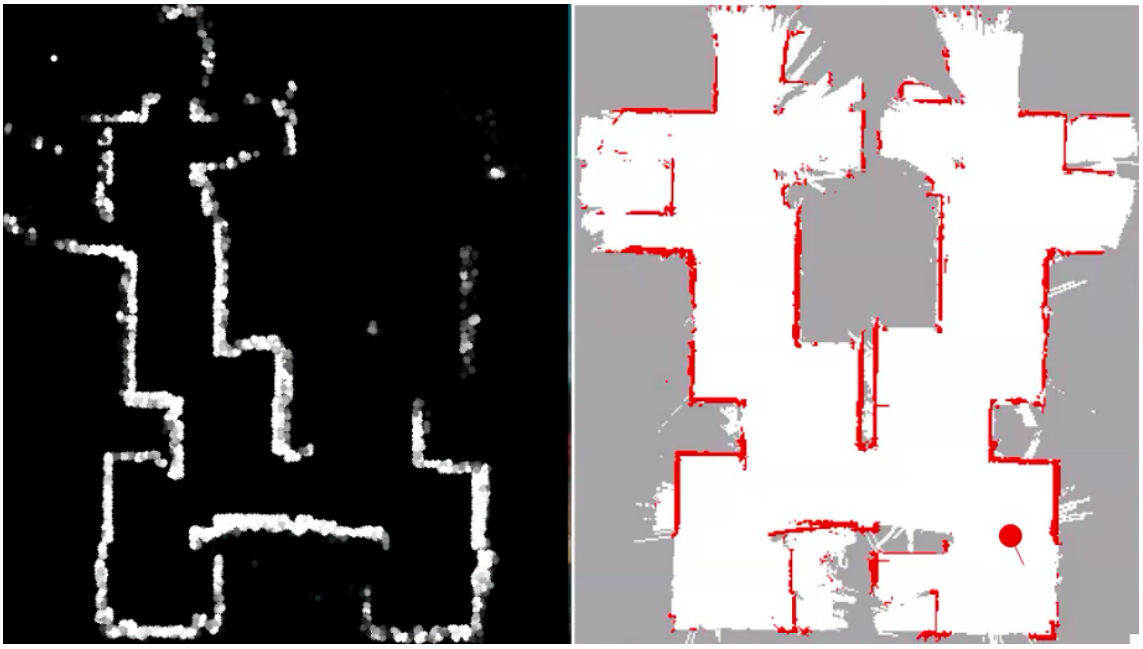


FIGURE 4.1 – La grille d'occupation satisfait les objectifs du SLAM, DATMO et planification du Robucar

Ce chapitre recueille les principaux résultats obtenus et présente aussi l'interface graphique conviviale que l'on a élaborée. Elle est intéressante pour diverses raisons : d'abord elle regroupe l'ensemble des programmes informatiques développés et permet de les combiner, ensuite elle constitue un module de DATMO assez complet et peut être embarquée sur différents robots.

4.1 L'interface graphique «W_DATMO»

Le module de DATMO n'a besoin que des trois modules GenoM suivants :

- ❑ « Communication » : pour la communication entre le robot et l'ordinateur distant.
- ❑ « Module_LMS » : pour l'affichage des données numériques du laser.
- ❑ « Localisation » : pour la localisation du robot à tout instant. Dans notre cas on acquiert à la fois les données de localisation odométrique et les données de localisation par la technique ICP_SLAM.

Le lancement de ces modules se fait simultanément et on peut suivre l'évolution des données qui défilent sur la fenêtre correspondante à chaque module :

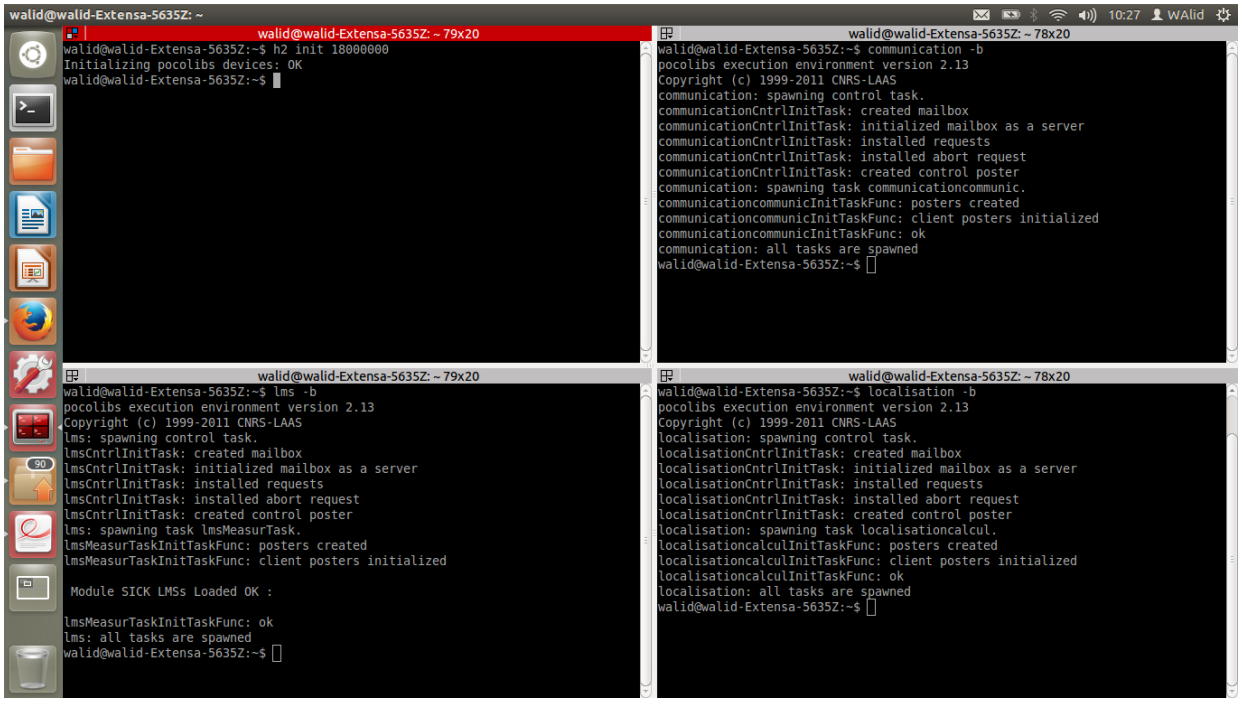


FIGURE 4.2 – Lancement simultané des différents modules essentiels en DATMO

Le but de ce travail est de réaliser un module DATMO qui exploite ces modules existants, le laser étant l'outil le plus important. Le cycle du LMS est de 13.33 ms (fonctionnement à 75Hz). C'est-à-dire qu'à chaque seconde il effectue 75 scans. L'interface graphique « W_DATMO » exploite ces données et permet de les visualiser et de reconstituer la scène réelle.

Scénario 3 :

« Le robot scrute la scène de la figure 4.3 et défile les données numériques correspondant aux mesures acquises par le module LMS. Les murs et le carton représentent des obstacles statiques. Un piéton entre dans la salle par la porte de droite et se dirige vers le carton puis il repart dans la direction du Robucar»

FIGURE 4.3 – Scène de l'expérience 3 effectuée aux locaux du CDTA

L'interface graphique ne se sert pas de la technique ICP utilisée lors de la première expérience. Elle récolte et interprète directement les données numériques du module LMS avant de les transformer vers une autre modélisation plus commode pour les environnements dynamiques.

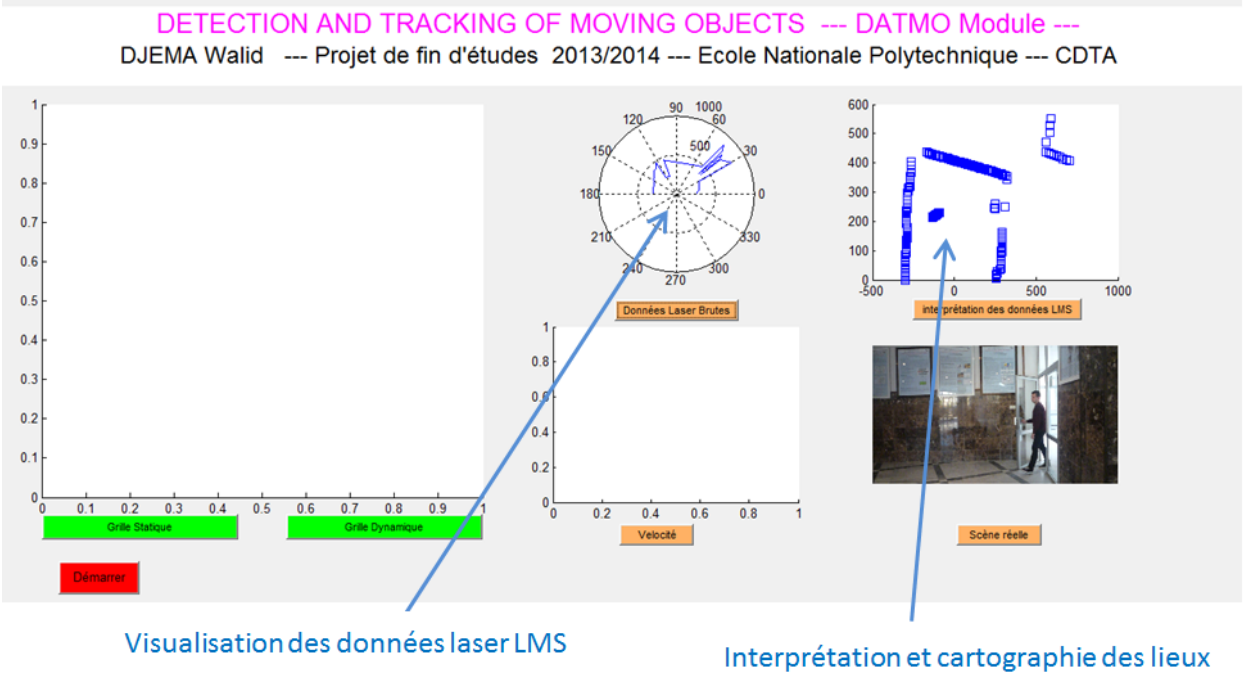


FIGURE 4.4 – Visualisation et interprétation des données numériques du module LMS

Le déroulement du sénario 3 peut être visionner suite au traitement par l'interface des données LMS :

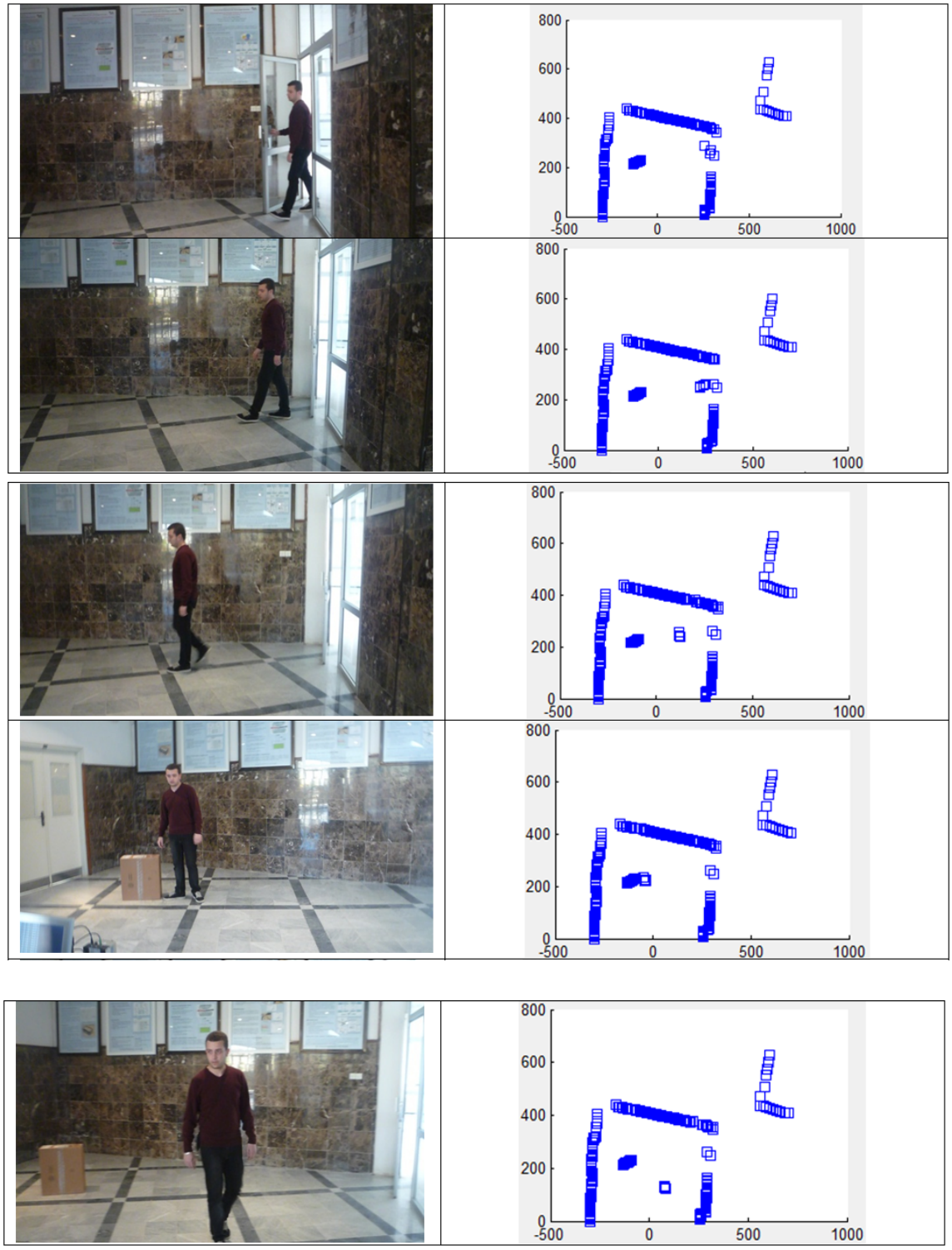


FIGURE 4.5 – Déroulement de l'expérience 3

Une fois qu'on est assuré que l'interface graphique interprète correctement les données laser, on passe alors à la seconde étape qui est la séparation entre les obstacles dynamiques et statiques. Pour y parvenir, on modélise d'abord l'environnement du robot (scène) par la méthode des grilles d'occupation illustrée au chapitre 2. Ensuite, on exploite l'algorithme intitulé «*Fast classification of static and dynamic environment*». Cet algorithme est très récent et le détail de son développement se trouve à la référence [51].

4.2 Modélisation de la scène par la méthode des grilles d'occupation

On associe au Robucar une grille locale d'une taille bien déterminée. En général, ce choix se fait selon la puissance et les capacités du calculateur du robot. Pour notre exemple, on choisit une grille rectangulaire de $72m^2$.

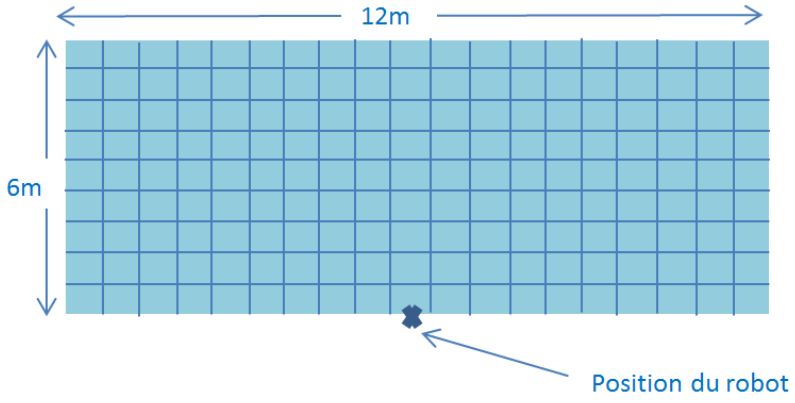


FIGURE 4.6 – Représentation qualitative de la grille locale associée au robucar

La taille d'une cellule de la grille importe plus que la taille de la grille globale. Pour notre cas, on a choisi de prendre la même taille de cellules que celle considérée dans les travaux de l'équipe-projet E-motion de l'INRIA et la standford university. Ces derniers utilisent des cellules carrées de 20 cm de côté.

On résume ainsi les caractéristiques de la grille associée au Robucar :

Nature :	Locale
Superficie :	72 m²
Longueur :	12m
Largeur :	6m
Taille de la cellule :	20cm x 20cm
Nombre de cellules :	1800

FIGURE 4.7 – Choix de la grille associée au Robucar

En cliquant sur le bouton *« Démarrer »* de l'interface on visualise la modélisation de la scène par la méthode de la grille. On visualise ainsi le déroulement du sénario sur la grille.

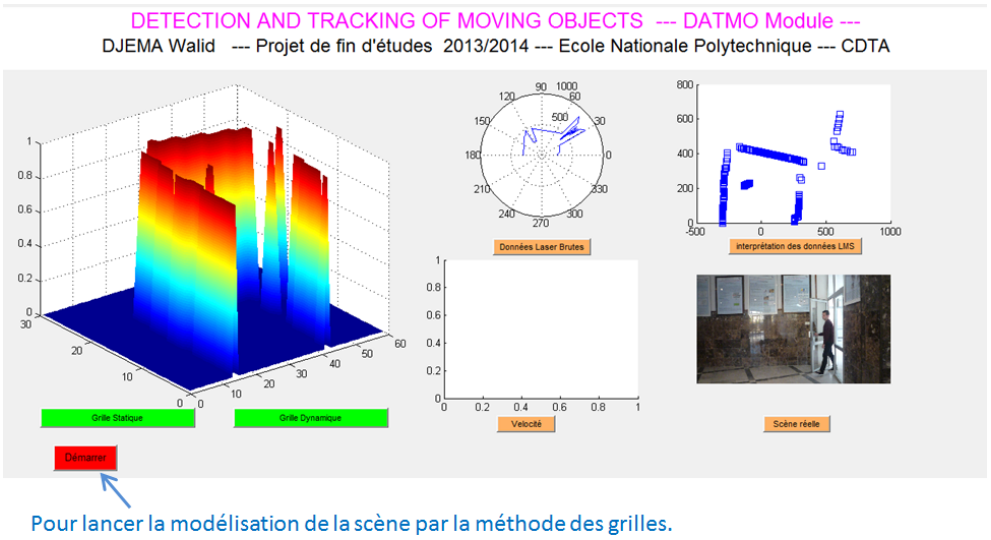


FIGURE 4.8 – Modélisation et visualisation de la scène par une grille d'occupation

Différentes situations à des instants distincts sont illustrées sur la figure ci-dessous :

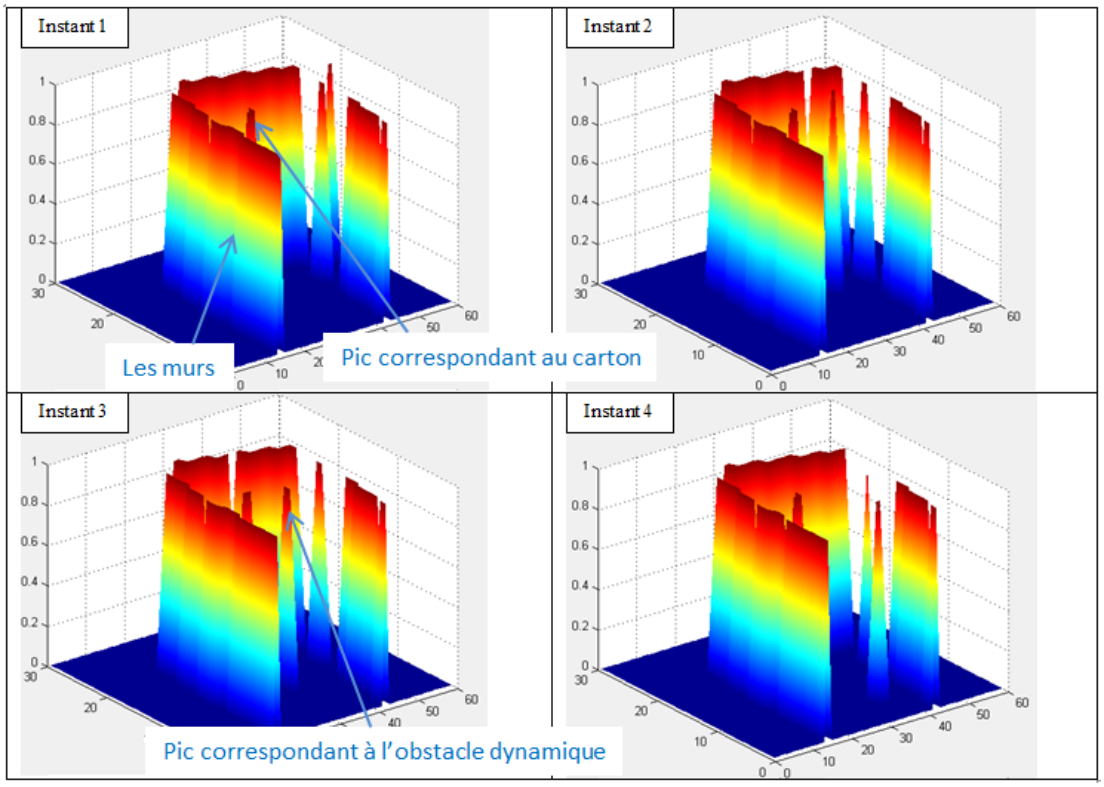


FIGURE 4.9 – Déroulement du sénario 3 vu par la grille d'occupation

4.3 Implémentation de l'algorithme *«Fast classification of static and dynamic environment»*

Dans cette expérience, on a exploité la représentation de l'environnement par la méthode des grilles d'occupation pour isoler l'environnement statique de celui dynamique. A chaque itération, des compteurs accumulatifs associés à chaque cellule de la grille comptent le nombre de fois qu'une cellule (parmi les 1800 cellules de la grille) a été détectée libre ou occupée.

Souvent en DATMO, on associe au robot une grille d'occupation locale et non pas globale. Cette spécification implique qu'à chaque mouvement du robot on doit tenir compte de la matrice de transformation homogène pour recaler la grille locale à un instant donné sur la grille locale à l'itération précédente. La technique d'ICP-SLAM fournit une bonne connaissance sur la localisation du Robucar et donc la considération du mouvement du robot n'a pas d'intérêt dans nos applications du DATMO. On a préféré maintenir le Robucar à l'arrêt et nous concentrer sur l'interprétation de la scène perçue par le capteur LMS-511.

Dans le cas où le Robucar se met en mouvement, la technique ICP de l'expérience 1 nous fournit à chaque itération la matrice de transformation homogène qui correspond au mouvement du robot et ainsi on recale la grille locale de l'itération t à celle de l'itération précédente $t-1$. Enfin, on note que cet algorithme nécessite un certain nombre d'itération avant de converger.

Ci-dessous la structure générale de l'algorithme à implémenter sur «W_DATMO» :

- **Initialization :**
  - $FreeCounter_t[i] = 1$, if $OG_t[i] < 0.5$
  - $OccupiedCounter_t[i] = 1$, if $OG_t[i] > 0.5$

- **Updating counters from previous time step:**
  - Mapping of cells of grid at time t-1 to grid at t
  - Update counters:
    - $FreeCounter_t[i] = FreeCounter_t[i] + FreeCounter_{t-1}[i]$
    - $OccupiedCounter_t[i] = OccupiedCounter_t[i] + OccupiedCounter_{t-1}[i]$

- **Motion:**
  - $MotionGrid[i] = \mathbf{F}(OGt[i], FreeCounter_t[i], OccupiedCounter_t[i])$
  - **F** can be a decision function, like:

$$MotionGrid_t[i] = \begin{cases} 1, & OG_t[i] > 0.5 \quad and \\ FreeCounter_t[i] > 2 * OccupiedCounter_t[i] \\ 0, & otherwise \end{cases}$$

FIGURE 4.10 – Etapes de l'algorithme *Fast classification of static and dynamic environment*

En cliquant sur le bouton *« Grille Dynamique »* on arrive à visualiser l'environnement dynamique entourant le Robucar :

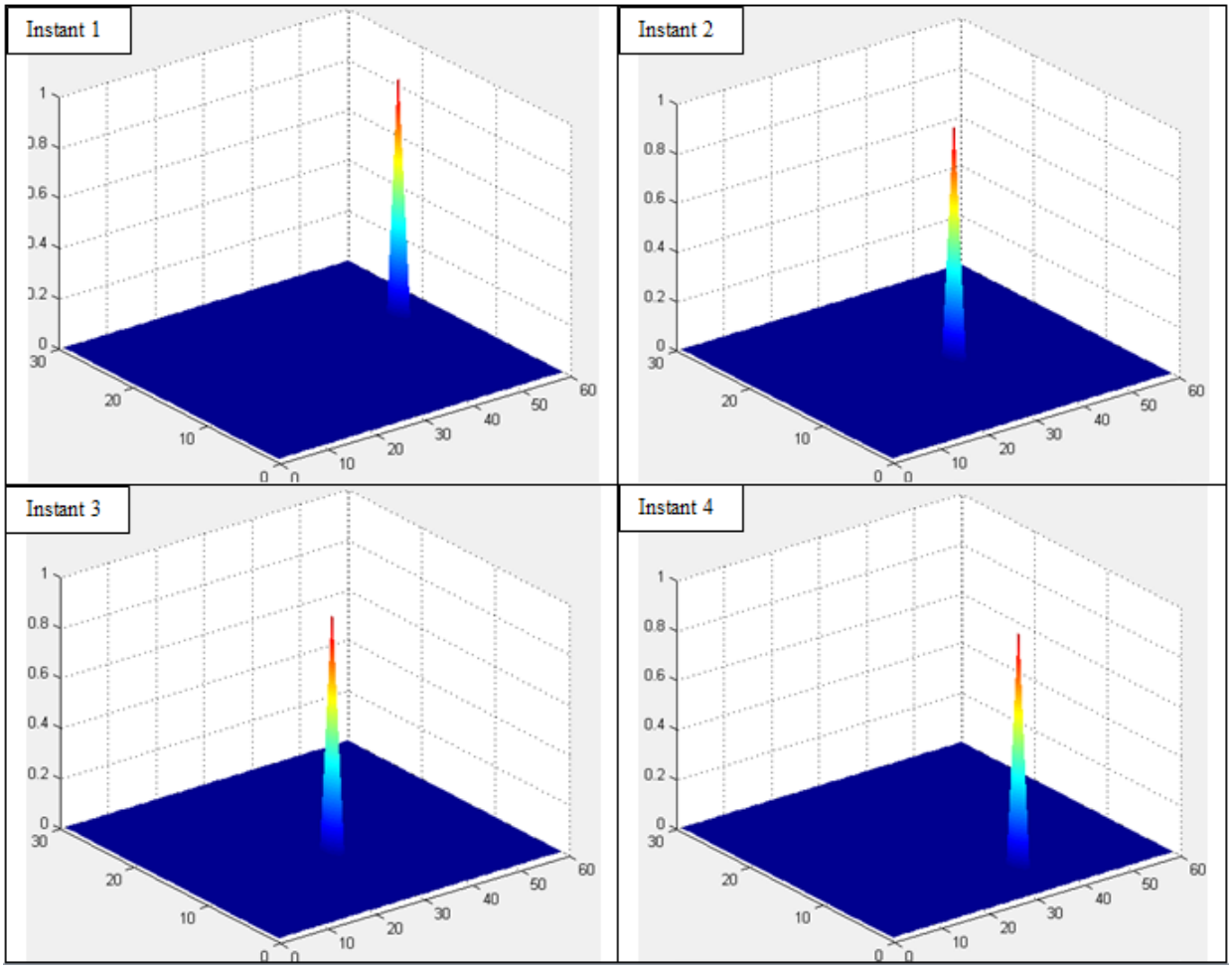


FIGURE 4.11 – Visualisation de l'environnement dynamique à différents instants

Commentaire :

Une première constatation lors de cette expérience est que, à des instants très brefs, des fausses détections peuvent apparaître. Ces fausses détections sont uniquement du type *«false positives»*, c'est-à-dire qu'on détecte des obstacles dynamiques là où ils n'existent pas. Au chapitre 1 de ce mémoire, nous avons comparé entre DATMO et ADAS et nous avons expliqué que les détections de ce type n'influent pas sur le fonctionnement du module DATMO contrairement à celui d'ADAS.

On note aussi l'efficacité et la robustesse de cet algorithme puisque l'objet dynamique a été détecté après quelques itérations seulement et puis il a été suivi dans son mouvement jusqu'à sa destination finale sans une seule fois disparaitre de la grille (en d'autres termes, aucune fausse détection du type *«false negatives»* n'a été signalée).

Le résultat qu'on vient d'obtenir est très important : d'abord il répond à la problématique de détection d'objets dynamiques pour le Robucar, ensuite il permet de corriger les cartes retracée par SLAM. La figure 4.12 est obtenue en inversant le critère de décision dans l'algorithme précédent. Ainsi, seul l'environnement statique apparaît.

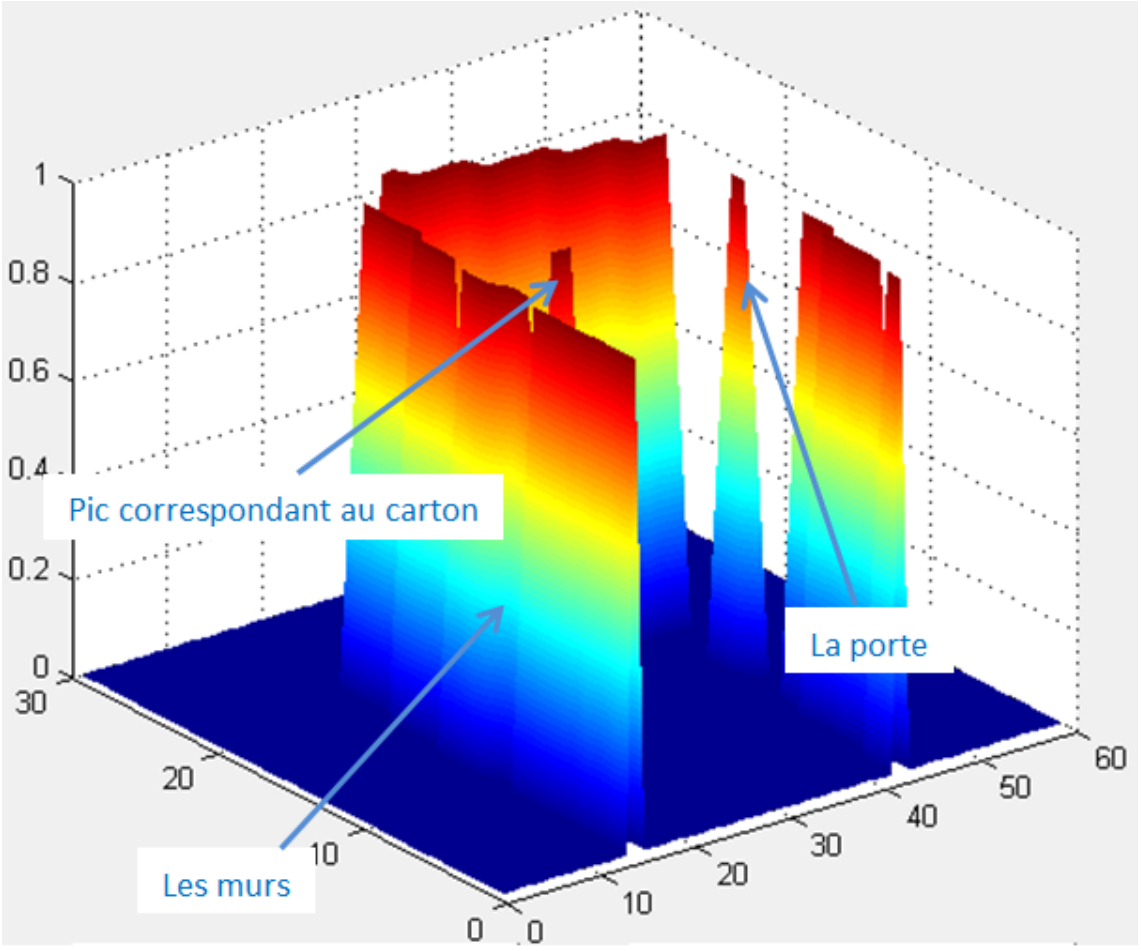


FIGURE 4.12 – Extraction de l'environnement statique entourant le Robucar

Le module de DATMO réalise la moitié de sa tâche, à savoir la détection d'objets dynamiques. Cette méthode de détection exige une modélisation de l'environnement par la méthode des grilles d'occupation et présente d'excellentes performances en implémentation et en temps de calcul. De plus, cet algorithme est particulièrement adapté à la détection des objets momentanément à l'arrêt.

A ce niveau, on est parvenu à retracer des environnements statiques. Cette démarche est très utilite en Grid-based SLAM. Par la suite, on s'intéresse à la prédiction du mouvement des obstacles dynamiques. L'algorithme qu'on vient de voir a été élaboré dans le but de s'implifier l'implémentation du BOF et de la réduire uniquement à une inférence en vélocité. Il en résulte une optimisation considérable en terme de temps de calcul et une réduction de 70% du taux de fausses détections comparé au BOF global.

4.4 Le "Bayesian Occupancy Filter (BOF)" pour l'estimation des vélocités

Jusqu'à présent on a réussi, dans une grille d'occupation modélisant la scène, à séparer les cases statiques des cases dynamiques. L'association de l'algorithme accomplissant cette tâche au principe du BOF illustré au chapitre 2 est devenue une nécessité. En effet, on a du mal à imaginer comment appliquer un filtrage bayésien pour toute case d'indice C (statique ou dynamique) même dans une grille de taille moyenne telle que la notre ($72\,m^2$).

La figure ci-dessous, illustre le principe d'enchainement entre les deux algorithmes :

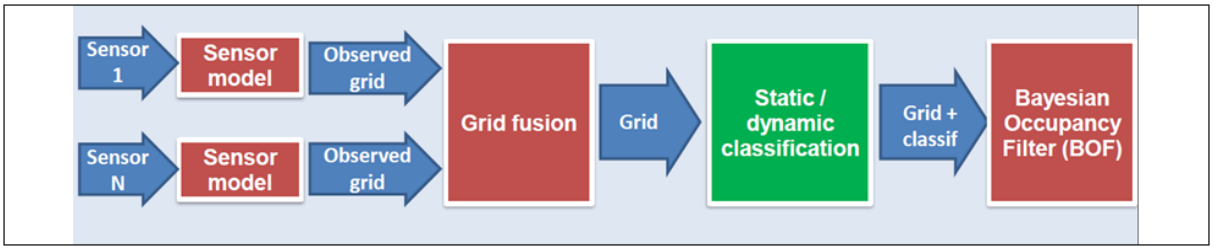


FIGURE 4.13 – Etapes et prétraitements avant l'application du BOF

Dans les récents travaux, l'algorithme de classification rapide des objets statiques et dynamiques est toujours utilisé comme entrée au BOF. Ainsi, l'estimation de vélocité n'est appliquée que sur les cases dynamiques. De plus, on retrouve dans [32], entre autres, cette nouvelle formulation du BOF,

$$P(A_c^{t-1}\, A_c^t\, O_c^t\, Z_1^t ... Z_S^t) \;=\; P(A_c^{t-1}) P(A_c^t | A_c^{t-1}) P(O_c^t | A_c^{t-1}) \prod_{i=1}^{S} P(Z_i^t | Ac^t\, O_c^t), \qquad (4.1)$$

où la variable discrète A désigne l'ensemble des cases antécédentes possibles de la case c, et S le nombre de capteurs utilisés. Souvent, en effet, plusieurs capteurs sont associés sur une même voiture autonome, d'une part pour fusionner les mesures d'une même grandeur et ainsi augmenter la précision de l'estimation, et d'autre part pour diversifier les mesures (plusieurs variables différentes). Quant au Robucar, il ne dispose que d'un capteur qui est le LMS et qui ne fournit aucune information directe sur la vitesse des objets. On doit donc passer par les *virtual sensors* (figure (2.8)) pour obtenir un capteur de vitesses. En bayésien il est très commun d'utiliser ces capteurs et leur principe est donné dans [22].

De la décomposition (4.1), on déduit que la vitesse V devient une variable implicite en partant du principe que **la connaissance des antécédents A d'une case permet d'estimer sa vélocité**. Néanmoins il est clairement indiqué, dans la même publication, que le champ des vitesses doit être discrétisé de la même manière que l'énoncé du BOF global illustré au chapitre 2. Par ailleurs, on note que cette reformulation du BOF n'est pas une simplification au sens où on émet des hypothèses. C'est juste une manière efficace d'implémenter ce dernier. La discrétisation des vitesses et l'échantillonnage d'itérations deviennent les critères d'évaluation les plus importants du module DATMO.

La figure ci-dessous illustre la structure de l'algorithme qu'on a élaboré dans le cas du Robucar et exploité par l'interface graphique «W_DATMO».

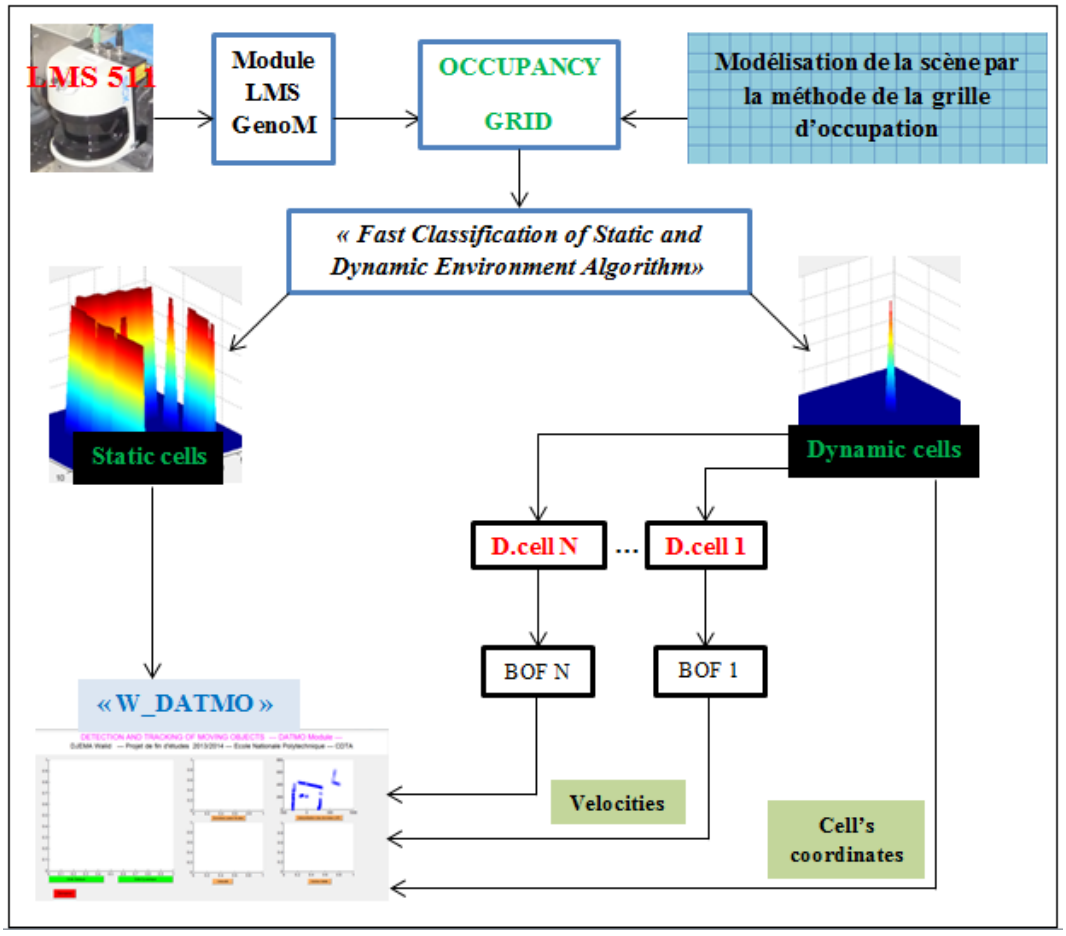


FIGURE 4.14 – L'implémentation du BOF va de l'acquisition des données à l'affichage des vitesses estimées

Dans notre approche, le champ des vitesses est discrétisé selon le module de la vitesse (noté V) et selon l'orientation (notée R). Le filtrage bayésien permet une estimation très large des vitesses et laisse le robot apprendre de ses expériences, contrairement à d'autres approches Markoviennes dans le domaine de prédiction de mouvement. On peut citer, parmi d'autres, [72] et [73]. Les réponses proposées par toutes ces approches restent approximatives mais souvent suffisantes au robot pour interagir avec son environnement. On le verra d'ailleurs lors de la planification de trajectoires.

L'implémentation des règles d'inférence est une étape très délicate. Un compromis doit être trouvé entre complexité et fiabilité. Les règles donnant les différentes mesures probables de vitesse sont de la forme :

$$\text{Si } [\{proposition\ 1\} \wedge \{proposition\ 2\} \wedge ... \wedge \{proposition\ n\}] \text{ alors } P_{\{V,R\}}, \tag{4.2}$$

où $P_{\{V,R\}}$ est une mesure probable de V et de R.

Si Z_V est une mesure de la vitesse d'une case donnée, et si Z_R est une mesure d'orientation, alors

$$P(V, R|Z_V, Z_R) = P(V|Z_V, Z_R)P(R|V, Z_V, Z_R). \tag{4.3}$$

Sauf que la connaissance d'une orientation ou de sa mesure probable n'apporte aucune information sur la vitesse ou la mesure de la vitesse. L'équation précédente s'écrit comme le produit de deux distributions indépendantes :

$$P(V, R|Z_V, Z_R) = P(V|Z_V)P(R|Z_R) \tag{4.4}$$

On choisit l'ensemble des points d'intérêts (POI) suivants :

$$\begin{aligned} Z_V \in \tilde{Z}_V &= \left\{V_{max}, \tfrac{7}{8}V_{max}, \tfrac{3}{4}V_{max}, \tfrac{5}{8}V_{max}, \tfrac{1}{2}V_{max}, \tfrac{3}{8}V_{max}, \tfrac{1}{4}V_{max}, \tfrac{1}{8}V_{max}\right\} \\ Z_R \in \tilde{Z}_R &= \left\{0, \tfrac{\pi}{4}, \tfrac{\pi}{2}, \tfrac{3\pi}{4}, \pi, \tfrac{5\pi}{4}, \tfrac{3\pi}{2}, \tfrac{7\pi}{4}\right\} \end{aligned} \tag{4.5}$$

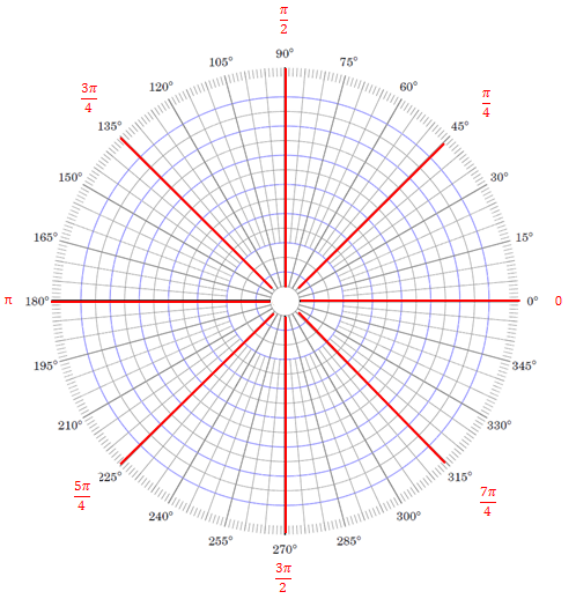


FIGURE 4.15 – Les directions phares considérées lors de l'inférence selon R

On rappelle que l'inférence, dans la logique du BOF, n'envisage aucune association entre les différentes cellules. L'absence de la notion de *piste* ou d'*objet* dans cette représentation permet de s'affranchir des difficultés de la phase d'association de données. Néanmoins, il est important d'associer aux cases suivies un "ordre de grandeur" qui définit la manière d'estimer leur mouvement, sans pour autant chercher à classifier ou à regrouper des cases entre elles.

Dans *«Awareness of Road Scene Participants for Autonomous Driving»* [22], on classe l'environnement dans lequel la voiture autonome évolue en trois classes : *« pedestrian zone driving »*, *« freeway driving »* et *« mixed urban driving»*. La première scène contient beaucoup d'adultes, d'enfants et de vélos comme dans un centre-ville. La deuxième scène est une autoroute dégagée et la troisième regroupe des éléments des deux précédentes.

Avec le LMS réglé à 75 Hz, on peut suivre à chaque itération les cases dont la vitesse est de l'ordre de 90 Km/h. En revanche, si la case occupée correspond en réalité à un piéton marchant à 1.2 m/s, il est inutile de vouloir le traquer à cette cadence. L'équation (4.1) donne le critère de sélection : "entre une itération et une autre la case en mouvement ne doit bouger que d'une case au maximum". On se sert de ces données pour implémenter un programme capable de traquer des cibles dans les trois environnements citées précédemment. Les premières itérations qui suivent la détection d'une case définissent la formule à appliquer. Pour le cas d'un piéton, on tient compte des réflexes humains définis dans la 1ère section (0.1 à 1 s). La vitesse maximale dépend donc du choix de la formule à appliquer pour chaque case. La vitesse d'un piéton peut varier dans un intervalle donné par $[0.2, 2m/s]$.

Par ailleurs, quelque soit la nature de la cible traquée, les règles d'inférence selon R ne changent pas. La figure ci-dessous montre l'ensemble des points d'intérêts (en bleu) dans le cas d'un piéton en prenant $V_{max} = 2$ m/s.

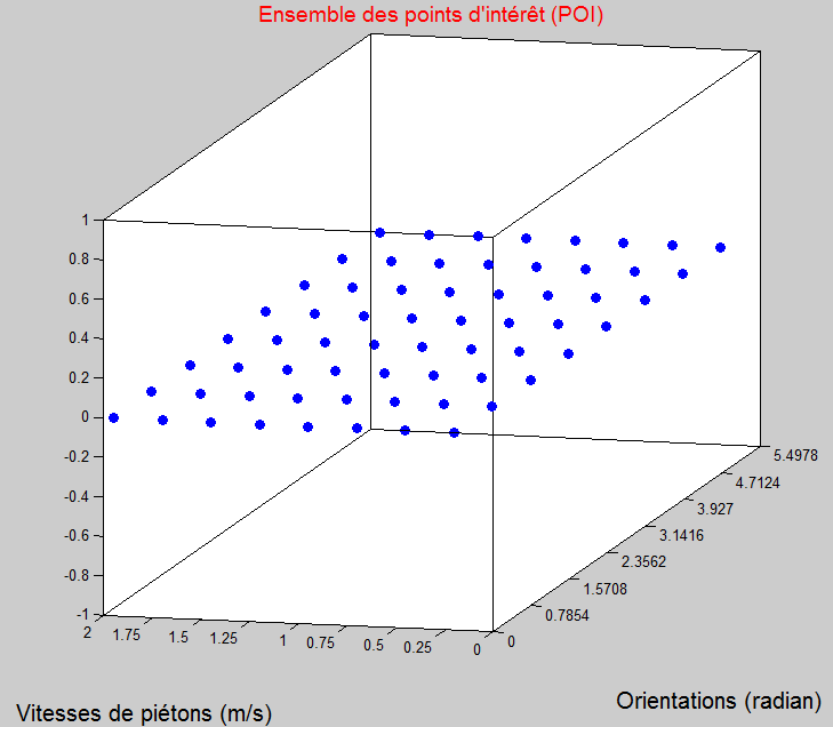


FIGURE 4.16 – Couples (V, R) qui donnent les profils principaux de vitesse

Chaque situation dans les règles d'inférence conduit à une mesure virtuelle de la vitesse. Le principe des *virtual sensors* impose de corrompre ces mesures par un bruit gaussien. Dans notre cas, puisqu'il s'agit de mesurer une vitesse, on a pris un écart-type relatif à l'accélération. Le BOF s'applique sur le résultat de l'étape d'inférence qui est en fait un ensemble de mesures probables de la vitesse sur un certain intervalle de temps (1 s pour la formule "piéton").

Le filtre bayésien prend en considération l'ordre chronologie de la prise de ces mesures et estime la vitesse dans un cycle de prédiction/estimation. Il tient également compte de toutes

les probabilités de mouvements enregistrées. Ici encore les modèles de capteurs interviennent durant l'étape d'estimation. Selon la formule désignée, les modèles de capteurs $P(Z_R|R)$ et $P(Z_V|V)$ vont être différents.

Pour une formule "piéton", le modèle Brownien décrit au mieux le comportement humain. La caractéristique principale des piétons et qu'ils peuvent rapidement changer de direction et d'orientation. Le mouvement est non prédectif (impulsif). Par conséquent, $P(Z_R|R)$ est une gaussienne avec un écart-type assez large et qui modélise une grande incertitude vis-à-vis de la direction. En revanche, les erreurs en vitesse V sont moins importantes. Toujours dans le cas d'un piéton, le résultat d'une estimation de vélocité (V, R) pour une case donnée peut être n'importe quel point de la grille verte sur la figure ci-dessous. La combinaison de toutes les incertitudes sur les mesures et le modèle du capteur laisse ainsi un large choix au robot pour estimer les vélocités.

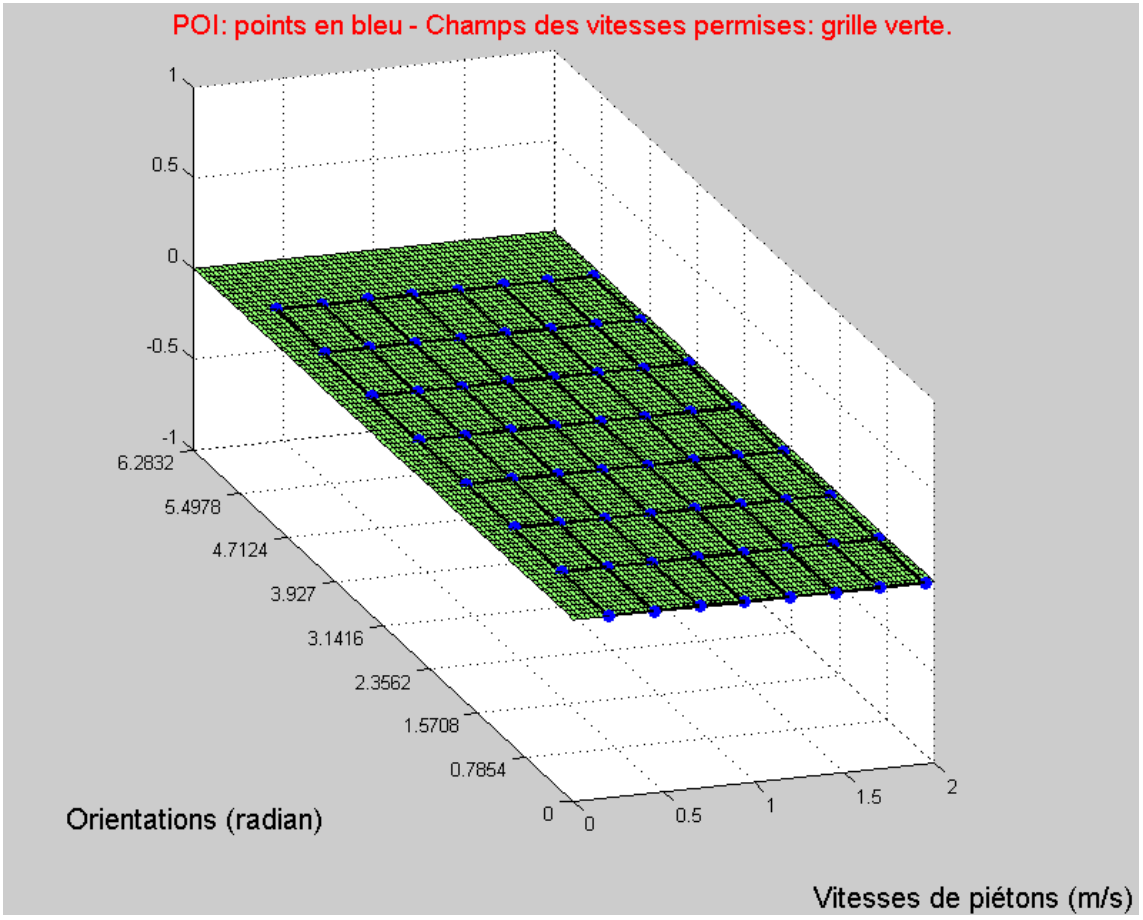


FIGURE 4.17 – Surface d'estimation de vélocité (V, R) pour une cellule occupée

Commentaire : Contrairement aux méthodes classiques d'association de données, le BOF ne tient pas compte de la notion de *"piste"*. Cette stratégie évite beaucoup de situations ambiguës ou de décisions prématurées. Néanmoins, elle cause certaines difficultés d'ordonnancement chronologique des probabilités de mesures.

4.4.1 Implémentation des fonctions de vraissemblance

On rappelle que les fonctions de vraissemblance codent la manière dont on acquiert les observations. Les mesures de vélocité d'une case donnée : $(Z_V,\ Z_R)$ sont acquises par un ensemble de règles d'inférence basées sur l'observation des antécédents possibles. Z_V et Z_R sont à la base un des 64 couples possibles formant les POI (figure 4.16). La logique des capteurs virtuels impose de corrompre ces mesures par un bruit gaussien. On obtient ainsi les mesures probables $P_{\{V,R\}}$ que le BOF intègre chronologiquement dans les étapes de prédiction et d'estimation.

On constate qu'on ne connait pas a priori la valeur exacte de $P_{\{V,\,R\}}$. En revanche, on a besoin d'implémenter le *measurement model*, $P(Z_V\ Z_R|P^*_{\{V^*,\,R^*\}})$, avec $P^*_{\{V^*,\,R^*\}}$ une mesure de la vitesse et de l'orientation supposées connues.

En DATMO, la distribution $P(Z_V\ Z_R|P^*_{\{V^*,\,R^*\}})$ suit une loi normale à deux dimensions qui admet la densité de probabilité suivante :

$$p_{\Omega,\Sigma}(Z) \quad = \quad \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(Z-\Omega)^T\Sigma^{-1}(Z-\Omega)}, \tag{4.6}$$

avec

$$\begin{cases} Z = (Z_V, Z_R)^T, \\ \\ \Sigma = \begin{pmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_R^2 \end{pmatrix} \\ \\ \Omega = (V^*, R^*)^T. \end{cases} \tag{4.7}$$

Les écarts-types σ_V et σ_R dépendent du modèle dynamique choisi. En effet, une variance σ_R^2 assez large caractérise le modèle Brownien des piétons et s'explique par le caractère impulsif de ces derniers. Pour les voitures, le modèle linéaire est utilisé : une plus grande valeur σ_V^2 est considérée en diminuant au même temps σ_R^2. Le *bicycle model* considère une grande dérive pour les deux variances.

La figure 4.18 donne la fonction de vraissemblance qui accompagne le modèle Brownien pour $\sigma_R = \frac{\pi}{5}\ (rad)$ et $\sigma_V = 0.3\ (m/s)$. Sur cette même figure, la mesure $P^*_{\{V^*,R^*\}}$ est prise égale à $(1\,m/s,\ \pi\,rad)$ et on observe l'allure de la fonction $P(Z_V\ Z_R|P^*_{\{V^*,R^*\}})$.

Il est important de noter que cette fonction n'est pas une distribution de probabilité au sens propre du terme. Elle doit être normalisée par le terme $P(Z_V\ Z_R)$. On implémente ce dernier par des lois de marginalisation.

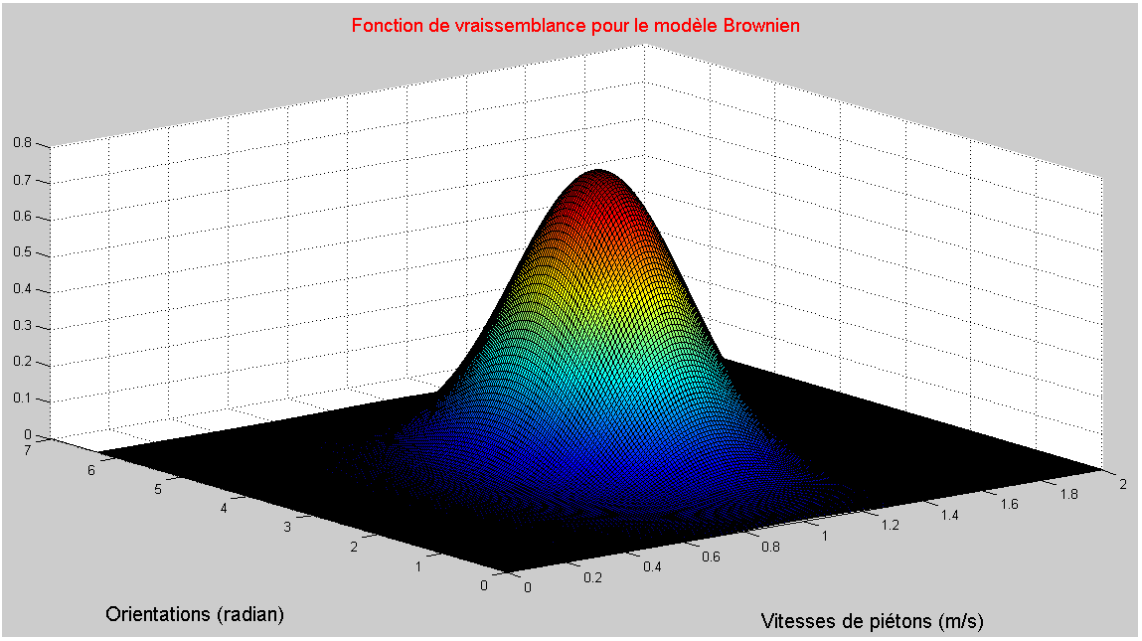


FIGURE 4.18 – Fonction de vraissemblance pour $P^*_{\{V^*,R^*\}} = (1\,m/s,\ \pi\,rad)$

4.4.2 Implémentation du BOF et application à l'estimation de vélocités

Le programme d'estimation des vitesses est formé d'un seul bloc qui exploite tous les résultats obtenus jusqu'à présent. Il va donc de l'acquisition des données LMS à l'application du filtre bayésien qui construit la meilleure croyance de vélocité, en passant par la représentation par grille d'occupation, l'extraction de la grille dynamique, l'échantillonnage des grilles selon la nature des objets à suivre, et enfin l'inférence et les pseudo-mesures de vitesses par ordre chronologique.

Avant d'arriver à l'exploitation des estimées de vélocités et à leur affichage sur l'inteface graphique (Cf. figure 4.14), on montre comment évolue l'estimation d'une case dynamique à un instant donné. On reprend les données réelles de l'expérience 3 et on applique le BOF à l'estimation d'une case dynamique. Plus particulièrement à la case définie par ses coordonnées ($e = 17$, $m = 42$) dans la grille d'occupation locale associée au Robucar. L'algorithme de séparation statique/dynamique avait détecté cette case comme occupée à un instant t donné et on aimerait estimer sa vélocité pour les instants à venir.

Les pseudo-mesures de vélocités constituent les couples $P_{\{V,\,R\}}$ qui sont en réalité les POI les plus probables, pour cette case, augmentés d'un bruit gaussien. L'extraction de ces mesures est illustrée sur la figure 4.19 où on montre l'ordonnancement chronologique lors de l'acquisition de ces données.

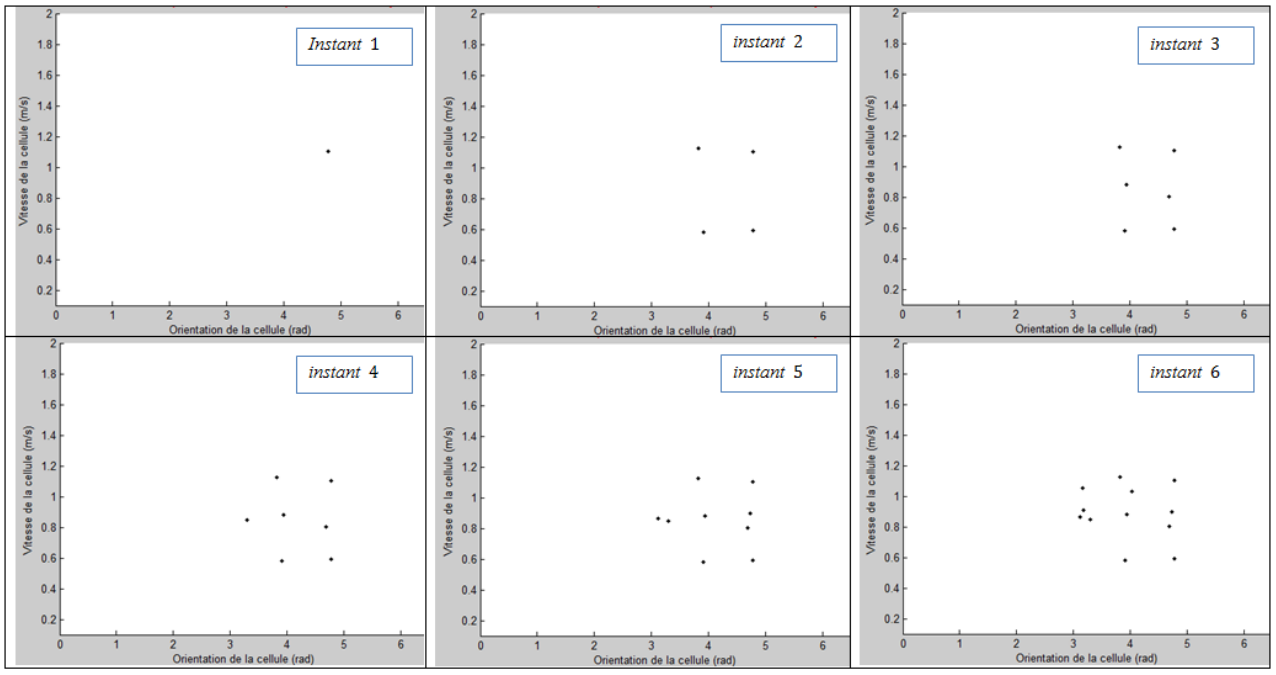


FIGURE 4.19 – Ensemble des couples $P_{\{V,R\}}$ pour la case ($e = 17$, $m = 42$)

Le BOF implémente ces connaissances et les combine en considérant la fonction de vraisemblance précédemment introduite, dans des cycles de prédiction/estimation qui, au final, fournissent la meilleure estimation de vélocité pour la cellule à l'instant considéré. La figure 4.20 montre une surface équiprobable ($P(i,j) = \frac{1}{400}$) indiquant l'absence de connaissances préalables.

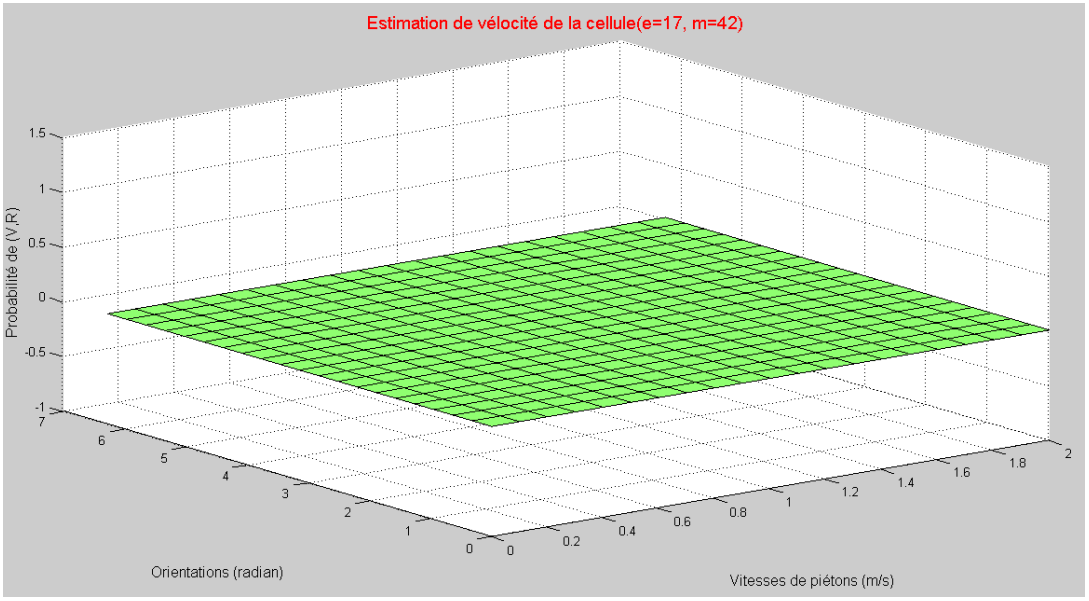
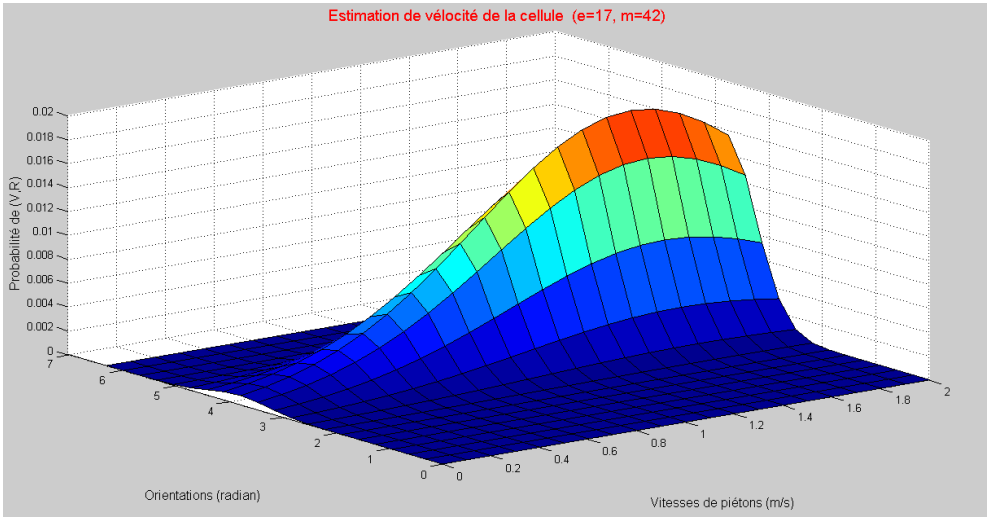
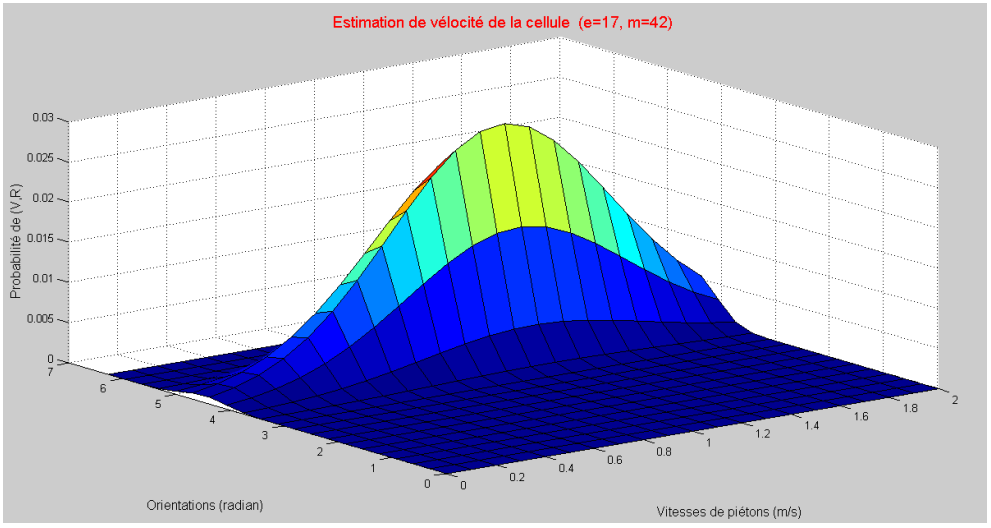
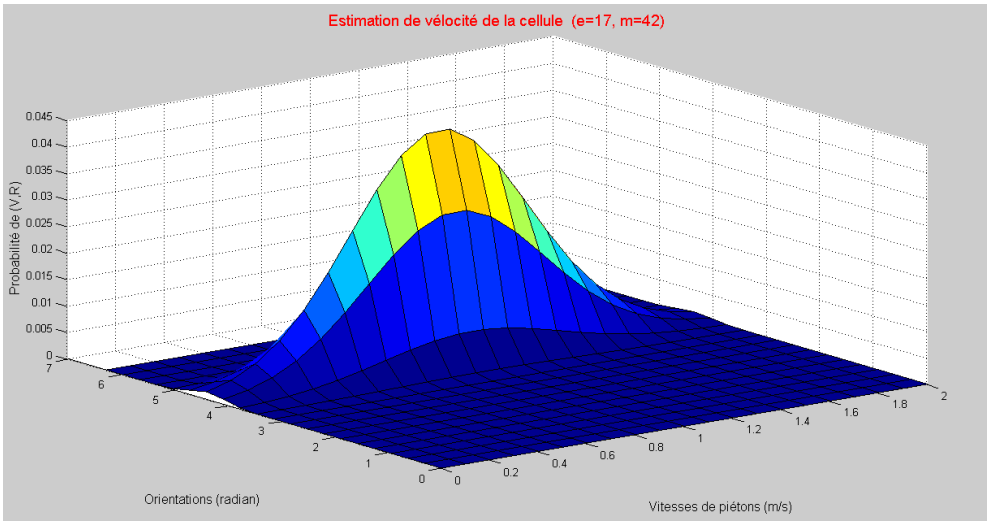


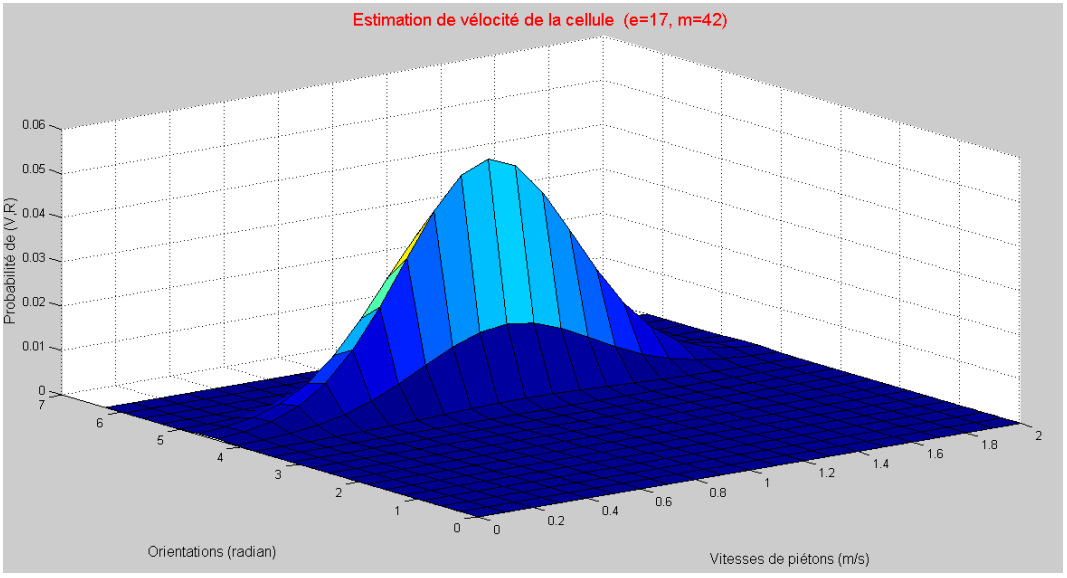
FIGURE 4.20 – La surface équiprobable pour ($e = 17$, $m = 42$) avant le début de l'estimation

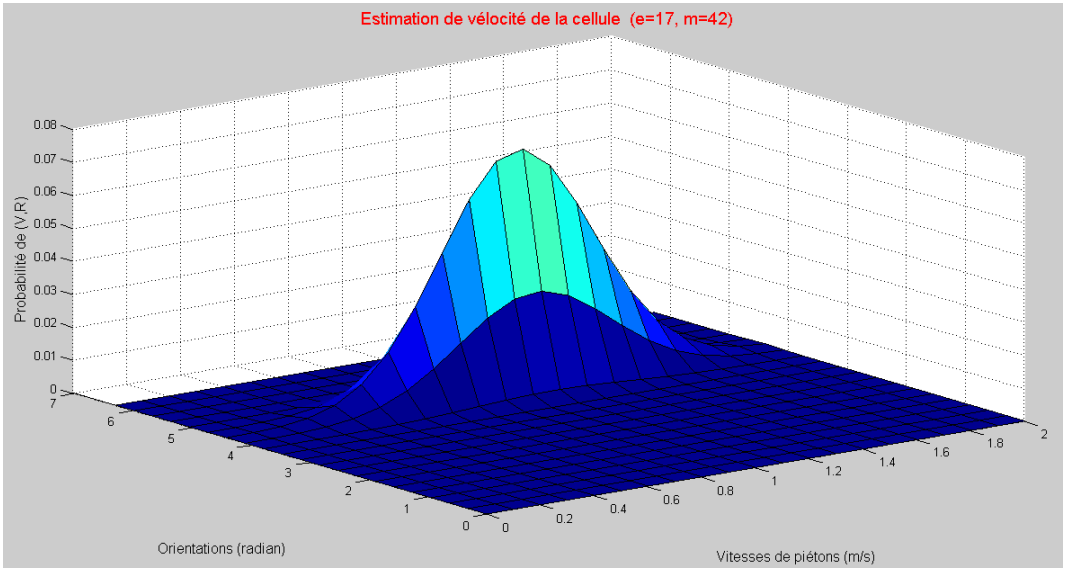
Les figures qui suivent illustrent comment le robot construit sa croyance par le cumul d'expériences. On note sur l'axe vertical l'évolution des valeurs de probabilité pour chaque couple $(V,\ R)$.

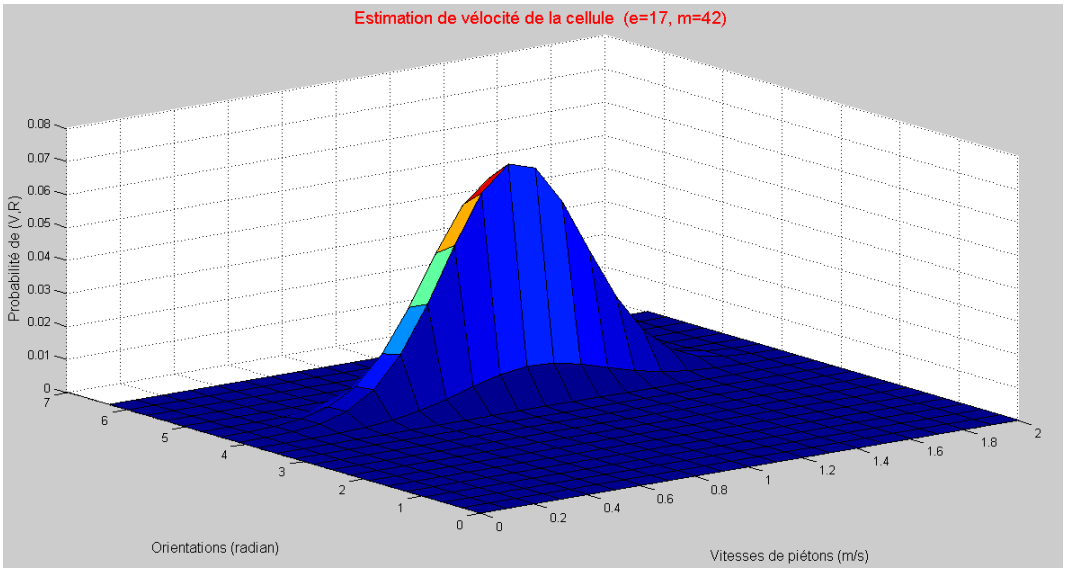







Estimation de vélocité de la cellule (e=17, m=42)
Probabilité de (V,R)
0.06
0.05
0.04
0.03
0.02
0.01
0
7
6
5
4
3
2
1
0
Orientations (radian)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Vitesses de piétons (m/s)


Estimation de vélocité de la cellule (e=17, m=42)
Probabilité de (V,R)
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
7
6
5
4
3
2
1
0
Orientations (radian)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Vitesses de piétons (m/s)


Estimation de vélocité de la cellule (e=17, m=42)
Probabilité de (V,R)
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
7
6
5
4
3
2
1
0
Orientations (radian)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Vitesses de piétons (m/s)

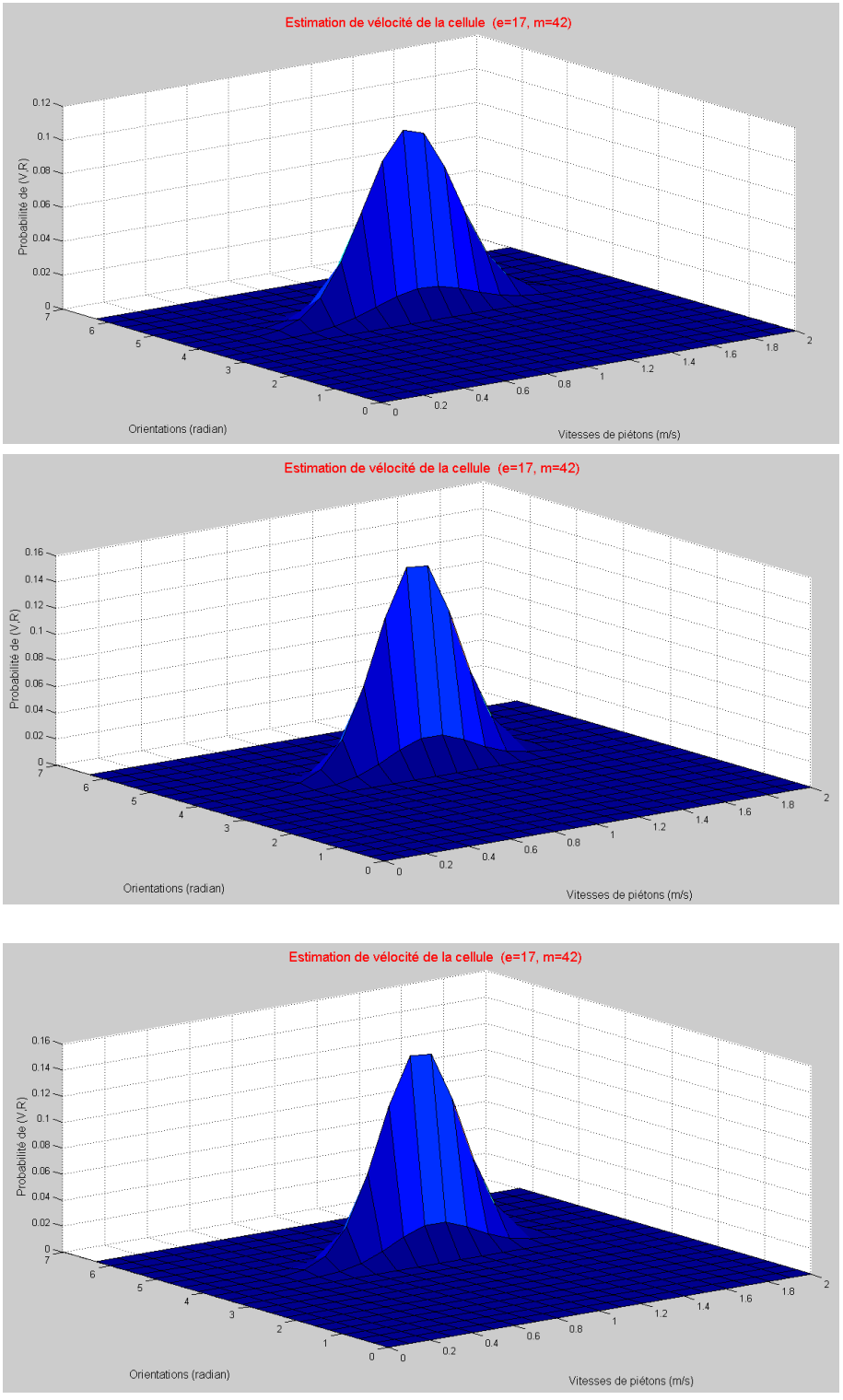
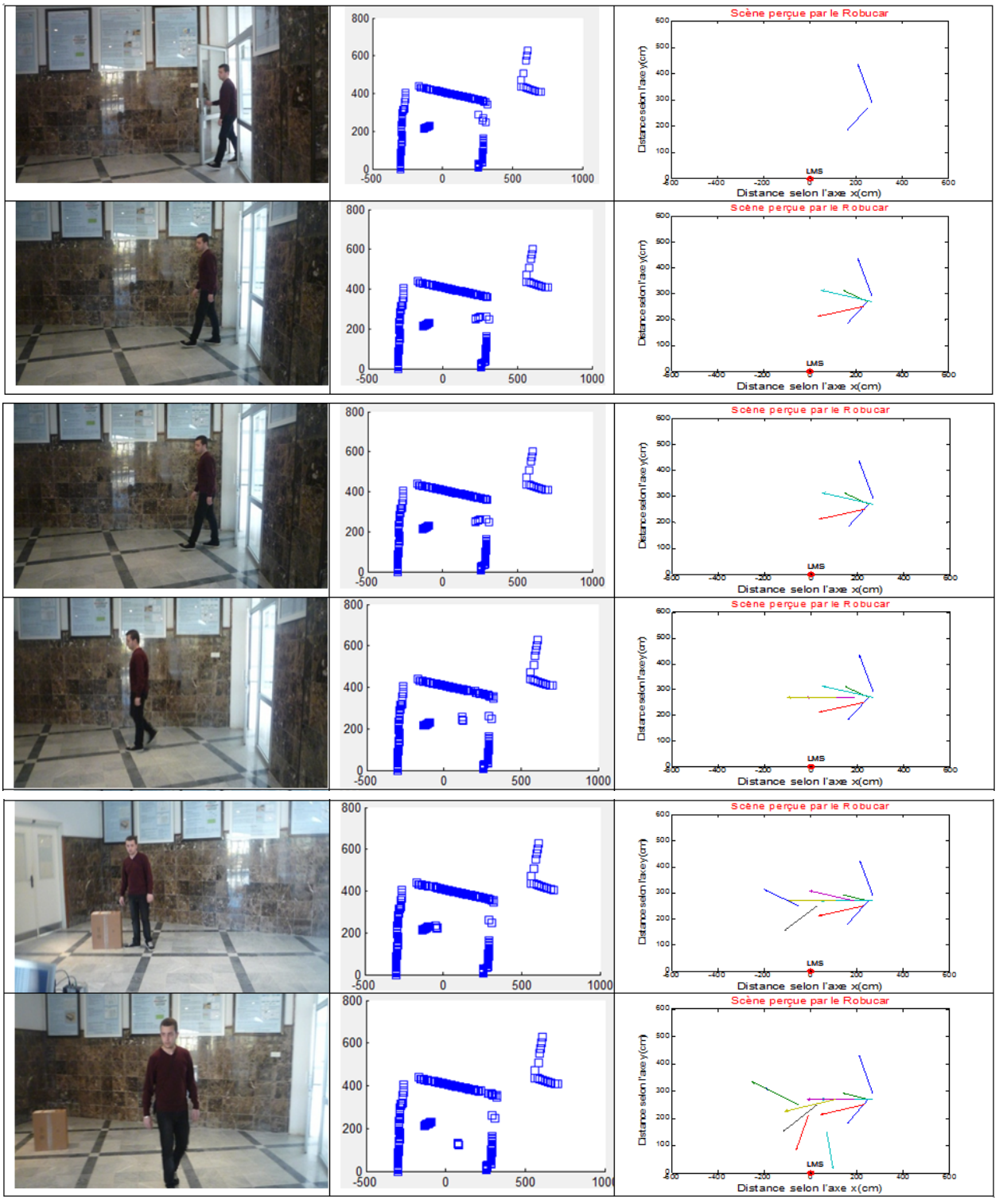


FIGURE 4.21 – Distribution a posteriori donnant l'estimée de vélocité de la cellule considérée

La valeur retenue est celle donnée par le couple (V, R) qui a la plus grande probabilité à la fin de l'estimation. Pour cet exemple on trouve : $(V_{estimee}, R_{estimee}) = (0.9m/s, 3.7699rad)$.

Evidemment il s'agit d'une valeur approximative mais suffisante au robot pour interagir avec ces obstacles. Un compromis doit être trouvé entre précision de l'estimation et temps de calcul car le BOF fait partie des algorithmes. En DATMO il est commun de considérer le *"constant velocity model"* $P(v_{t+1}|v_t)$ où la vitesse reste constante durant l'intervalle $[t,\, t+1]$. L'interface graphique exploite le BOF pour afficher les vecteurs de vitesses les plus probables des cases occupées à chaque seconde. Comme illustré par la série de figures suivante :



4.5 Apport du BOF par rapport à l'approche classique

On revient sur les performances du BOF (par le biai) de l'approche classique. Pour cette dernière, la perception des situations complexes de conduite est basée sur des algorithmes de poursuite multicapteur et multipiste. Les différents acteurs (voitures, piétons, etc) et les difficultés posées par le milieu urbain (nombreuses occlusions, apparitions et disparitions) rendent la poursuite multipiste peu fiable, et par conséquent le contrôle du véhicule impossible.

Dans [50], on explique que ce manque de fiabilité tient aux décisions prises au cours de la phase d'association pistes/observations de la poursuite, qui entraine une perte d'information trop importante. Une représentation alternative de l'environnement du véhicule a été proposée. Celle-ci est basée sur les grilles d'occupation et évite ainsi toute prise de décision prématurée en s'affranchissant de la notion de piste. On va revenir d'abord sur les principaux algorithmes d'association de données puis les comparer au BOF.

Poursuite multicible : problèmes spécifiques et solutions classiques

La principale difficulté de la poursuite multiobjet vient du fait qu'on ne sait pas associer *a priori* une observation du capteur à un objet présent dans le volume d'observation du capteur [50]. La difficulté est renforcée si on ne connaît pas le nombre d'objets présents dans le volume d'observation, et si ce nombre peut évoluer au cours du temps.

Les algorithmes classiques cherchent à maintenir une liste de *pistes*, c'est à dire une liste d'objets identifiées et actuellement poursuivis. Quand une nouvelle série d'observations capteur est disponible, la première chose à faire est d'associer ces nouvelles observations aux pistes maintenues, c'est à dire de décider quelle piste est à l'origine de chaque observation. Au vu de cette association, la phase de maintenance consiste à décider si on doit :

- ❏ Créer de nouvelles pistes si un nouvel objet non encore poursuivi est apparu ;
- ❏ Détruire d'anciennes pistes si des objets disparaissent ;
- ❏ Maintenir une piste existante, qu'une nouvelle observation lui ait été attribuée ou non.

La phase de filtrage n'est pas spécifique à la poursuite multipiste, puisqu'elle consiste pour chaque piste en la mise à jour de la prédiction de l'état de l'objet. Ceci est généralement fait à l'aide d'un filtre de Kalman. Enfin, la dernière phase est une phase de fenêtrage. Elle a pour but de simplifier la phase d'association : une observation capteur pourra être considérée comme pouvant provenir de cette piste si sa distance à l'état prédit de l'objet est inférieur à un seuil. Déterminer ce seuil n'est pas trivial. La "fenêtre de validation" doit être suffisamment grande pour qu'un impact provenant d'une piste s'y trouve et suffisamment petite pour pouvoir efficacement filtrer les autres impacts. Les deux phases spécifiques à la poursuite multicible, association piste-observation sont en générale résumées sous le terme d'**association de données**.

- *Méthode du plus proche voisin :*

  Cette méthode est la plus simple et la plus populaire grâce à son très faible coût calculatoire. Pour chaque nouvel ensemble d'observations, le but est de trouver l'association la plus probable entre une observation et une piste existante ou entre une nouvelle observation et l'hypothèse d'une nouvelle piste. La décision est immédiate et irrévocable : l'hypothèse la plus probable est considérée comme vraie. Ce qui induit à de mauvaises décisions.

- *Joint Probability Data Association Filter "JPDAF" :*

  Il s'agit d'une extension d'un plus ancien algorithme, le *PDAF*, la différence entre les deux réside dans l'évaluation des probabilités des associations. Pour la poursuite multiobjet, le nombre d'objets M est supposé connu et constant.

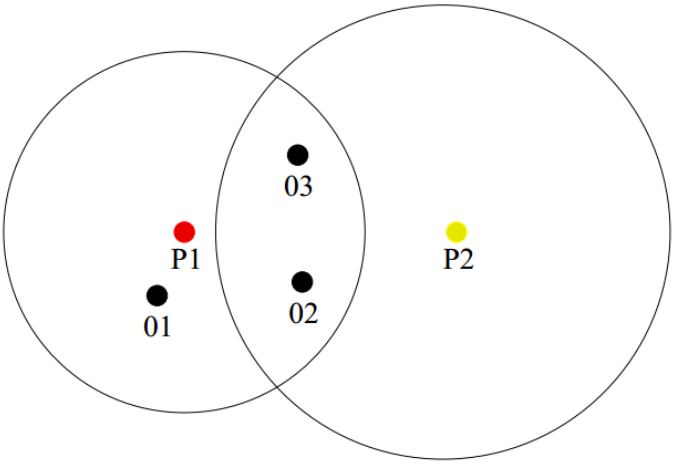


FIGURE 4.22 – Exemple de situation conflictuelle en JPDAF/PDAF

La figure 4.22 monte une situation conflictuelle pour comprendre le principe du *JPDAF*. Deux pistes sont formées et poursuivies dans cet exemple. P1 et P2 sont les états prédits de ces deux pistes à un instant donné. Les cerclent représentent les deux fenêtres de validation et on constate qu'elles se chevauchent partiellement. A ce même instant, le capteur envoie trois observations. La première est située dans la fenêtre de validation de la première piste uniquement, les deux autres sont situées dans la zone commune aux deux fenêtres de validation. Comme les trois observations sont dans la fenêtre de la première piste, la mise à jour de cette dernière se fera en tenant compte de ces trois observations, sous forme d'une somme pondérée par la probabilité d'association. Toutefois, la probabilité d'association, et donc le poids, des observations O2 et O3 sera réduite pour tenir compte de leur présence dans la fenêtre commune aux deux pistes. La mise à jour de P2 se fera en tenant compte des deux observations O2 et O2.

Création et destruction de pistes : Récemment, le *JPDA* a été amélioré pour tenir compte des variations du nombre d'objets M et donc du nombre de pistes à poursuivre. La création d'une nouvelle piste est envisagée pour les observation ayant obtenues une faible probabilité d'association avec une piste existante. Pour chaque piste, et pour

chaque tentative de nouvelle piste, la probabilité que cette piste corresponde à un objet existant est estimée. La destruction de pistes existantes et la confirmation de nouvelles pistes sont décidées par un seuillage sur cette probabilité.

- *Multi-Hypothesis Tracker (MHT) :*

  Un autre algorithme classique est le *MHT*. Le nombre d'objets est supposé supérieur à un mais non connu *a priori* et pouvant croître au cours du temps. Le principe est de considérer systématiquement toutes les hypothèses d'association possibles et de créer une piste pour chacune de ces hypothèses. L'idée est d'attendre que de nouvelles observations capteurs lèvent l'ambiguïté et permettent de décider quelles hypothèses sont à abandonner et quelles hypothèses sont à garder.

  Par exemple, si on reprend la situation de la figure 4.22, dix hypothèses d'association sont à retenir pour le *MHT* contre deux seulement pour le *JPDAF*. Les dix hypothèses vont être maintenues et le nombre de pistes va donc passer de deux à dix. L'ensemble des associations possibles au cours du temps est souvent représenté sous forme d'arbre. A chaque branche est associé une valeur de probabilité d'existence. La taille de cet arbre croît exponentiellement avec le nombre d'observations du capteur. On constate que cette méthode est inexploitable tel quel. Il est nécessaire d'élaguer les branches de l'arbre dont les probabilités sont trop faibles. Plus le nombre de pistes supprimées est élevé, plus la complexité de l'algorithme est réduite. Il est cependant fatal d'éliminer trop rapidement une hypothèse de probabilité à un instant donné qui se serait révélée intéressante par la suite.

Discussion : Sur la base de ce qui a été dit précédemment et au vu des expériences menées et de l'implémentation du BOF, on peut conclure que ce dernier permet d'éviter pas mal de situations conflictuelles. C'est d'ailleurs son rôle.
D'abord les hypothèses considérées par les méthodes d'association de données sont très discutables. La situation d'objet momentannément à l'arrêt n'est pas envisagée contrairement à l'algorithme *Fast Classification of static and dynamic environment* implémenté dans le BOF.

Dans des environnements encombrés, le choix du seuil de création/suppression de pistes est un problème délicat. La taille de fenêtrage l'est encore plus. Le *MHT* évite ce dernier mais son coût calculatoire ne permet pas de garder toutes les pistes possibles. Enfin, le BOF ne tient pas compte des pistes ou des objets. C'est un algorithme de filtrage et de lissage qui impose une autre modélisation de la scène et raisonne sur les antécédents d'une cellule occupée à un instant donné. Tout comme le MHT, tenir compte de beaucoup d'antécédents serait trop coûteux en terme de temps de calcul. Un compromis doit être trouvé entre précision et complexité. Toutes les méthodes de DATMO fournissent seulement des estimations et des approximations des variables pertinentes choisies. Ces approximations sont suffisantes pour que le véhicule évolue sereinement dans son environnement. L'algorithme du BOF ne commet pas d'erreur d'association d'objet, de création ou suppression de piste, et permet de générer des estimations de vitesse pour toute cellule détectée dynamique.

4.6 Exploitation du DATMO en planification de trajectoire

Dans cette partie, on relie ce qui a été fait précédemment à la planification de trajectoire pour le Robucar. On rappelle qu'on est parti du robot qui scrutait la scène sans différencier les obstacles statiques des dynamiques. Le module de DATMO lui permet désormais de faire la distinction entre ces deux classes et le module de SLAM en bénéficie. De plus, une estimation des vélocités est obtenue grâce au BOF. La connaissance des vitesses, même approximatives, est utile pour le module de planification de trajectoire. C'est ce principe qu'on va illustrer dans cette dernière section.

4.6.1 Planification de trajectoire par l'algorithme A star

L'un des plus fameux algorithmes d'intelligence artificielle et de recherche de chemins est le A star (ou simplement A*). On a choisi de l'associer à l'interface "W_DATMO" pour son efficacité, sa simplicité et les excellentes performances en temps de calcul. L'algorithme est parfaitement implémentable sur la grille d'occupation et peut exploiter directement les données fournies par le module de DATMO. Il s'agit là d'un autre avantage et pas des moindres pour le BOF en comparaison à l'approche classique du DATMO.

En partant d'une case de départ, l'algorithme A* cherche à se rapprocher à chaque itération de sa destination. Les chemins qui l'éloignent ou qui ne sont pas optimaux sont automatiquement écartés mais pas supprimés. En effet, si un chemin n'aboutit pas à cause d'obstacles ou d'une impasse alors le A* examinera d'autres possibilités. La capacité de l'algorithme à reconsidérer d'anciennes pistes garantit la convergence vers la solution possible, quand elle existe.

❑ Stratégie de l'algorithme A*

La distance qui sépare la case de départ de la case d'arrivée est exprimée par une fonction heuristique notée habituellement h. Dans le cas particulier de la grille d'occupation, si la case de départ c_1 a pour coordonnées $(e_1,\ m_1)$ et si la case d'arrivée c_f a pour coordonnées $(e_f,\ m_f)$, on peut considérer l'heuristique qui somme la différence entre les indices de lignes et les indices de colonnes. Dans ce cas la valeur de h en c_1 est :

$$h(c_1) = |e_1 - e_f| + |m_1 - m_f|. \tag{4.8}$$

Pour une case $c_i(e_i, m_i)$ quelconque, la valeur de la fonction heuristique est simplement notée $h(c_i)$. Le coût pour atteindre la case courante c_i à partir de la case de départ c_1 est exprimé par la fonction g. Enfin, le coût total de la solution est quantifié par la fonction f. On déduit que

$$f(c_i) = g(c_i) + h(c_i), \tag{4.9}$$

avec

$$g(c_i) = \Sigma_{k=1}^{i-1}\ \text{Coût}(c_k,\ c_{k+1}), \tag{4.10}$$

et pour tout i

$$h(c_i) = |e_i - e_f| + |m_i - m_f|. \tag{4.11}$$

L'algorithme A* est optimal du moment que l'heuristique h est consistante. Ceci veut simplement dire que que le coût estimé $h(c_i)$ ente la case c_i et la case d'arrivée c_f vérifie l'inégalité :

$$h(c_i) \leq \text{Coût}(c_i, c_{i+1}) + h(c_{i+1}). \tag{4.12}$$

La fonction h considérée dans (4.8) vérifie la condition de consistance et donc le Robucar trouvera son chemin optimal dans la grille si ce dernier existe.

❑ Implémentation de l'algorithme A*

Le A* considère deux listes : la *open-list* et la *closed-list*. La deuxième liste contient au premier instant la case que le Robucar considère comme point de départ. Puisque en DATMO la grille est locale (associée au véhicule) on a choisi une case qui est juste à coté du LMS. On a choisi aussi une case quelconque comme case d'arrivée. Maintenant la *open-list* va contenir les cases voisines à la case de départ et que le Robucar va examiner une par une. En fonction de la case d'arrivée et du coût de passage, l'algorithme choisit la case du plus faible coût même si on ne sait a priori si c'est le bon chemin. L'important est que, à cet instant, c'est la case qui rapproche le Robucar le plus de sa destination finale. La meilleure case passe dans la *closed-list*.

A l'itération suivante, la *open-list* va contenir en plus les cases voisines de celle qui vient de passer en *closed-list*. On note que toute case qui a déjà été examinée par l'algorithme et qui n'a pas été retenue dans la *closed-list* doit quand-même figurer dans la *open-list* pour les prochaines itérations. C'est ainsi que le A* pourra revenir sur ses pas et examiner d'autres possibilités si jamais un de ses précédents choix s'est révélé mauvais par la suite. Enfin, l'algorithme s'arrête quand la case finale est atteinte.

Détermination des cellules voisines : Chaque cellule de la grille est entourée par huits autres cellules suceptibles d'être le prochain objectif de planification. Dans l'ordre il faut vérifier que la cellule candidate ne correspond pas à un obstacle ou à une zone à risque. Ensuite il faut aussi vérifier si elle ne fait pas partie de la *closed-list*. Le fait que la cellule figure dans la *open-list* n'est pas problématique. Au contraire, il faut révaluer le chemin maisen changeant aussi le parent. Enfin, si la cellule adjaçante ne figure pas dans le *open-list* alors on l'ajoute et on estime son coût $f(c_i)$.

Parent d'une cellule : Le parent d'une cellule c_i est la cellule par laquelle on accède à celle-ci dans le chemin planifié. Il est important d'organiser ces parents pour retrouver le chemin en suivant à chaque fois les parents des noeuds présents dans la *closed-list* et on remonte le fil jusqu'à arriver au point de départ.

❑ Combinaison du BOF et de l'algorithme A*

Chaque seconde, le BOf estime les vitesses des cellules détectées dynamiques à cet instant. Les informations disponibles sont utilisées durant la seconde qui suit. Pour le cas d'obstacles type "piétons", on ne voit pas l'intérêt de vouloir estimer la vitesse plus rapidement même si la cadence du LMS le permet. On imagine que pour la seconde qui suit une estimation (et donc jusqu'à la prochaine estimation de vélocité), le Robucar peut se servir de ce temps

pour planifier une trajectoire en tenant compte : des objets statiques, des objets dynamiques et des vélocités approximées à cet instant. Notre objectif donc est de fournir en entrée à un module de planification une grille complète qui fusionne toutes les informations. La figure suivante illustre la stratégie déployée pour l'élaboration du programme global :

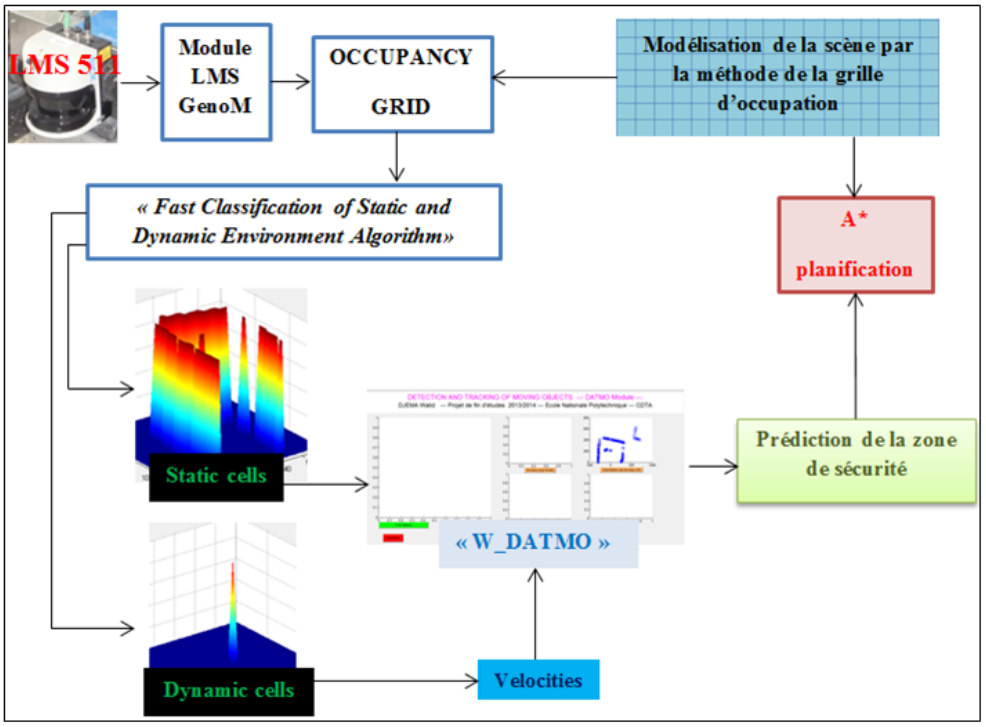


FIGURE 4.23 – Principe d'implémentation du programme global incluant BOF et A*

Selon les vélocités estimées, une zone de la grille ne va pas être considérée lors de l'étape de planification car on soupçonne les objets dynamiques de s'y trouver pour les prochains instants. De mêmel pour les cellules statiques, on prévoit une distance de sécurité pour les manoeuvres du Robucar. Ainsi, le véhicule peut planifier son trajet de manière sûre. On reprend le sénario précédent pour tester l'algorithme élaboré. A un instant donné le BOF fournit une estimation des vélocités des cases occupées. L'algorithme se sert surtout de l'orientation estimée pour éliminer un certain nombre de cellules orientées dans cette direction.

La figure 4.24 montre la fonction heuristique h à l'instant initial. La cellule $(e = 28, m = 29)$, voisine au LMS, est toujours considérée comme cellule de départ par le A*. Pour cet exemple seulement, on considère la cellule d'arrivée $(e = 11, m = 32)$ qui est un bon objectif pour illuster l'interaction BOF-A*. La couleur rouge indique que la valeur de h pour la case considérée est relativement importante. En revanche, la couleur bleu indique une valeur moins importante. On voit les creux ($h = 0$ en bleu foncé) indiquant que les cellules occupées (objets statiques ou dynamiques) ne sont pas considérées par cette fonction. Il est de même des zones interdites par précaution. Pour le A*, ceci se traduit par une impossibilité d'intégrer ces cellules dans la *open-list* ou la *closed-list*.

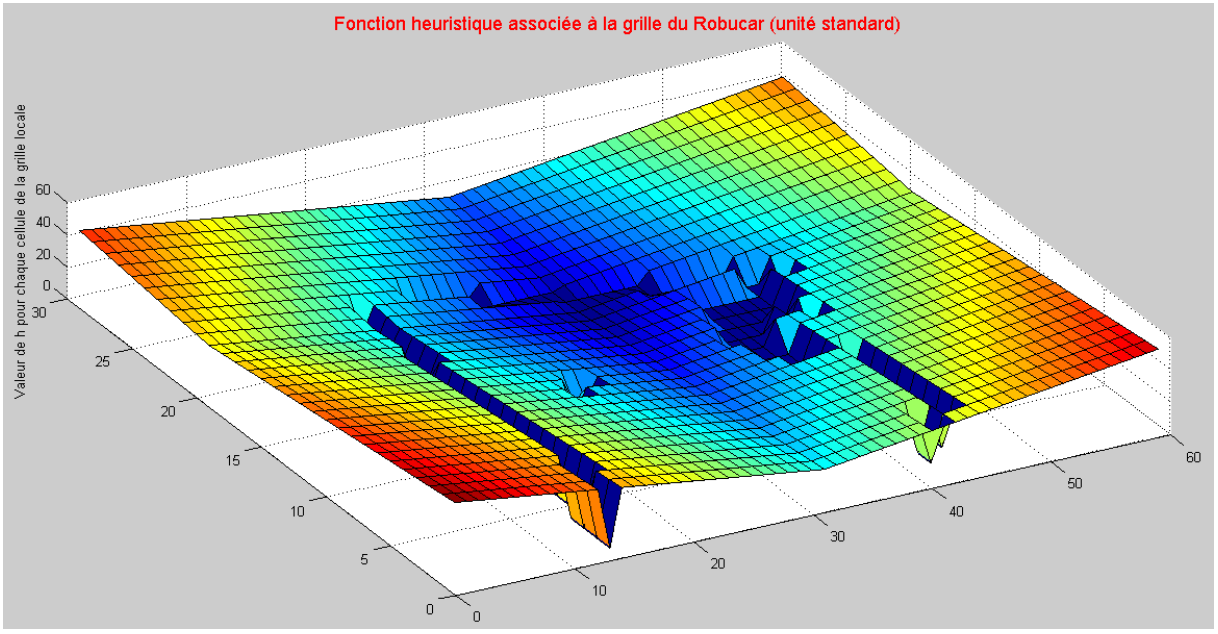


FIGURE 4.24 – heuristique du Robucar à un instant t donné de l'expérience 3

Les figures 4.25 et 4.26 montrent les grilles transmises par le module DATMO à l'algorithme de planification pour différents degrés de précaution. Selon la taille du Robucar, les contraintes holonomiques de ce dernier ou la fiabilité de l'estimation de vitesse, on peut prévoir des zones de sécurité plus ou moins conservatives.

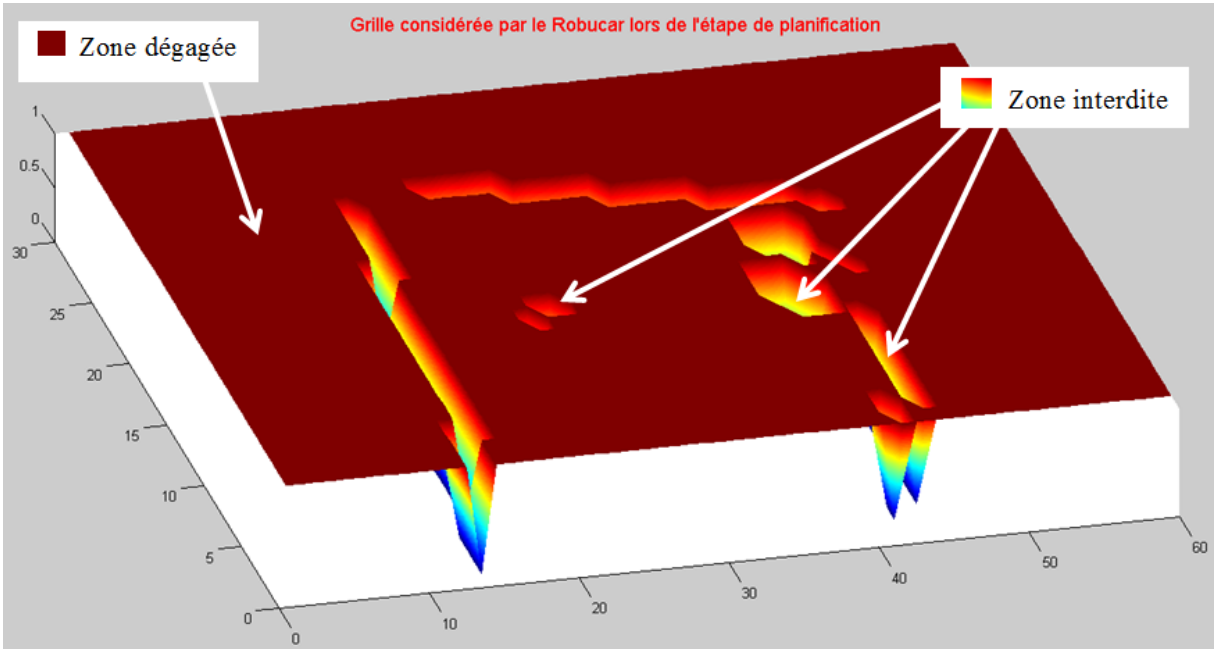


FIGURE 4.25 – Grille transmise par le BOF au module de planification pour une précaution modérée

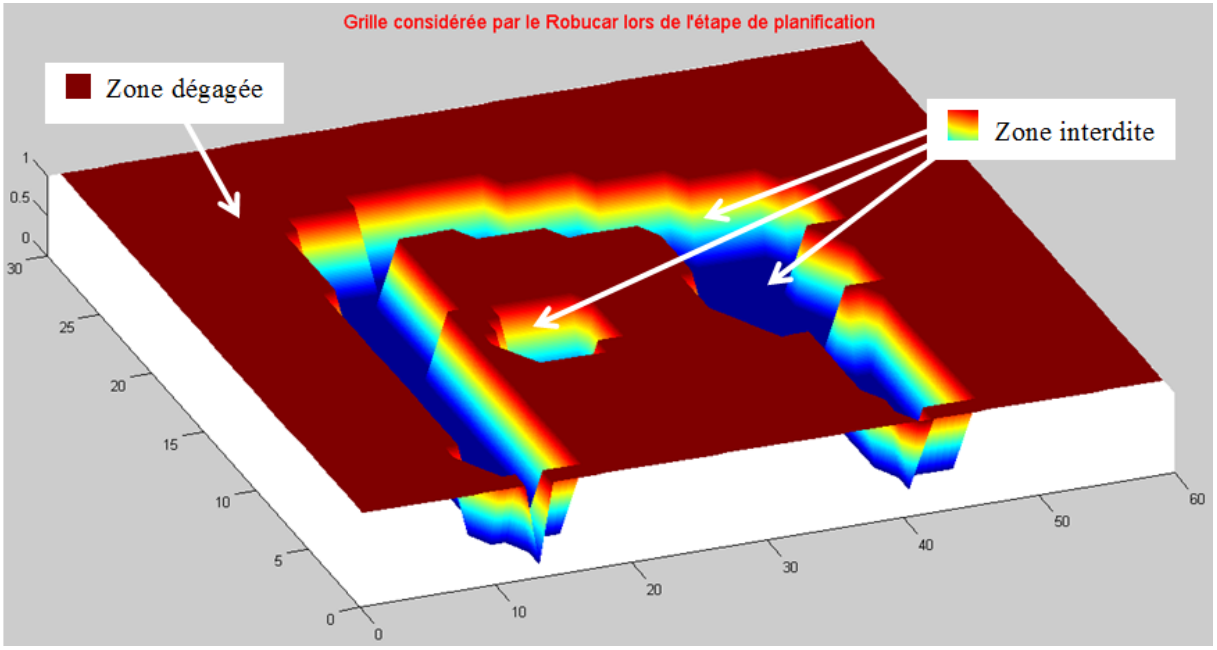


FIGURE 4.26 – Grille transmise par le BOF au module de planification pour une précaution exagérée

La figure ci-dessous illustre la planification de chemin par le A* sur la grille transmise par le BOF à l'instant précédent :

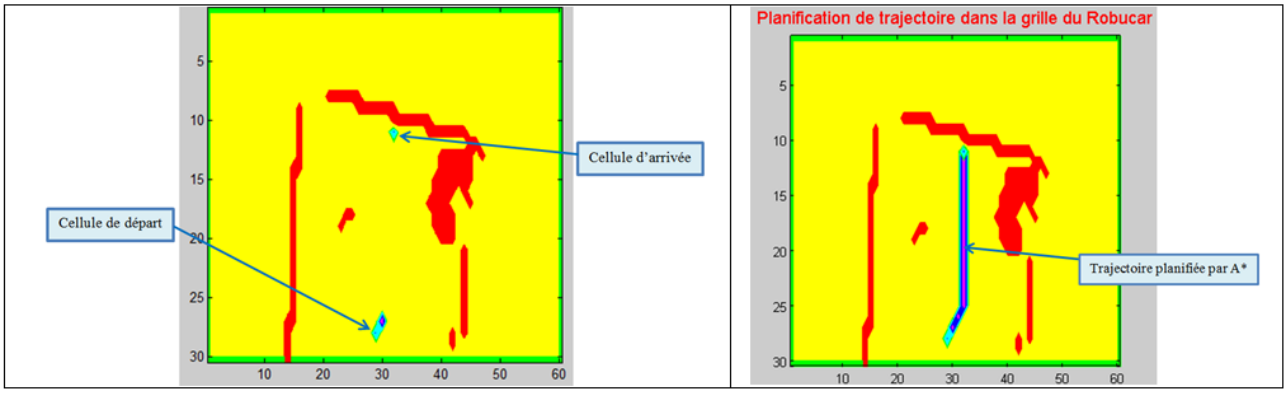


FIGURE 4.27 – Résultat de la planification de trajectoire par A*

Par un fonctionnement alternatif entre BOF et A*, le Robucar planifie ses trajectoires, instant par instant, sans se projeter dans des situations compliquées ou impossibles. Un dernier exemple de planification est donné sur les figures suivantes où, à un instant différent, le BOF transmet une grille qui tient compte des vélocités estimées : Sans elles, l'algorithme A* classique planifie une trajectoire optimale mais problématique pour le Robucar.

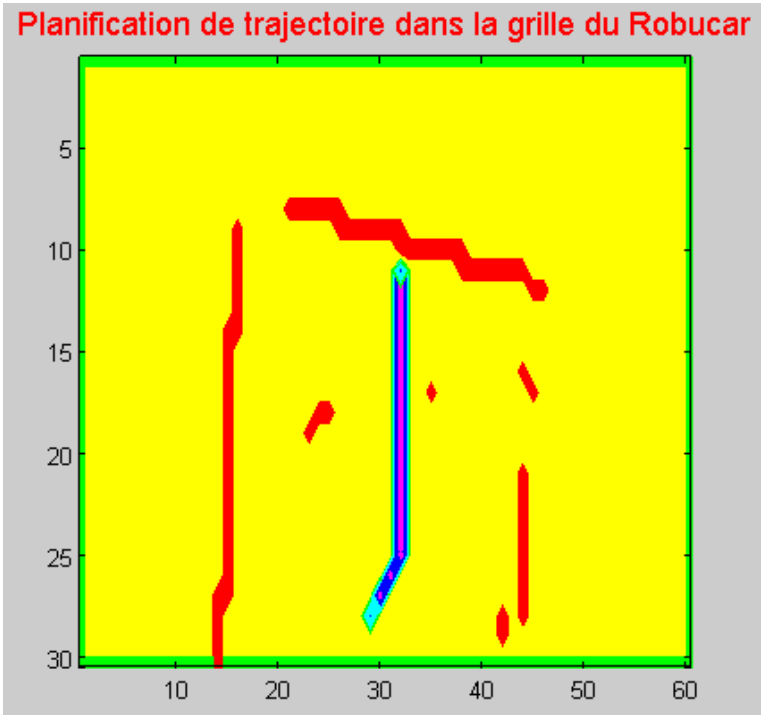


FIGURE 4.28 – Planification A* sans tenir compte du BOF

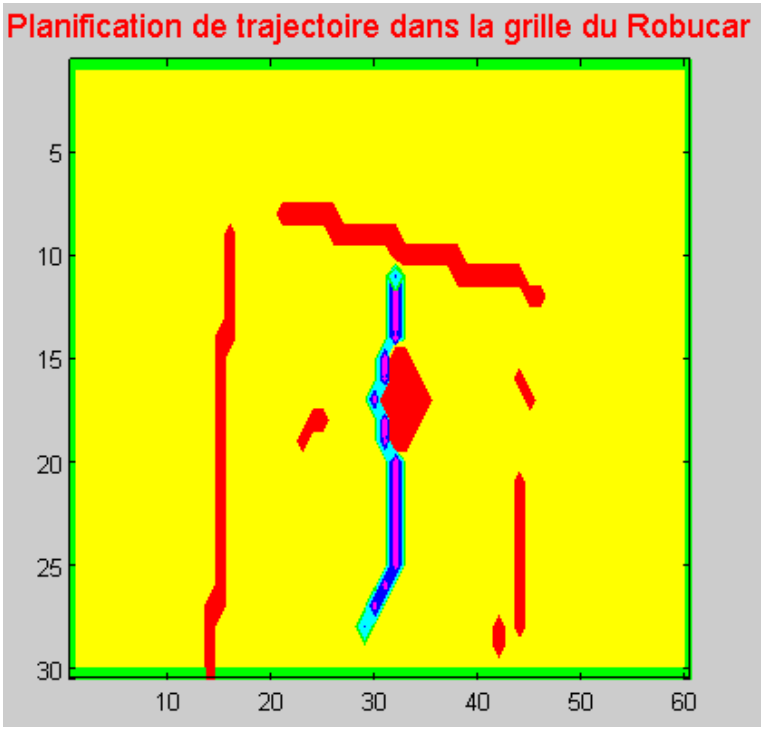


FIGURE 4.29 – Planification A* en sachant qu'un objet dynamique existant

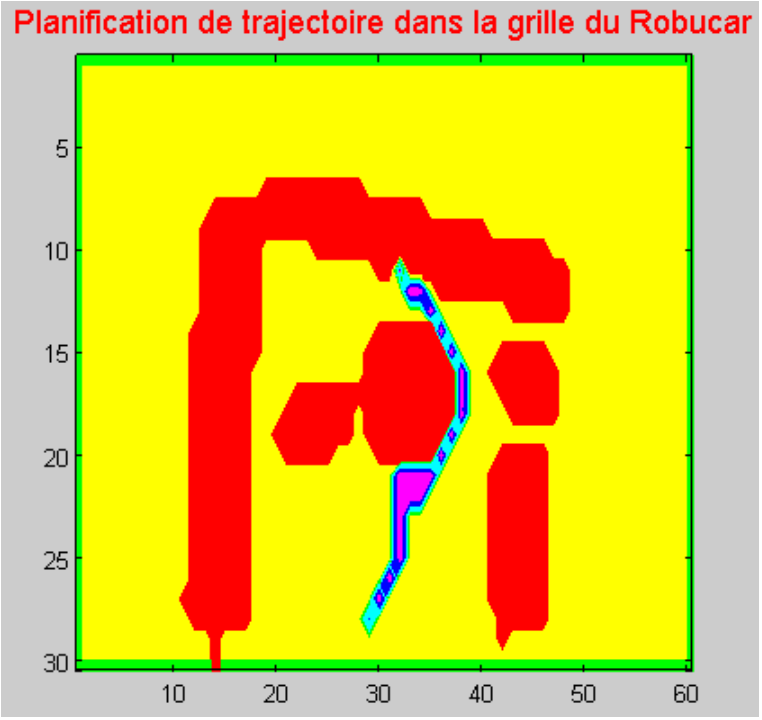


FIGURE 4.30 – Planification A* en tenant compte des précautions et de l'estimation du BOF

4.7 Conclusion

Le Robucar à l'origine n'arrivait pas à distinguer les objets statiques des dynamiques et l'expérience de l 'ICP-SLAM a révélé que cela nuit aussi bien au module de planification que celui de SLAM.

Dans ce chapitre, on a pu implémenter un module de DATMO basé sur les grilles d'occupation qui, dans un premier temps, a doté le Robucar d'un moyen fiable de séparer les objets statiques et dynamiques. Le module de SLAM en a bénificé. Puis le BOF a résolu les problèmes liés à la planification et des expérimentations on a montré comment les deux modules (DATMO et planification) collaborent pour assurer la sécurité du véhicule autonome.

Comparé à l'approche classique du DATMO, le BOF n'apporte que des bénéfices : la convergence vers un résultat, même approximatif, est toujours garantie. L'implémentation de l'algorithme de séparation statique/dynamique est très attractive en terme de temps de calculs. La détection d'objets momentannément à l'arrêt est assurée. Le grid-based SLAM est accompli. Le fait de disposer d'une représentation de l'environnement (qui est la grille) permet d'exploiter les estimations de vélocité de façon instantanée par le module de planification. Ce qui n'est pas le cas de l'approche classique où une étape supplémentaire doit intervenir.

Conclusion générale et perspectives

Dans quelques années, les critères d'autonomie décisionnelle d'une voiture, remplaceront les critères de confort ou de puissance qui distinguent les voitures d'aujourd'hui. On peut imaginer deux axes qui sous-tendront les recherches, celui de la responsabilisation sociétale (un maximum de sécurité) et celui plus pragmatique de l'enjeu économique.

C'est dans ce sens que le travail a été orienté. Pour le Robucar, le but était de le doter d'un maximum d'autonomie. L'approche « F+I » caractérisant la logique classique s'arrête à la programmation des robots manipulateurs où la connaissance de l'environnement est parfaite. Il se trouve qu'elle n'est plus applicable en robotique mobile où les environnements sont complexes et souvent non prédictibles. La logique probabiliste est une théorie alternative des sciences cognitives qui permet au véhicule mobile d'interagir avec son environnement. La quantité très importante de travaux ou articles qui traitent de la programmation bayésienne des robots ces quinze dernières années montre le succès de cette approche.

La partie expérimentale commence par l'approche «F+D» appliquée au Robucar. Grâce à l'approche bayésienne, on a montré comment le Robucar, qui perçoit son environnement à travers un capteur LMS bruité et des données imprécises, peut finalement arriver à prendre des décisions rationnelles. Le modèle probabiliste du capteur élaboré est surtout utile en fusion de données. Quant au principe de filtrage bayésien, il est appliqué dans divers domaines relatifs à la robotique mobile autonome telle que la localisation. Il a été appliqué ici au DATMO uniquement. Et puisque on ne définit pas mieux le DATMO que par le SLAM, on a enchainé sur la technique ICP-SLAM : l'expérience réalisée en SLAM a montré les limites de cette dernière dans des environnements dynamiques. En revanche, on a vu que la localisation ICP donne généralement de bons résultats à partir du moment où les objets dynamiques ne sont pas nombreux. Le chapitre 2 a couvert le fondement théorique ainsi que les principales techniques du DATMO. Par la suite, l'implémentation du module DATMO au chapitre 3 s'est faite sur la base de la plus récente approche de résolution : celle basée sur la grille d'occupation. L'introduction de cette dernière est due aux problèmes de *«data-association»* rencontrées habituellement en approche classique. On a donc commencé par la modélisation de l'environnement : le choix de la taille de la grille et de la cellule s'est fait en s'appuyant sur des travaux similaires dans ce domaine. L'application par la suite de l'algorithme *«Fast classification of static and dynamic environment»* a permis d'isoler les environnements statiques et dynamiques et a donné de très bons résultats sur des données et des expériences réelles. De par sa simplicité, cet algorithme est attractif pour les applications temps réel où les exigences en puissance et temps de calcul sont primordiales. De plus, il est particulièrement adapté à la détection d'objets momentanément à l'arrêt : un très grand avantage par rapport à l'approche classique où cette classe d'objets pose une difficulté majeure. Enfin, ce résultat

a été directement exploitable par le *«grid-based SLAM»* en *«mapping»* d'environnements statiques uniquement. Il s'agit donc d'une représentation alternative de l'environnement du Robucar.

Dans un second lieu, l'application du *«Bayesian Occupancy Filter»* a permis de donner une estimation des vélocités des cellules dynamiques par rapprochement du filtre bayésien à la modélisation par grille d'occupation. Les distributions probabilistes de vélocité sont apprises sur la base de connaissances préalables et des expériences cumulées par le Robucar. Comme dans toutes les approches «F+D», les réponses proposées sont approximatives mais souvent suffisantes au robot pour interagir avec son environnement. Ainsi, le BOF a permis au Robucar de comprendre, interpréter et décider de façon rationnelle. La décision lui permet par la suite d'agir dans son environnement ce qui a été illustré par l'association du module de DATMO complet à la planification de trajectoires par A*. En effet, dans la dernière partie du mémoire, on a exploité le DATMO dans le but d'une planification efficace. Un autre avantage de l'utilisation de cette récente approche DATMO est justement sa capacité à exploiter l'estimation de vitesse directement par les autres fonctionnalités de la robotique autonome, dont la planification, grâce à la représentation de la scène par une grille.

Toujours sur des données expérimentales réelles, l'implémentation du «BOF complet» et du «BOF-A*» a donné de bons résultats. Il resterait à tester ces algorithmes sur plusieurs autres scénarii pour obtenir une estimation significative de la fiabilité du module DATMO que l'on propose. C'est la perspective à court terme envisagée. L'utilisation d'un second capteur laser ou d'une caméra améliorerait considérablement la fiabilité du Robucar. Enfin, ce travail a été une première pour ce véhicule en programmation bayésienne. On espère augmenter le nombre de modules implémentés sur le véhicule et opter pour cette approche notamment en localisation.

Bibliographie

[1] Ilya A. Volfson, Jeffrey A. Stock *Urologic robotic surgery.* A product of Humana Press. 240p. 2008.

[2] D. Daney *Cours de robotique fondamental.* Projet Coprin INRIA Sophia Antipolis. 2008.

[3] P. Coiffet *Robots industriels : concepts, définitions et classifications.* Techniques de l'ingénieur, s7700. 2007.

[4] A. PRUSKI *Robots mobiles autonomes.* Techniques de l'ingénieur, r7850. 1998.

[5] A. Pancham, N. Tlale, G. Bright *Robots mobiles autonomes.* Presented at the 4th Robotics and Mechatronics Conference of South Africa (ROBMECH 2011) 23-25 November 2011, CSIR Pretoria South Africa.

[6] B. BAYLE, publication personelle, *Robotique mobile.* Télécom Physique Strasbourg 76p.

[7] D. FILLIAT, publication personnelle *Robotique mobile.* Ecole Nationale Supérieure de Techniques Avancées ParisTech.175p. 2007.

[8] N. Morette *Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive.* thèse de doctorat. 2009.

[9] C. Mertz, L. E. Navarro-Serment, R. MacLachlan *Moving Object Detection with Laser Scanners.* Journal of Field Robotics, 1–27p. May 2012.

[10] A. Alfes *Using Occupancy Grids for Mobile Robot Perception and Navigation.* IEEE Computer, Special Issue on Autonomous Intelligent Machines, Juin 1989.

[11] L. Montesano *Detection and tracking of moving objects from a mobile platform. Application to navigation and multi-robot localization.* Ph.D Dissertation. 243p. Zaragoza university. February 2006.

[12] Stéphanie Lefevre, Ruzena Bajcsy, Christian Laugier *Probabilistic decision making for collision avoidance systems : Postponing decisions.* IROS 2013 : 4370-4375

[13] R. Labayrade, D. Gruyer, C. Royere *Obstacle Detection Based on Fusion Between Stereovision and 2D Laser Scanner.* Mobile Robots : Perception and Navigation, Book edited by : Sascha Kolski, ISBN 3-86611-283-1, pp. 704, February 2007, Plv/ARS, Germany

[14] L. Montesano *ADAS for the Car of the Future : Interface Concepts for Advanced Driver Assistant Systems in a Sustainable Mobility Concept of 2020* Faculty of Engineering Technology / Industrial Design University of Twente. April/June 2006.

[15] Cheih-CHeih Wang, Sebastian Thrun *Simultaneous Localization, Mapping and Moving Object Tracking* Carnegie Mellon University Research Showcase @ CMU Robotics Institute School of Computer Science. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Washington, DC.

[16] Wang, C.-C., Thorpe, C., and Suppe, A. *Ladar-Based Detection and Tracking of Moving Objects from a Ground Vehicle at High Speeds.* In Proceedings of the IEEE Intelligent Vehicles Symposium, Columbus, OH. 2003.

[17] Pierre Bessiere and Olivier Lebeltel *Basic Concepts of Bayesian Programming* Prob. Reason. Deci. Mak., STAR 46, pp. 19–48, 2008. Springer-Verlag Berlin Heidelberg 2008.

[18] Hans P.Moravec *Sensor Fusion in Certainty Grids for Mobile Robots* All Magazine Volume 9 Number 2 AAAI. 1988.

[19] Anna Petrovskaya, Sebastian Thrun *Model Based Vehicle Tracking in Urban Environments* Workshop on Safe Navigation. ICRA 2009.

[20] Anna Petrovskaya, Sebastian Thrun *Credibilist Occupancy Grids for Vehicle Perception in Dynamic Environments* IEEE International Conference on Robotics and Automation, May 9-13, Shanghai, China, 2011.

[21] Edouard Ivanjko, Ivan Petrovic, Misel Brezak *Experimental Comparison of Sonar Based Occupancy Grid Mapping Methods* ISSN 0005—1144 ATKAAF 50(1—2), 65—79 (2009)

[22] Edouard Ivanjko, Ivan Petrovic, Misel Brezak *Awareness of Road Scene Participants for Autonomous Driving* A. Eskandarian (ed.),Handbook of Intelligent Vehicles, DOI 10.1007/978-0-85729-085-4_54, Springer-Verlag London Ltd. 2012

[23] Michel Devy, David Marquez-Gamez *SLAM visuel avec détection et suivi d'objets mobiles par une approche de segmentation/classification* Manuscrit auteur, publié dans "RFIA 2012 (Reconnaissance des Formes et Intelligence Artificielle), Lyon : France (2012).

[24] Anna Petrovskaya *Model Based Vehicle Tracking for Autonomous Driving in Urban Environments.* RSS 2008, Zurich, Switzerland.

[25] O. Aycard, T. D. Vu, Q. Baig *An Occupancy Grid Based Architecture for ADAS.* Université of Grenoble, Safety and Driver Assistance p189-210.

[26] Nadeen SALAMEH *Conception d'un système d'alerte embarqué basé sur les communications entre véhicules* Thèse de doctorat, INSA de Rouen, 150p, 2012.

[27] Robert Stengel *Linear-Optimal Estimation.* Optimal Control and Estimation MAE 546, Princeton University, 2013.

[28] James V Stone *Bayes' Rule : A tutorial introduction to bayesian analysis.* Cover Design by Stefan Brazzo. Third printing. ISBN 978-0-9563728-4-0. First Edition, 2013.

[29] Project-Team E-MOTION *Geometry and Probability for Motion and Action* IN COLLABORATION WITH : Laboratoire d'Informatique de Grenoble (LIG). Activity Report 2012.

[30] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier *Dynamic Obstacle A voidance in uncertain environment combining PVOs and Occupancy Grid* 2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.

[31] E. T. Jaynes *PROBABILITY THEORY THE LOGIC OF SCIENCE.* Cambridge University Press. Cambridge, New York, 757p. 2003.

[32] Olivier LEBELTEL *Programmation bayésienne des robots.* Thèse de doctorat de l'institut national polytechnique de Grenoble, France, p 262, 1999.

[33] Mark Steyvers, publication personnelle *Computational Statistics with Matlab.* May 13, 2011.

[34] Tobias Gindele, Sebastian Brechtel. *Bayesian Occupancy Grid Filter for Dynamic Environments Using Prior Map Knowledge.* 978-1-4244-3504-3/09/ 2009 IEEE.

[35] Operating instructions, lms datasheet *Laser Measurement Systems of the LMS500 Product Family* SICK Sensor Intelligence 2010.

[36] Kamel Mekhnacha, Yong Mao, David Raulo and Christian Laugier *The "Fast Clustering-Tracking" Algorithm in the Bayesian Occupancy Filter Framework.* Multisensor Fusion and Integration for Intelligent Systems, Lecture Notes in Electrical Engineering 35, DOI 10.1007/978-3-540-89859-715, Springer-Verlag Berlin Heidelberg 2009.

[37] Asma Azim and Olivier Aycard *Detection, Classification and Tracking of Moving Objects in a 3D Environment.* 802 - 807 Intelligent Vehicles Symposium (IV), 2012 IEEE.

[38] Anna Petrovskaya, Oussama Khatib, Sebastian Thrun, Andrew Y. Ng *Bayesian Estimation for Autonomous Object Manipulation Based on Tactile Sensors.* IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida, 2006.

[39] Jean-Michel MARIN *Statistique bayésienne : les bases.* Techniques de l'ingénieur, af605. 2009.

[40] Christian Laugier, Igor E. Paromtchik, Mathias Perrollaz, Yong Mao, John-David Yoder, Christopher Tay, Kamel Mekhnacha, Amaury Nègre. *Probabilistic Analysis of Dynamic Scenes and Collision Risks Assessment to Improve Driving Safety.* IEEE Intell. Transport. Syst. Mag. 3(4) : 4-19 (2011)

[41] Eva SIMONIN *Carte bayésienne et apprentissage.* Ecole Doctorale "Mathématique, sciences et technologie de l'information, informatique", Université Joseph Fourier, 2004.

[42] Wang, C.-C. and Thorpe, C. *Online simultaneous localization and mapping with detection and tracking of moving objects : Theory and results from a ground vehicle in crowded urban areas.* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan. (2004).

[43] Christophe Coué, Cédric Pradalier, Christian Laugier, Thierry Fraichard, Pierre Bessière *Bayesian Occupancy Filtering for Multitarget Tracking : An Automotive Application.* I. J. Robotic Res. 25(1) : 19-30 (2006).

[44] D.Alazard *Introduction au filtre de Kalman.* Publication personnelle, 2005.

[45] Lindsay Kleeman *Understanding and Applying Kalman Filtering.* Department of Electrical and Computer Systems Engineering Monash University, Clayton.

[46] Anna Petrovskaya and Oussama Khatib *Global Localization of Objects via Touch.* IEEE Transactions on Robotics, vol. 27(3), pp. 569–585, June 2011.

[47] Christian Laugier *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems.* p378. Springer-Verlag Berlin Heidelberg.2008.

[48] Qadeer Baig, Mathias Perrollaz, Christian Laugier *A Robust Motion Detection Technique for Dynamic Environment Monitoring : A Framework for Grid-Based Monitoring of the Dynamic Environment.* IEEE Robot. Automat. Mag. 21(1) : 40-48 (2014).

[49] Marcelo Becker and Richard Hall, *2D Laser-based Probabilistic Motion Tracking in Urban-like Environments.* J. of the Braz. Soc. of Mech. Sci. and Eng, April-June 2009.

[50] Christophe COUE *Modèle bayésien pour l'analyse multimodale d'environnements dynamiques et encombrées : Application à l'assistance à la conduite en milieu urbain.* Thèse de doctorat de l'Ecole doctorale de Grenoble. 2003.

[51] Christophe Coué, Cédric Pradalier, Christian Laugier, *Fast classification of static and dynamic environment for Bayesian Occupancy Filter (BOF).* ICARCV 2012 : 656-661.

[52] Olivier Lebeltel, Emmanuel Mazer, *Bayesian Robot Programming.* Autonomous Robots 16, 49–79, 2004 Kluwer Academic Publishers. Manufactured in The Netherlands. 2004.

[53] Cheng Chen, Christopher Tay, Christian Laugier, Kamel Mekhnacha *Dynamic Environment Modeling with Gridmap : A Multiple-Object Tracking Application.* ICARCV 2006 : 1-6

[54] Juan David Adarve, Mathias Perrollaz, Alexandros Makris and Christian Laugier *Computing Occupancy Grids From multiple Sensors Using Linear Opinion Pools* IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA May 14-18, 2012

[55] Olivier Aycard, Qadeer Baig *Improving Moving Objects Tracking Using Road Model For Laser Data* Intelligent Vehicles Symposium Alcalá de Henares, Spain, June 3-7, 2012.

[56] Qadeer Baig, Mathias Perrollaz, Jander Botelho Do Nascimento, Christian Laugier *Using fast classification of static and dynamic environment for improving Bayesian occupancy filter (BOF) and tracking.* ICARCV 2012 : 656-661.

[57] BAYLE Bernard *Matériels courants en robotique mobile*, Publication personnelle, Chapitre 4 ,18p. ENSPS Université de Strasbourg.

[58] Anna Petrovskaya and Sebastian Thrun. *Efficient Techniques for Dynamic Vehicle Detection.* ISER, Athens, Greece 2008.

[59] Jorge Rios-Martinez, Alessandro Renzaglia, Anne Spalanzani, Agostino Martinelli, Christian Laugier *Navigating between people : A stochastic optimization approach.* ICRA 2012 : 2880-2885.

[60] Sara Fleury et Matthieu Herrb, *Manuel d'utilisation de GenoM*, 2003.

[61] Boreux, Jean-Jacques, Parent, Eric, Bernier, Jacques, *Pratique du calcul bayésien*, Collection Statistique et probabilités appliquées. Springer 2010, XXIV, 335 p.

[62] R. Tiar, N. Ouadah, O. Azouaoui *ICP-SLAM Methods Implementation on a Bi-steerable Mobile Robot.*

[63] Pierre Bessière Eric Dedieu Olivier Lebeltel Emmanuel Mazer Kamel Mekhnacha *Interprétation ou Description : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs*, Rapports du Laboratoire LEIBNIZ - CNRS. LAAS Toulouse.

[64] Olivier Lebeltel, Pierre Bessière, Julien Diard, Emmanuel Mazer *Programmation bayésienne des robots* Rapport Laboratoire GRAVIR / IMAG INRIA Rhône-Alpes ZIRST – 655 Av. de l'Europe 38330 Montbonnot St Martin. 39p. 2001.

[65] Michael Rubinstein *Introduction to recursive Bayesian filtering* Compaq Systems Research Center ICCV 2001.

[66] Wolfram Burgard *Recursive Bayes Filtering* Advanced AI - tutorials and personal publications of the author.

[67] Kamel Mekhnacha, Yong Mao, David Raulo, Christian Laugier *Bayesian Occupancy Filter based "Fast Clustering-Tracking" algorithm.* IROS 2008 2nd Workshop : Planning, Perception and Navigation for Intelligent Vehicles. 2008.

[68] Paul Checchin, Samuel Gidel, Christophe Blanc *Système de détection de piétons à bord de véhicules : approche partélémètrie laser* LASMEA– UMR6602 CNRS / IUTdeMontluçon, UniversitéBlaise Pascal, Clermont-Ferrand II.

[69] Bradley Efron *Large-Scale Inference Empirical Bayes Methods for Estimation, Testing and Prediction* ims Monographs CAMBRIDGE Cambridge University Press The Edinburgh Building, Cambridge CB2 8RU, UK. 277p, First Edition 2010.

[70] Robert Christian *Le choix bayésien : Principes et pratique* Collection : Statistique et probabilités appliquées. Springer ISBN 978-2-287-31120-8. 2006, XXIV, 638 p.

[71] Renée Veysseyre *Statistique et probabilitées pour l'ingénieur* ISBN-10 : 2100499947. Broché : 475 pages. Collection : Aide-memoire de l'ingenieur. Editeur : Dunod ; Édition : 2e édition (29 septembre 2006)

[72] Sébastien Gambs et Marc-Olivier Killijian *Next Place Prediction using Mobility Markov Chains* MPM12, April 10, 2012, Bern, Switzerland. Copyright 2012 ACM 978-1-4503-1223-3/12/04. 2012.

[73] A. Asahara, A. Sato *Pedestrian-movement Prediction based on Mixed Markov-chain Model.* ACM SIGSPATIAL GIS '11November 1-4, 2011. Chicago, IL, USA Copyright 2011 ACM 978-1-4503-1031-4/11/11. 2011.

[74] Edouard Ivanjko, Ivan Petrovic *Experimental Comparison of Sonar Based Occupancy Grid Mapping Methods* ISSN 0005—1144 ATKAAF 50(1—2), 65—79 (2009)

[75] Jaroslav Machan, Christian Laugier *Intelligent Vehicles as an Integral Part of Intelligent Transport Systems.* ERCIM News 2013(94). 2013.

[76] Wang, C.-C., Lo, T.-C., and Yang, S.-W. *Interacting object tracking in crowded urban areas.* In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). 2007.

[77] Bar-Shalom, Y. and Fortman, T.E. (1988)*Tracking and Data Association* Academic Press.

[78] H. Tazebinte, K. Lameche *Planification et optimisation de trajectoire d'un robot mobile par les algorithmes évolutionnaires.* Mémoire d'ingénieur d'état et de master en Automatique à l'Ecole Nationale Plytechnique. 2012.

[79] Bar-Shalom, Y. and Li, X. (1995)*Multitarget Multisensor Tracking : Principles and Techniques*. YBS Publishing

[80] Tay .C, Mekhnacha K., Chen C., Yguel M., and Laugier C., *An efficient formulation of the bayesian occupation filter for target tracking in dynamic environments* International Journal Of Autonomous Vehicles, vol. 6, no. 1/2, pp. 155–171, 2008